FastBench® for Open Mediation Solutions

Technical White Paper

NETMANSYS Inc.
6021 Morriss Road, Suite 101
Flower Mound, TX 75028-USA

Notice

© Copyright 1998 NETMANSYS Inc. All rights reserved.

This document is protected by copyright and distributed under licenses restricting its use, copying and distribution. No part of this document may be translated or reproduced in any form by any means without the prior written authorization of NETMANSYS Inc. and its licensors. Any specific warranty related to FastBench Suite as to which this material relates is available in NETMANSYS Inc.'s licensing terms and conditions. The information in this document is subject to change without notice.

FastBench® and NETMANSYS® are registered trademarks of NETMANSYS Inc.

NETMANSYS INC. MAKES NO WARRANTY OF ANY KIND WITH REGARD TO FASTBENCH PRODUCTS INCLUDING DOCUMENTATION, AND SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NETMANSYS INC. SHALL NOT BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR ANY DAMAGES, WHETHER DIRECT, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR SPECIAL, IN CONNECTION WITH FASTBENCH PRODUCTS.

NETMANSYS Inc.,
6021 Morriss Road, Suite 101,
Flower Mound, TX 75028

Part no. WHT-W-XX-X-FA-10

Revision A, printed in the USA, September 1998

Restricted Rights Legend

© Copyright 1988, 1989, Carnegie Mellon University.

© Copyright 1988, 1989, Massachusetts Institute of Technology.

© Copyright 1979, 1980, 1983, 1985-1988 The Regents of the University of California.

© Copyright 1994 Hewlett-Packard Company

© Copyright 1996, 1997 Silicon Graphics Computer Systems, Inc.

© Copyright 1997 Moscow Center for SPARC Technology

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

UNIX is a registered trademark in the United States and other countries and is exclusively licensed by X/Open Company Ltd.

Microsoft and Windows NT are registered trademarks of Microsoft Corporation in the United States of America and other countries.

Sun, Sun Microsystems, Solaris, Solstice, SunLink and TMNscript are trademarks of Sun Microsystems, Inc.

CORBA, ORB, the Object Request Broker, IIOP, UML and the OMG Interface Definition Language (IDL) are trademarks or registered trademarks of the Object Management Group, Inc. in the UnitedStates of America and other countries.

All other product names referenced herein are trademarks of their respective companies.

Abstract

The number and complexity of communications networks in today's communication society continue to grow. Telecommunications network operators and Internet service providers are facing the crucial challenge of effectively controlling and monitoring networks employing multiple protocols and heterogeneous resources in order to win customer satisfaction and survive in a consumer-oriented marketplace. Mediation — the translation, interpretation and handling of data between heterogeneous data representations and protocols — has become the buzz word and one of the main objectives of today's network management application development efforts.

This paper provides a technical overview of the FastBench Mediation Development Environment, the NETMANSYS approach to helping telecom and datacom equipment manufacturers, systems integrators, independent software vendors (ISVs) and service providers meet the demand for high-performance mediation solutions. The Mediation Development Environment provides a means of rapidly building open, standards-based, integrated interfaces for the management of heterogeneous network resources.

Contents

Abstract 3
The Challenge 7
The FastBench Solution 9
Mediation WorkShop 11
Mediation Framework 13
Mediation Model 13
Mediation Model Implementation 15
Common Management Interface 16
Key Programmatic Concepts 17
Common Management Services 20
Interface to Managed Objects 21
Dispatcher 21
Common Development Framework 22
Default Mediators 23
SNMP-CORBA Mediator 23
TL1-CORBA Mediator 23
Q3-CORBA Mediator 23
SNMP-Q3 Mediator 24
Benefits 24
Glossary 26
Bibliography 27
For More Information 29

The Challenge

Telecommunications and Internet service providers are facing crucial challenges. The deregulation of the telecommunications market has forced companies old and new to compete in a new, consumer-oriented marketplace. The players in this new market-place need to get the maximum from their networks, operate their infrastructure cost-effectively and provide a new range of services which very often require the deployment of new technologies.

One key way to meet these needs is to have an up-to-date, flexible and scalable operations support system (OSS). However, the use of an OSS based on a single, usually proprietary, protocol only goes halfway to solving the problem of effective network management. The rapid increase in the need to incorporate multiple protocols and heterogeneous resources in what were previously homogeneous, proprietary networks requires the availability of open management interfaces between the OSS and the different network resources. Such management interfaces require an underlying mechanism to *mediate* — translate, interpret and handle data — between the various data representations and protocols. This process is known as *mediation*, and is realized by a mediation application.

The integration of a mediation application in a typical network environment is illustrated in Figure 1.

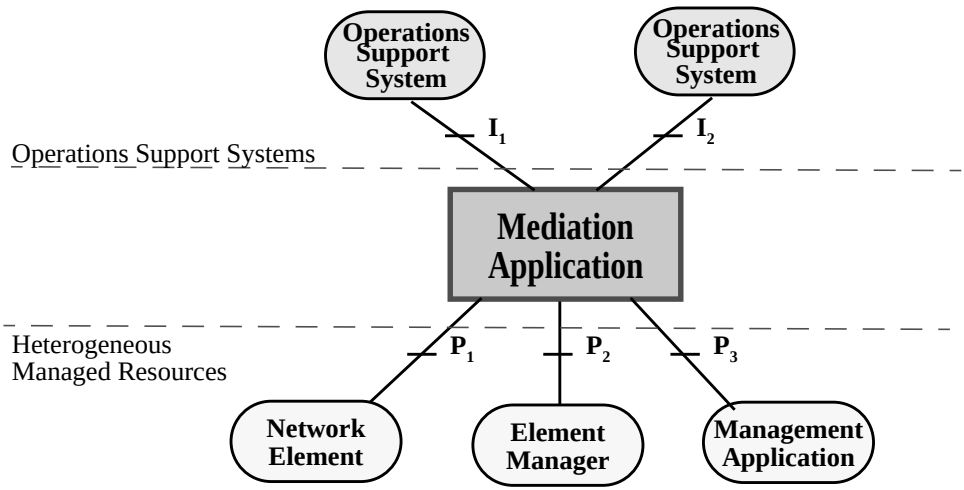

Figure 1 Accessing Heterogeneous Network Resources through Open Management Interfaces

Telecom and datacom equipment manufacturers are under increasing pressure from their customers — telecommunications and Internet service providers — to provide open, standards-based management interfaces. The challenge is therefore to rapidly build scalable, performant and flexible mediation applications which offer open management interfaces on top of various legacy protocols. Such applications will ultimately allow OSS implementors to concentrate on developing value added services (through efficient processing of management information) instead of struggling to recover and reformat network data.

This paper discusses the FastBench Mediation Development Environment, the NETMANSYS approach to helping application developers meet the mediation challenge. The Mediation Development Environment helps OSS builders (service providers, systems integrators and independent software vendors) and telecom and datacom equipment manufacturers to rapidly build open, standards-based and integrated management interfaces to access network resources.

The FastBench Solution

The Mediation Development Environment (MDE) provides a powerful, easy-to-use, object-oriented toolkit for the construction of open CORBA, Q3, web-based or user-defined interfaces to access managed resources. These resources may use a variety of protocols including SNMP, TL1/MML, CMIS/CMIP, bitstream, etc.

The Mediation Development Environment comprises two key components:

- **Mediation WorkShop:** The Mediation WorkShop provides integrated, intuitive graphical tools for application development. The Mediation WorkShop can be seen as the mediation application factory. It covers the whole development cycle, from the initial specification of a management interface model and data mapping through application coding and debugging to testing and deployment.
- **Mediation Framework:** The Mediation Framework provides a set of consistent object-oriented libraries:
  - **Management Protocol Interfaces:** C++ APIs allowing easy access to network resources through SNMP, CMIS/CMIP, bitstream-based, TL1/MML protocols. All of these APIs are designed by NETMANSYS in a consistent way. They share the same, ASN.1/C++ -based data representation and the same programming model. **If you learn one of them, you learn them all.**
  - **Common Management Services:** a set of standard-based, ready-to-use services such as event filtering and reporting, event logging, object persistency, naming, etc.
  - **Default Mediators:** a set of components implementing standards-based and open mappings such as SNMP-Q3, SNMP-CORBA, TL1-CORBA, Q3-CORBA, etc.

Figure 2 gives an example of how a Q3 mediation application is developed using the Mediation Development Environment.

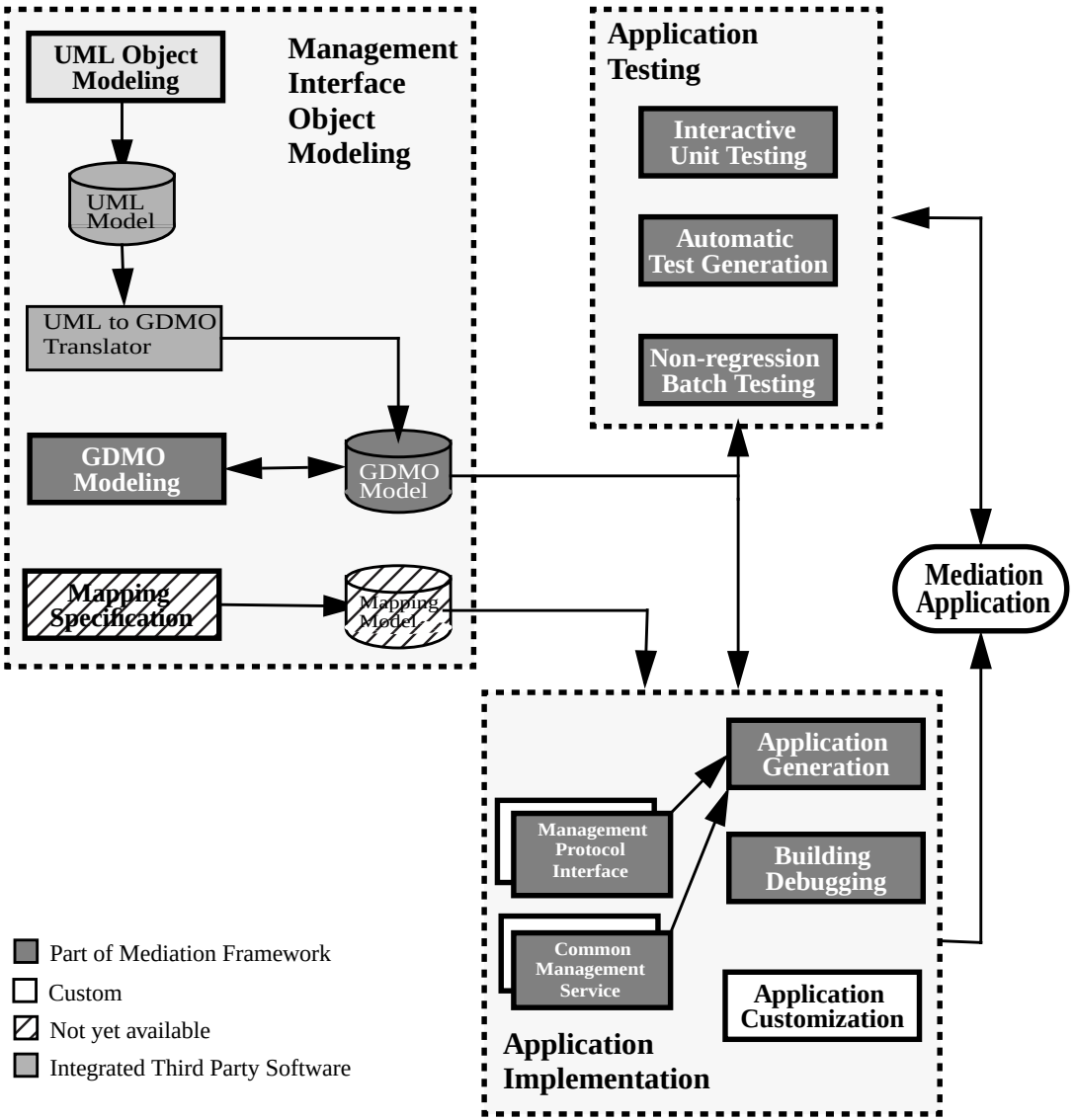

Figure 2 Example of the Development of a Q3 Mediation Application with the FastBench Mediation Development Environment

The following sections discuss each part of the Mediation Development Environment in more detail.

Mediation WorkShop

The Mediation WorkShop provides a graphical environment for mediation application development. Each of the development steps is supported by intuitive graphical tools.

- Managed object classes can be modeled by using FastBench MIM Builder to construct a GDMO model, or by using a Unified Modeling Language (UML) case tool to build an object model which can be used as is or converted to GDMO.
- The mapping between the information model and the managed resource interfaces can be developed using the graphical mapping specification tool.
- Application code can be generated using the Code Generator tool, and then customized via a rich, high-level C++ programming interface.
- Unit and non-regression tests can be designed and run using FastBench Agent Tester.

The grouping of all these tasks in a single development environment means that applications can be developed incrementally, for example by progressively customizing a default mediator provided by the Framework.

The development of an application within the WorkShop is centralized in a development *project*. A single window, the *project panel*, provides a graphical interface to the different project data components and development tools. If several projects are edited simultaneously, each project is assigned its own project panel visible in the environment workspace.

Figure 3 gives an example of how project data is represented in the project panel.

.

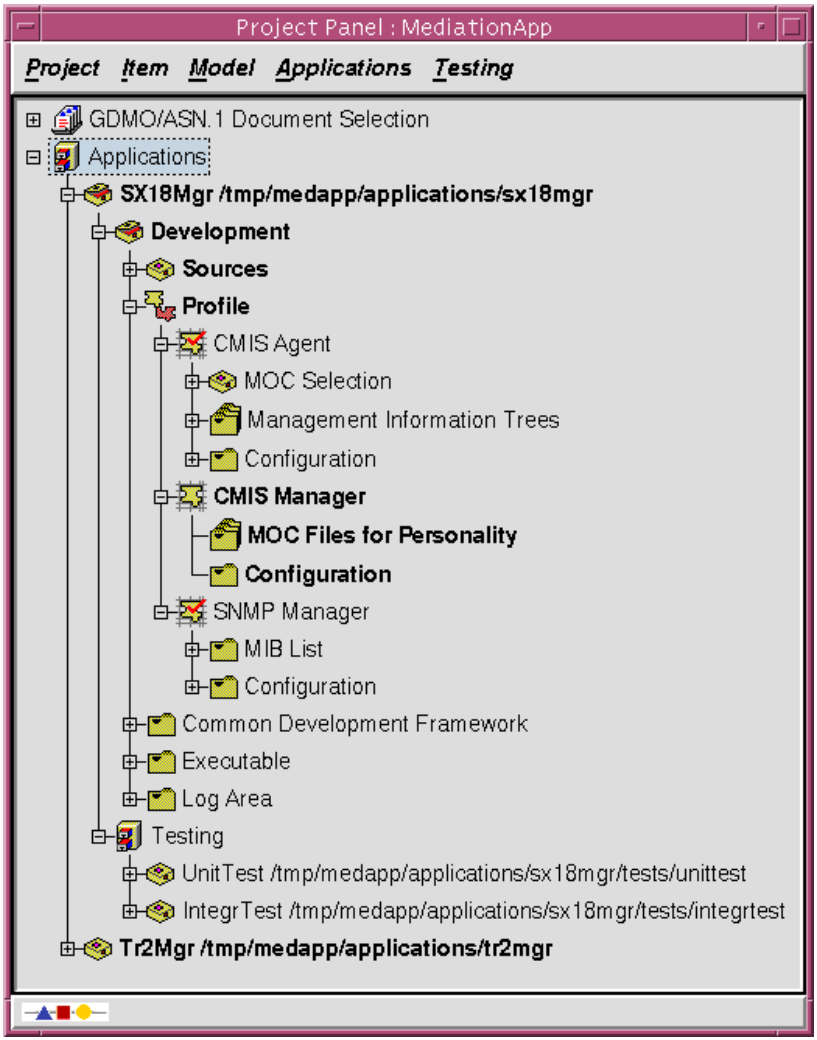

Figure 3 The Mediation WorkShop Project Panel

Mediation Framework

The Mediation Framework provides a set of customizable and integrated C++ libraries allowing rapid development of custom mediation applications.

Mediation Model

A mediation application is built upon a modular, scalable and extensible object-oriented model (see Figure 4). A mediation application can be seen as smart proxy between one or more manager applications (OSS, network or element management systems, etc.) on the top side, and a set of agents on the bottom side. An agent is the software application in the network that supports the management capabilities of a network element, element management system or a management platform. An agent can be standards-based or proprietary, embedded in a network element or run on separate hardware.

Figure 4 shows the mediation model and its different components.

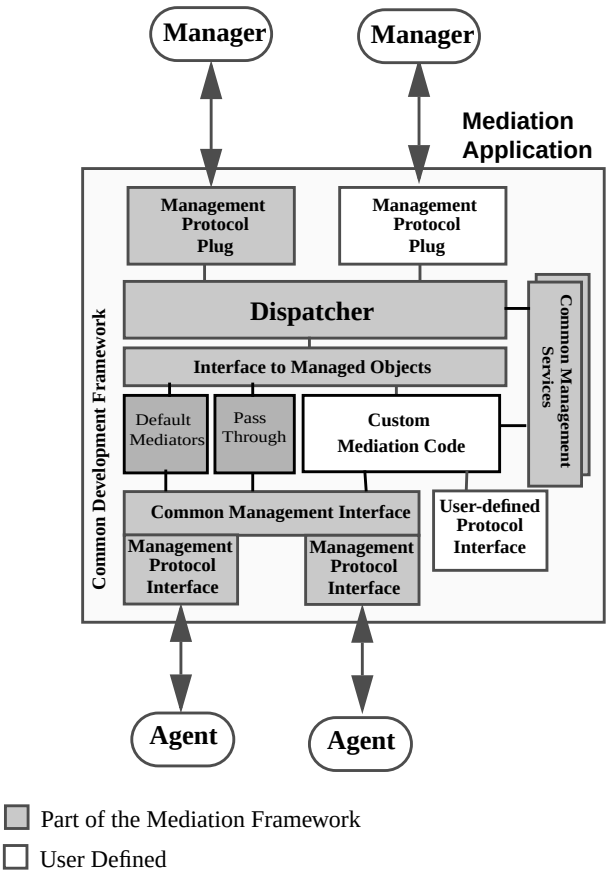

Figure 4 FastBench Mediation Model

The mediation model components are:

- **Management Protocol Plugs**: adapters to plug in management protocols such as CMIP, IIOP and HTTP. They are in charge of translating protocol-specific data and operations and passing them to the Dispatcher's C++ interface.

- **Dispatcher**: dispatches management operations to the managed objects and handles the reporting of asynchronous events.
- **Interface to Managed Objects**: C++ interface used to customize the behavior of managed objects.
- **Common Management Interface**: set of C++ APIs which allow easy access to network resources via a set of management protocol interfaces.
- **Common Management Services**: services such as event reporting, event logging, object naming and the transfer and recovery of objects to and from persistent storage (for example a hard disk or database system).
- **Default Mediators**: ready-to-use mapping modules.
- **Common Development Framework (CDF)**: provides the back-plane for the different Framework components and offers a set of high-level, OS-independent C++ classes for multi-threading, inter-process communication, tracing and logging and the administration console, etc.

Using the mediation model, a management operation sent by a manager application is received at a Management Protocol Plug which passes it to the Dispatcher. The Dispatcher checks the validity of the operation and routes it through the Interface to Managed Objects to the target managed object or objects which trigger the defined behavior. The custom code implements the expected behavior and returns the results of the operation to the Dispatcher, which forwards them to the sending application. Custom code can access the appropriate Common Management Services or use the Common Management Interface APIs to access network resources.

The mediation model allows developers to combine built-in components and extend the application by plugging in user-defined management interfaces, protocols and custom code.

The next section describes how the mediation model is applied to provide a run-time application.

Mediation Model Implementation

Figure 5 illustrates the run-time architecture of mediation application.

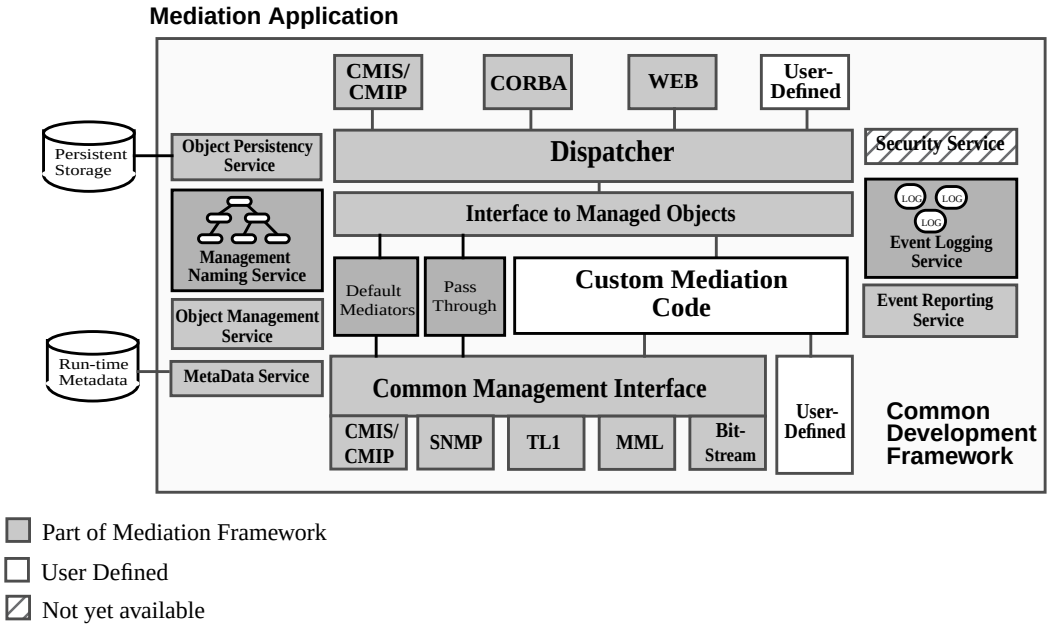

Figure 5 FastBench Mediation Architecture

Common Management Interface

The Common Management Interface is a set of consistent, object-oriented libraries which allow easy access to network resources supporting the following standard protocols or a user-defined protocol:

- CMIS/CMIP,
- SNMPv1 and SNMPv2,
- TL1/MML,
- bitstream-based

The Common Management Interface is based on the NMF TMN/C++ platform. Managed attribute values are handled via the rich NMF ASN.1/C++ API.

Key Programmatic Concepts

Each management protocol interface comprises:

- **Manager Framework** class:
  - Initializes, starts and shuts down the management protocol interface.
  - Sets up global (cross-agent) operation callbacks (invoked for all incoming messages, regardless of the source agent).
  - Sets up additional event callbacks. Multiple pre/post-callbacks can be registered for incoming events which will all be invoked.
  - Sets up an optional catch-all callback.
  - Adds/removes manager identities. One or more manager identities can be set.
- **Agent Handle Factory** class:
  - Creates agent handles.
  - Sets up default responses and event callbacks for newly-created agent handles.
- **Agent Handle** class:
  - Binds/unbinds handles to actual agents.
  - Invokes operations to request GET, SET, CREATE and DELETE operations synchronously and asynchronously.
  - Sets up response callbacks either when invoking the operation method, or at the agent handle level.
  - Adds/removes remote agent identities (name, address, security parameters, etc.)
  - Any number of handles can reference a given agent.

- **Configuration file**: each management protocol interface has its own configuration file allowing the off-line definition of remote agent and local manager identities. Identities added at execution time are saved in a file for recovery at start-up. A graphical configuration tool is provided for each management protocol interface.
- **Easy Memory Management:** the use of a straightforward reference counting scheme simplifies C++ object memory handling between the management protocol interface and custom code. When an object is needed, the reference count is increased. The reference count is only decreased when the object is no longer needed. When the object reference count reaches zero, the object is automatically destroyed.
- **Safe Down-Castable classes**: the management protocol interface classes are instrumented for safe run-time typing.

SNMP Management Interface

The SNMP Management Interface is a collection of high-level TMN/C++ -like classes which allow access to SNMP agents. It provides a means of catching SNMPv1 traps and SNMPv2 notifications. Management operations can be invoked in synchronous and asynchronous modes. Retries of lost requests are handled automatically, providing a solution to SNMP wire unreliability. Segmentation of large protocol data units (PDUs) is carried out transparently.

TL1/MML Management Interface

The TL1/MML Management Interface provides an easy TMN/C++ -like API for accessing network equipment that uses a TL1/MML-compliant protocol. The API can be used for invoking management operations in synchronous and asynchronous modes and for catching incoming alarms. The specifics of the network equipment's TL1/MML variation are defined using a TL1/MML description which is stored in a configuration file. The TL1/MML Management Interface provides a common programming interface to handle all varieties of TL1/MML-based network elements. As a result, custom code does not need to be changed when the TL1/MML protocol varies between network elements. Furthermore, the TL1/MML configuration file can be dynamically updated (at execution time) avoiding system downtime.

A dedicated graphical editor is provided for the creation and modification of the TL1/MML configuration file, avoiding syntax errors and so increasing overall productivity.

A set of ready-to-use configuration files for common equipment is also available on request.

CMIS Management Interface

The CMIS Management Interface enables connection to multiple CMIS-based agents. It provides a high level NMF GDMO/C++ -like API for sending synchronous or asynchronous CMIS operations, receive responses and catch event reports. The C++ classes representing ASN.1 and GDMO are generated from the information model, and provide methods which allow the developer to manipulate attributes, notifications and data types in a consistent way.

Bitstream-based Management Interface

The Bitstream-based Management Interface provides a mechanism to manage, through C++ classes, network equipment which communicates using message blocks (known as binary frames). The Management Interface can be developed using FastBench BitStream Driver and the Common Development Framework. FastBench BitStream Driver allows the developer to specify the general form of each message block using a subset of ASN.1, meaning that messages can be easily factorized, manipulated and debugged as they are passed to and from the network elements.

The ASN.1 description is compiled to C++ classes which allow conversion to and from binary frames. Poorly-defined message data structures can thus be manipulated in a well-structured way.

User-Defined Interface

The Common Development Framework can be customized for building custom network element drivers. The inter-process communication classes and built-in device controllers (TCP/IP, UDP, etc.) make it easy to communicate with remote network elements.

Common Management Services

The Mediation Framework offers a set of built-in, ready-to-use, standards-based Common Management Services. A mediation application can be linked with the required service libraries.

Each service offers a well-defined, easy-to-use C++ interface which can be used by custom code. In addition, each service supports a standard Q3 interface, and an IDL generic interface.

The Mediation Framework currently includes the following common services:

- **Management Naming Service:** Each managed object is identified by a unique distinguished name that refers to the position of the managed object in the global management information tree (MIT).

  The Management Naming Service supports Q3 and IDL interfaces for managed object registration/de-registration and searches. It also offers services for saving the persistent objects and dispatching user-defined operations to a sub-tree of the management information tree.

- **Object Management Service:** The Object Management Service is based on the ITU X.730, X.731 and X.732 recommendations. It automatically submits notifications signaling object creation, object deletion and attribute value changes. The Object Management Service can be enabled/disabled at configuration time or execution time.

- **Event Logging Service:** Based on the ITU X.735 recommendation, the Event Logging Service supports Q3 and IDL interfaces, enabling the creation/deletion and reading/modification of log contents. The Dispatcher automatically passes events to the Event Logging Service. The Event Logging Service routes the event to the existing logs which, in turn, apply any defined filters and log the event if necessary.

- **Event Reporting Service:** Based on the ITU X.734 recommendation, the Event Reporting Service supports Q3 and IDL interfaces, allowing the creation/deletion and reading/modification of event forwarding discriminator

objects (EFDs). The Dispatcher automatically passes received events to the Event Reporting Service which routes the event to the existing EFDs. The EFDs in turn filter and forward the event to the pre-registered manager applications.

- **Attribute Monitoring Service:** Based on the ITU X.739 recommendation, the Attribute Monitoring Service enables automatic polling of managed attributes.
- **MetaData Service:** the MetaData Service offers a C++ interface to the metadata repository. The metadata repository stores the managed object properties in a C++ representation tuned to run-time usage (performant and economic on memory usage).
- **Object Persistency Service:** transparently saves and recovers managed objects to and from persistent storage. A managed object is made persistent by overloading a very simple callback method. User-defined persistent data can also be saved and recovered.

Interface to Managed Objects

The Interface to Managed Objects is the C++ API via which the managed object classes are customized.

The Interface to Managed Objects is GDMO-centric and based on the NMF TMN/C++ API. Developers familiar with the GDMO can use FastBench MIM Builder to construct the GDMO model of managed resources. A model can also be developed using a general-purpose modeling language such as UML and then either used as is or converted to GDMO using the UML-GDMO translation tool.

The user classes supporting the GDMO model can be automatically generated via the Mediation WorkShop graphical tools.

Dispatcher

The Dispatcher constitutes the heart of the mediation application. It is in charge of servicing management requests. The Dispatcher is responsible for the following tasks:

- interfacing with the various Management Protocol Plugs,

- validating operations against the managed object properties defined by the MetaData Service,
- dispatching incoming operations to the target managed objects,
- forwarding replies to the sending application,
- transparently handling management information tree distribution.
- forwarding asynchronous events to the Event Reporting Service and Event Logging Service.

Common Development Framework

The Common Development Framework is the foundation of all components in the run-time mediation architecture. It exposes a collection of high level, customizable OS-independent C++ classes which support the mediation application, providing facilities such as:

- multi-threading,
- inter-process communication (IPC),
- signal handling,
- logging/tracing,
- an administration console (craft terminal or networked),
- basic utility classes (STL, etc.).

The Common Development Framework is shared by all mediation components, increasing component consistency. The Common Development Framework hides OS specifics, ensuring the portability of application source code across different Unix versions, Windows NT or real-time operating systems (RTOS).

Default Mediators

The Mediation Development Environment can be used to rapidly build custom mediation applications to implement a specific information model/protocol mapping or to add user-defined management protocol plugs and interfaces. The Default Mediators provide ready-to-use mapping modules which can be used as is or customized within the Mediation Development Environment.

SNMP-CORBA Mediator

The SNMP-CORBA mediator allows a CORBA-enabled management application to easily control SNMP agents. The SNMP-CORBA mediator provides generic (SNMP MIB-independent), easy-to-use IDL interfaces for catching SNMP events, performing management operations (GET, SET, etc.) and administering the mediator itself.

TL1-CORBA Mediator

Like the SNMP-CORBA mediator, the TL1-CORBA mediator supports a set of generic IDL interfaces allowing CORBA-enabled OSSs to handle TL1-based network elements. Developers can easily catch alarms and perform management operations. TL1 message varieties are handled via a configuration file, meaning that the IDL interfaces are unchanged when the TL1 message are updated.

Q3-CORBA Mediator

The Q3-CORBA Mediator provides a generic (GDMO MIB-independent) IDL interface for the creation, deletion, reading and modification of managed objects. Via the Q3-CORBA IDL interfaces, CORBA management applications can perform regular management operations (GET, SET, CREATE and DELETE) and register to catch notifications.

SNMP-Q3 Mediator

The SNMP-Q3 mediator implements the NMF CMIP to SNMP interworking specifications (NMF 026, NMF 028, NMF 029). It provides a natural mapping of SNMP MIBs to GDMO models. The SNMP-Q3 mediator is based on a data-driven mapping engine, eliminating the need to rebuild the mediation application when new SNMP MIBs are incorporated in the network. The SNMP-to-GDMO off-line translation tool is used to map SNMP MIBs to a GDMO model and to generate the associated run-time metadata which is loaded by the mapping engine to take the new MIBs into account.

Where applicable, the provided IDL interfaces share the same programming concepts with equivalent C++ APIs, increasing the consistency of Mediation Development Environment components and consequently improving developer productivity.

Benefits

Integrated, Graphic-Oriented Approach Speeds Application Development

The Mediation WorkShop's graphical tools hide the complexity of the representation and manipulation of underlying data. All development tasks can be accomplished within a single environment, meaning code can be tested as it is developed.

Ready-to-Use Management Services and Mediators Give Developers a Head Start

The availability of a comprehensive set of fully-tested common management services and default mediators takes the donkey work out of mediation development, allowing developers to concentrate on application specifics instead of coding common protocols and services.

Comprehensive, Consistent Set of Management Protocol Interfaces Simplifies Access to Heterogeneous Resources

A wide range of consistent, high-level C++ APIs make it easy to manage the multiple protocols present in today's heterogeneous networks. APIs share a common structure, reducing the learning curve.

Standard-based, Object-Oriented Framework Reduces Development Costs

The use of a standard (TMN/C++) -based, object oriented framework increases the portability of application code, meaning code can be reused and maintenance and development costs reduced.

Glossary

The following is a list of the abbreviations used in this document and their meaning.

ASN.1	Abstract Syntax Notation 1
CDF	The FastBench Common Development Framework
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CORBA	Common Object Request Broker Architecture
GDMO	Guidelines for the Definition of Managed Objects
HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IPC	Inter-Process Communication
MDE	The FastBench Mediation Development Environment
MIB	Management Information Base
MML	Man-Machine Language

RTOS	Real-Time Operating System
SNMP	Simple Network Management Protocol
TL1	Transaction Language 1
UML	Unified Modeling Language

Bibliography

ITU X.730 | ISO/IEC 10164-1 - Information technology - Open Systems Interconnection - Systems Management: Object management function

ITU X.731 | ISO/IEC 10164-2 - Information technology - Open Systems Interconnection - Systems management: State management function

ITU X.732 | ISO/IEC 10164-3 - Information technology - Open Systems Interconnection - Systems Management: Attributes for representing relationships

ITU X.734 | ISO/IEC 10164-5 - Information technology - Open Systems Interconnection - Systems Management: Event report management function

ITU X.735 | ISO/IEC 10164-6:1995 - Information technology - Open Systems Interconnection - Systems Management: Log control function

Bibliography

ITU X.739 | ISO/IEC 10164-11:1993 - Information technology - Open Systems Interconnection - Systems Management: Metric objects and attributes

NMF 026, Issue 1.0: Translation of Internet MIBs to ISO/ITU GDMO MIBs

NMF 028, Issue 1.0: Translation of ISO/ITU to Internet Management Proxy

NMF 029, Issue 1.0: Translation of Internet MIB-II (RFC 1213) to ISO/ITU GDMO MIB

NMF 040-1, Issue 1.0: ASN.1/C++ Application Programming Interface Part 1 - Base Classes and Specific Interface

NMF 042-1, Issue 1.0: GDMO/C++ Application Programming Interface Part 1 - Architecture

NMF 042-2, Issue 1.0: GDMO/C++ Application Programming Interface Part 2 - Class Reference

NMF 043, Issue 1.0: TMN/C++ Application Programming Interface

For More Information

This paper is one of a range of documents presenting the technology and services in the FastBench range of TMN agent, manager and open mediation solutions. For more information about available documentation and the products and services we provide, please contact us:

Americas/Asia Pacific:
NETMANSYS Inc.
6021 Morriss Road, Suite 101
Flower Mound, TX 75028 - USA
Telephone: +1 972 355 6333
Fax: +1 972 355 6334

Europe/Middle East/Africa:
NETMANSYS S.A.
30, chemin du Vieux Chêne
38240 Meylan - FRANCE
Telephone: +33 4 76 90 98 00
Fax: +33 4 76 41 86 03

http://www.netmansys.com
info@netmansys.com

Copyright © 1998
NETMANSYS Inc.

Part no.: WHT-W-XX-X-FA-10

Revision A, printed in the USA, September 1998