

MASTER'S THESIS

Network management using WBEM (Web-Based Enterprise Management)

Kristofer Sandlund

Civilingenjörsprogrammet Elektroteknik

Institutionen för Systemteknik
Avdelningen för Datorkommunikation

Network management using WBEM (Web- Based Enterprise Management)

Kristofer Sandlund for Ericsson Erisoft

Preface

This report is a Master's thesis report for Luleå Tekniska Universitet (LTU), Department of Computer Science and Electrical Engineering, Division of Computer Communications. The thesis work has been performed at Ericsson Erisoft in Luleå, Sweden from August 2000 until January 2001.

The author would like to thank everyone who has helped realizing this project.

Abstract

In this report, the Web-Based Enterprise Management (WBEM) standard is evaluated for the possible use in the Ericsson NMS (Network Management System). The WBEM standard is still being developed, but has already received the backing of most of the network management industry. This report has evaluated the different parts that make up the WBEM standard and found it to be an extremely versatile and generalized standard that can be applied in almost any area of network management.

The downside of being that general is of course that the performance can not reach the levels attained by a simpler protocol. To get a reference to current network management standard, a comparison of WBEM with SNMP has been performed. Also, to see that these theoretical conclusions hold up, a prototype of a WBEM access module for NMS has been developed. That prototype has been tested and its performance has been compared to an SNMP equivalent. This prototype has also shown how WBEM can be integrated into NMS in a simple and extendable fashion, which is an important lesson if the future requires NMS to manage network elements with a WBEM management interface.

The results of all this shows that WBEM will most likely have a large impact on how network management is performed and most systems will have to consider supporting WBEM. But it also shows that on a network with limited bandwidth, WBEM might not be an ideal choice for the manager.

Table of contents

| | | |
|------|---|----|
| | Preface..... | 2 |
| | Abstract | 2 |
| | Table of contents | 3 |
| 1.0 | Introduction | 4 |
| 2.0 | Delimitations | 4 |
| 3.0 | Acronyms and abbreviations..... | 5 |
| 4.0 | About Network Management..... | 6 |
| 5.0 | What is NMS?..... | 6 |
| 6.0 | WBEM Introduction | 7 |
| 6.1 | Who develops WBEM? | 7 |
| 6.2 | Where is it used? | 7 |
| 6.3 | Introduction to the components of WBEM..... | 7 |
| 6.4 | WBEM history | 9 |
| 7.0 | The standards that make up WBEM | 9 |
| 7.1 | CIM | 9 |
| 7.2 | XML..... | 10 |
| 7.3 | HTTP | 10 |
| 7.4 | MOF | 11 |
| 8.0 | Evaluation of WBEM..... | 11 |
| 8.1 | Complexity..... | 11 |
| 8.2 | Software development for WBEM | 12 |
| 8.3 | Performance | 13 |
| 8.4 | Security | 14 |
| 8.5 | Naming in CIM | 16 |
| 8.6 | Generality..... | 17 |
| 9.0 | What is SNMP?..... | 17 |
| 10.0 | WBEM compared to SNMP | 17 |
| 10.1 | Performance | 17 |
| 10.2 | Complexity..... | 18 |
| 10.3 | Security | 18 |
| 10.4 | Naming..... | 20 |
| 10.5 | Generality..... | 20 |
| 11.0 | WBEM in NMS | 20 |
| 11.1 | General design criteria for WBEM AMs | 20 |
| 11.2 | The prototype built for this thesis | 21 |
| 11.3 | Performance testing on the prototype | 26 |
| 12.0 | Conclusions..... | 28 |
| 12.1 | The future of WBEM | 28 |
| 12.2 | WBEM in NMS | 29 |
| 13.0 | References..... | 30 |
| | Appendix A List of WBEM standard methods..... | 31 |
| | Appendix B The Fictional_NE MOFs | 32 |

1.0 Introduction

This report consists of an investigation into the WBEM standards that are currently being evolved. The intent of WBEM is to unify the area of network management into a single standard. The investigation is based on the needs of the telecommunications management system NMS and how it can make use of future network elements using a WBEM interface. As many of the network elements today are using an SNMP interface, this report will also try to make a comparison between WBEM and SNMP. The SNMP comparison tries to find out in what ways WBEM improves, replaces or integrates today's SNMP units and where SNMP still is a better option than WBEM.

Another aspect of this report is to specify design criteria for an Access Module for WBEM network elements to be used in the NMS system. This design will set up demands on how a generalized framework for WBEM Access Modules should be designed. Such a framework must try to take advantage of as many features as possible from both the TeMIP and from WBEM, advantages that might conflict with each other on many aspects.

During the work on this report a prototype Access Module has been developed according to the design criteria specified in the investigation. This prototype has then been used for some simple performance testing to prove/disprove the conclusions drawn in the theoretical part of the report.

2.0 Delimitations

This report will focus on how WBEM can be used for telecommunications management, which means that certain aspects of WBEM will be thoroughly examined, while other parts will only get a cursory examination. Also, currently existing WBEM applications are using multiple transport protocols, but this report will only discuss the use of HTTP as this is the standard transport in the WBEM specifications. Neither will this report give any detailed description of CIM, HTTP or the CIM-XML encodings. Short introductions to them and discussions on their advantages and disadvantages will be given though.

One part of this report performs a comparison between SNMP and WBEM. This comparison will be against only two versions of SNMP (v2c and v3) as these versions are of relevance for NMS. SNMP can as well as WBEM be run over a number of different transport protocols, but in this report only UDP will be examined as it is the most frequently used protocol for SNMP units in NMS.

The implementation of a prototype access module that has been made for this report is not meant to take advantage of the complete functionality of WBEM, its primary goal is to see how WBEM can be integrated into NMS. Its secondary goal is to offer a testing facility to give more substance to the conclusions drawn in the comparison with SNMP.

3.0 Acronyms and abbreviations

| | |
|-------|--|
| AM | Access Module (in the TeMIP system) |
| API | Application Programming Interface |
| ASN.1 | Abstract Syntax Notation One |
| BER | Basic Encoding Rules |
| CBC | Cipher Block Chaining |
| CIM | Common Information Model |
| CIMOM | CIM Object Manager |
| CMIP | Common Management Information Protocol |
| CORBA | Common Object Request Broker Architecture |
| DES | Data Encryption Standard |
| DMTF | Distributed Management Task Force |
| DTD | Document Type Definition |
| HMMP | HyperMedia Management Protocol |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| MD5 | Message Digest 5 |
| MIB | Management Information Base |
| MOF | Managed Object Format |
| NE | Network Element |
| NMS | Network Management System |
| RC4 | Rivest Cipher #4 |
| RFC | Request For Comments |
| RSA | Rivest-Shamir-Adelman encryption |
| SDK | Software Development Kit |
| SGML | Standard Generalized Markup Language |
| SHA | Secure Hash Algorithm |
| SHTTP | Secure HTTP |
| SNMP | Simple Network Management Protocol |
| SQL | Structured Query Language |
| SSL | Secure Socket Layer |
| TeMIP | Telecommunications Management Information Platform |
| URI | Uniform Resource Identifier |
| VACM | View-based Access Control Model |
| WBEM | Web Based Enterprise Management |
| XML | eXtensible Markup Language |
| XSL | eXtensible Stylesheet Language |

4.0 About Network Management

Network management is the task of monitoring, configuring and maintaining a network environment. Network management can be divided into a number of sub-disciplines such as fault management, performance management, configurations management, event management and asset management. Fault management is the task of monitoring and controlling the network resources while performance management means the collection of performance data for the network. Configurations management consists of tasks such as installation, administration and configuring of the network and all its components. Event management (also referred to as alarm management) is performed when states change in the network and this area encompasses trouble tickets, work orders etc. The final area listed here is asset management, which includes inventory of both physical elements and of network configuration data.

Network management is a very important area for any network administrator who wants a working network environment. This discipline has always been an area of high costs and high demands on the administrators as there have always been a number of conflicting standards with no common models. Managed units have not been compatible with each other causing the network administrator to use multiple different management applications to be able to control the network. Some of the standards currently being used in different aspects of network management are SNMP, CMIP and CORBA. Adding to the confusion is the fact that even different versions of the same standards might not be able to coexist (i.e. SNMPv1 and SNMPv3).

This situation has made for a great need for a standard that can unify the current standards while also be able to model all thinkable forms of network elements in a single information model. An attempt to achieve this goal is the WBEM (Web Based Enterprise Management) standards.

5.0 What is NMS?

NMS is a network management system developed by Ericsson that is mainly used to manage wireless network systems. NMS contains all the areas of network management listed above. Some of the goals for NMS is to reduce the costs of managing the network, while at the same time giving as much functionality as possible. NMS is built upon the TeMIP (Telecommunications Management Information Platform) framework from Compaq. The TeMIP is extendable so that by implementing new management modules, it is possible to build almost any kind of management system with the TeMIP.

NMS is built on components from multiple vendors where Ericsson has integrated these components into a complete network management system. NMS can also be extended to integrate relevant applications requested by the customer. NMS is primarily sold to telecommunications operators, who need to control huge networks with tens of thousands of network elements spread over large geographical areas. The most important areas of the NMS are the event management and fault management as such a large amount of network elements cause a huge number of events. Some NMS installations might receive more than 100.000 alarms per day which puts demands on the communications to be efficient and easy to use.

6.0 WBEM Introduction

WBEM is a group of management standards that are combined with regular Internet standards. WBEM is meant to allow management of heterogeneous systems using equipment from any number of different vendors. The primary goal of the WBEM initiative is to unify enterprise management into a single standard that can integrate (but not necessarily replace) older standards like SNMP and CMIP.

6.1 Who develops WBEM?

The WBEM specification is developed by the Distributed Management Task Force (DMTF), which consists of a large number of companies involved in the network management scene. Some of the board members of DMTF are Sun Microsystems Inc., Microsoft Corporation, Cisco Systems Inc., Intel Corporation, 3Com Corporation and Compaq Computer Corporation. A number of working groups (WG) for developing different parts of the standards are part of DMTF. Example of DMTF working groups are the Events WG and the CIM WG.

Any employee of a DMTF member company can join the DMTF and possibly DMTF working groups and in this way read/influence the work in progress. The method gives the DMTF input from a large part of the people working in the network management industry and hopefully increases the chance of user acceptance of the standards developed.

6.2 Where is it used?

A large number of equipment vendors have started to release products supporting WBEM. For example, many network management systems have started to be able to manage elements with WBEM interfaces. Examples of management systems capable of managing WBEM elements are BMC Patrol and CiscoWorks. Both Microsoft and Sun have proven their commitment to WBEM by allowing Windows NT and Solaris to run as WBEM servers. Compaq's Tru64 UNIX on AlphaServer is another example of an operating system capable of being managed with WBEM. As not all the parts of WBEM are fully specified (namely the event model), these implementations are either not complete in their functionality or they are using some hybrid form of WBEM that might for example be using another transport protocol than the standard transport, which is HTTP.

6.3 Introduction to the components of WBEM.

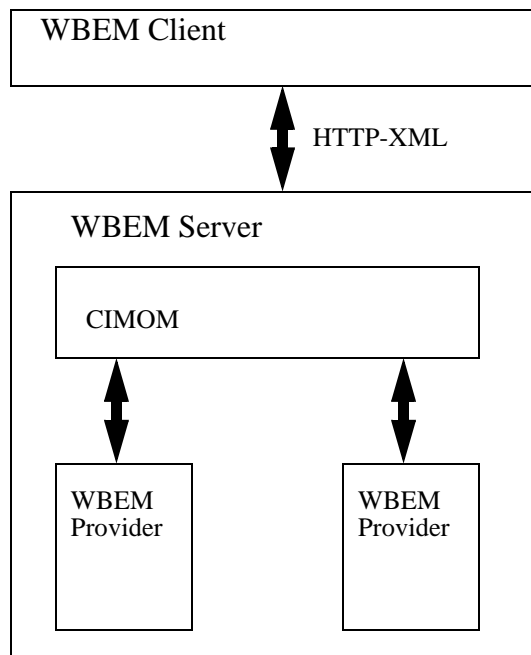
WBEM is built around the Common Information Model (CIM). CIM is a standard for representation of management information. CIM is an object-oriented representation containing standard models for lots of different equipment such as computer systems, networks, devices and applications. The ultimate goal of CIM is that it should be possible to model any form of managed equipment. CIM also offers possibilities for vendors to develop their own object schemas by extending the existing CIM model.

To build a system capable of WBEM there needs to be a WBEM client and (at least) one WBEM server. A WBEM server consists of any number of WBEM providers (called agents in most other standards), a CIMOM (CIM Object Manager) and an interface for communication with a WBEM client. A WBEM client is usually a

management application, but can be some other form of consumer of WBEM data. This structure makes it so that the client does not directly access the provider of the data, but instead gets its information from the CIMOM, which in turn interacts with any given number of WBEM providers.

FIGURE 1.

Structure of WBEM elements.



Another standard for representing managed objects is MOF (Managed Object Format), which is used as a textual representation of CIM. WBEM servers usually contain a MOF compiler that compiles MOF text files into CIM format for the CIMOM to use. The usage of MOF is not an absolute requirement in WBEM, but in most existing solutions it is being used.

WBEM uses HTTP as the transport protocol for communication between hosts as specified in [1]. The payload in the HTTP-packets is CIM information encoded in XML according to [2]. Many of the current products supporting WBEM are using other transport protocols as many of them have been released prior to the DMTF's decision to use HTTP. An example of this is Microsoft's WMI that uses DCOM as the transport protocol when communicating with other WMI elements. WMI will in future releases provide a XML-HTTP interface for communicating with non-WMI WBEM elements, but looks to keep the DCOM interface for intra-WMI communication. For a unit to be WBEM compliant, it will have to support XML-HTTP, but it is up to the developer to decide if the unit should also support other transport protocols as in the WMI example. Another example of a product that is a hybrid of WBEM is Compaq's WBEM for Tru64 UNIX on AlphaServer that uses HTTP for request and reply messages, but SNMP traps to send alarms.

At the time of this writing, at least the following implementations of CIM are available: Microsoft WMI, Sun WBEM SDK, Java CIMOM from SNIA and Paula for Linux.

XML is a language for representation of structured information. XML does not have a standard set of tags like HTML, but instead lets the user define own tags. To check the syntactical correctness of an XML-document, there needs to be a DTD (Document Type Definition) that specifies the exact syntax allowed for this specific document type. XML does not either specify how a document should be displayed on-screen and every user can specify his own XSL style sheet to display each new type of XML document.

For the CIM representation in XML, the DMTF has provided a special DTD for CIM information [3] to allow parsers to check that a WBEM message is syntactically correct.

6.4 WBEM history

The WBEM initiative was started in July 1996 by BMC Software, Cisco Systems, Compaq, Intel and Microsoft. In the early efforts, the group wanted to design a completely new transport protocol for use with WBEM, the HMMP (HyperMedia Management Protocol). That effort was later dropped in favor of using HTTP extension headers so that there was no need for a specific protocol for WBEM. At the same time, it was decided to use the CIM standard from DMTF to represent management information in WBEM. More and more vendors gradually decided to support the evolving WBEM initiative and in June 1998 the group decided to give the development of WBEM over to the DMTF (where all the WBEM-founding companies now are board members).

7.0 The standards that make up WBEM

7.1 CIM

CIM stands for Common Information Model and is an Object-Oriented representation of management information. The goal of CIM is to be able to model all kinds of information with the same information model. CIM is not exclusively used in WBEM, it is an industry standard being developed by DMTF. An example of another use of CIM is DEN (Directory Enabled Networks) which is yet another DMTF standard. CIM has a Core schema that describes the basic classes of CIM. The Core schema is supposed to be stable and the DMTF are not expected to do any modifications or deletions from this schema. There can still be additions to the Core schema but these are not expected to have any devastating effects on existing WBEM elements.

The Core schema is extended by a number of Common schemas that describe different types of managed equipment. Examples of Common schemas are: Applications schema, Systems schema and Device schema. These schemas are provided by the DMTF and developers are not allowed to change them. Vendors wanting to develop specific object schemas for their equipment have the option of extending the Common schemas with so-called Extension schemas. The new classes defined in extension schemas are supposed to inherit from some Core/Common schema classes, but can also be completely independent.

These schemas give CIM a three-level hierarchy where the central level is stable, the middle level can only be changed by the DMTF and the third level is vendor-specific.

7.2 XML

XML is a language for representing structured information. XML is a subset of the more general SGML, that is considered too complex for most usages. XML allows the user to specify his own format for representing every specific type of information.

Each different information representation should provide a DTD (Document Type Definition) that specifies the rules for encoding the information in XML. A DTD is primarily used to check the syntactical correctness of an XML document. The XML standard is spreading quickly across the Internet world where it is being used in more and more areas.

In WBEM, the CIM information is XML-encoded according to a DTD for CIM [3]. Due to the structure of XML, this information encoded in XML is always directly human-readable, while still keeping a structure that is suited for machine-reading. If an application wants to display the information in a specific fashion, then all you have to do is create a XSL style sheet that will tell an XML-application how the information is to be displayed on-screen.

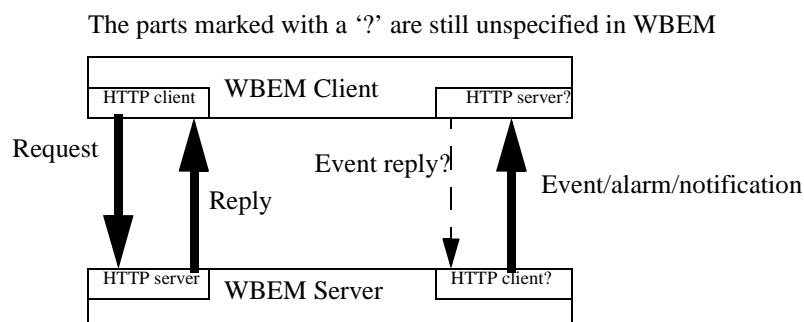
7.3 HTTP

HTTP is the regular protocol used for example by web browsers to get information from a web server. HTTP is run on top of TCP and will therefore be using all the functionality of TCP. The HTTP has two types of messages, request and response. The part issuing request messages is called an HTTP client and the part issuing the response messages is called an HTTP server.

WBEM uses the HTTP extension framework to tell the server and client applications that it is CIM information that is being sent. This is done by the definition of a number of HTTP extension headers for CIM (these are defined by the DMTF). The use of these header extensions will allow proxies to set specific filters for CIM information. The document "Specification for CIM operations over HTTP" [1] contains a specification of how WBEM units communicate over HTTP. As the WBEM client will request information, it will need to have an HTTP client running.

The WBEM server on its side must be able to receive requests and issue replies which creates a need to run an HTTP server. The initial specification (v1.0) of [1] does not contain any support for the sending of events/alarms from the WBEM server to the WBEM client. One of the problems with this is that with events, it will have to be the WBEM server that issues the requests, which will mean that the WBEM server will need an HTTP client in addition to the HTTP server it is already running. Also, the WBEM client will have to be able to receive this request with an HTTP server of its own with the special behaviour that no reply is needed (it is very unusual that events/alarms need a reply in network management).

FIGURE 2. The WBEM event problem



This problem is currently being addressed by DMTF's 'WBEM interoperability work group' and a solution to this will most likely provide a new version of the specification above with some additions.

Another issue that is being worked on in conjunction with the event model is how a server is supposed to know where to send the events generated by its providers. The suggested solution is that the WBEM client issues an 'event subscription', where it specifies an event filter that tells the server what kind of events this client wants to receive. Then the WBEM server will know where to send events and even what type of events to send to each subscribing client. The public release of the WBEM event model is planned for December 2000, but a revision of [1] might still have to wait for some time.

7.4 MOF

MOF stands for Managed Object Format and is a textual format for representing management information. Support for all the features in CIM is provided by MOF so that all CIM information can be mapped to MOF. MOF is not exactly a part of the WBEM standards, but it is used in existing WBEM implementations of CIM and the use of MOF is supported by the DMTF. It can be pointed out that even though MOF is not required in an implementation of CIM, its use can facilitate the operation of a CIMOM. The regular way of defining new CIM classes or instances is to first write them in MOF and then put them through a MOF compiler that outputs this new information as CIM to be used by a CIMOM or a WBEM client application.

8.0 Evaluation of WBEM

8.1 Complexity

The CIM over HTTP specification v1.0 [1] offers only four different operations messages: SIMPLEREQ, SIMPLERSP, MULTIREQ, MULTIRSP, where the last two are used to bundle multiple commands. Still, the demands on the communication mechanisms are quite high, as it needs to support all the header extensions defined in the document plus all the normal functions provided in HTTP v1.1. The operations messages can contain either one (or many) standard (intrinsic) method-call(s) (getting a class, deleting an instance etc.) or it can use an extrinsic method

call, which means that the operation calls a method that is defined specifically for the object being operated on.

There are 23 intrinsic methods defined in the CIM specification (a list of these are included in an appendix to this report) and a client or server should be able to support all of them. These operation messages (intrinsic and extrinsic) allow WBEM to support almost any kind of operation, but it complicates the construction and decoding of messages. The addition of spontaneous event handling is likely to add some new message type(s) to the specification, but those will be necessary if WBEM wants to be able to perform alarms management.

WBEM also provides a query language, WBEMSQL (still under development), that is a subset of the SQL. This query language will be used by clients to specify criteria for a set of objects to be operated upon. This is used so that the network usage can be reduced by sending one query for operating on a number of objects fulfilling a certain criteria instead of an identical call for each of these objects. The introduction of a query language adds even more complexity to a WBEM server that will have to support parsing of SQL queries to be able to fully support WBEM. To reduce this extra complexity, the WBEMSQL has four different levels of increasing support for SQL-methods so that a server can decide to support only Level 1 (the simplest level) of WBEMSQL. In this way the handling of queries will not slow down the server operation as much as WBEM Level 4 support might have done.

Encoding CIM information in XML during transport makes it possible to read CIM 'by hand', which in turn makes it possible to manage WBEM compliant units from a simple text-based window like telnet. This is an important feature that simplifies WBEM as the network operator can read messages without having to use a specialized application to decode them first. The downside with this usage of XML is that it will demand more processing power from the server and client applications than if a binary representation had been used. This is because a text-based message uses more computing power for parsing than a simple binary message would use.

8.2 Software development for WBEM

The promise of SDKs and other application frameworks being developed by numerous manufacturers will facilitate development of both client and server applications for WBEM by providing APIs that are ready to use. For example, the SUN WBEM SDK is a Java-based API that can be used to implement WBEM solutions on all Java-supporting platforms, which today means practically every computer. Another similar SDK that facilitates development of WBEM servers is the SNIA CIMOM, a free product with open source code that is compatible with the Sun WBEM SDK API.

Developing WBEM applications without the help of existing SDKs, though, could be quite a complex venture as implementation of all the WBEM standards will have to be performed.

The use of standard internet technologies eases the development of WBEM applications as one can use pre-existing implementations of for example HTTP to build the communication parts of WBEM. A problem when developing WBEM applications will always be to find (or develop) mappings from other standards that might already be used in the target network.

8.3 Performance

There are a number of issues regarding the performance of WBEM. The area of efficient performance is where WBEM might be weak as a unifying standard relies more on generality than efficiency.

8.3.1 Encapsulation overhead

To send a message to a WBEM server, the client will encapsulate the message within XML code. As the XML encoding of CIM is built more to be general and easy to understand than it meant to be efficient, this is quite a large overhead. Such a message will also have the overhead of IP header, TCP header, HTTP header and possibly even more overhead used for encryption and/or authentication.

8.3.2 Connection reliability

As WBEM will send its messages with HTTP, it makes use of the underlying functionality of TCP. This will mean that all transmissions will be reliable in the sense that the sender will know if the transmission was received correctly or not. The sender will attempt multiple re-tries to get the message across if no confirmation is received.

8.3.3 Mapping translations

As WBEM is meant to unify all existing management standards, mapping operations for messages from these standards to the CIM format will be necessary. This will add some extra processing for existing network elements that want to be able to be managed with WBEM. These mappings will have to be performed in the WBEM provider application before passing the information to the CIMOM. Similar mapping might also be needed in existing management applications that are to be extended to support WBEM. This is because the internal structures of the application might not be compatible with CIM representations.

8.3.4 Event filtering

The event model that is currently being worked at is set to work in the following fashion: Any WBEM Client application wanting to receive events will set a subscription at the server. This subscription will have to state a filter that in turn sets criteria for the events it wants to receive from the server. This mechanism will make it so that filtering of events will take place at the server itself and therefore reduce the traffic that has to go through the network. The downside of this scheme is that the operator of the client system will have to be very familiar with how to set these subscriptions so that he/she gets only the alarms that are wanted and still does not miss any important events. To facilitate for the operator there is the option to wild-card the alarms filter, effectively telling the server to pass on all its alarms to the client application.

8.3.5 Associations

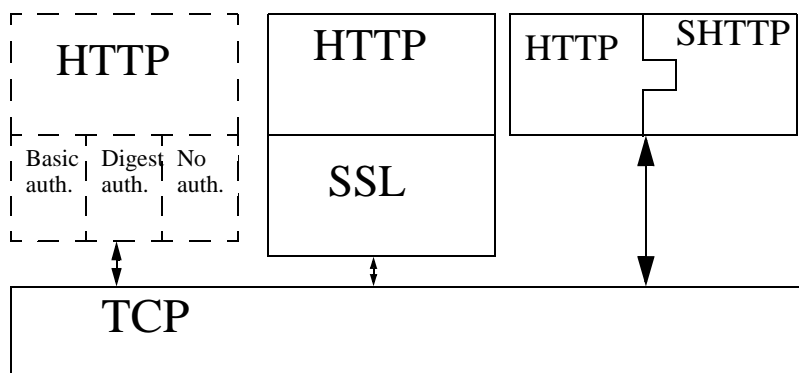
CIM classes and instances can often contain a large amount of associations to all objects that in some way are related to them. The associations will be very useful for the system manager who can easily see the correlation between the different parts of the managed system. These associations will make the structures in the CIMOM very complex and keeping track of the changes in associations could pro-

vide the CIMOM with high performance costs. The severity of this perceived problem will very much depend on the implementation of the CIMOM.

8.4 Security

WBEM in itself does not offer any specific security features, but the use of HTTP provides a number of security options that are widely accepted and thoroughly tested in regular HTTP implementations. These security options range from no security whatsoever, up to the use of Secure HTTP. How strong level of security is needed can be decided by the user of the system. A description of the different modes of authentication follows.

FIGURE 3. The security mechanisms for WBEM.



8.4.1 No authentication

No authentication means that anyone can perform any operations without supplying identity information. HTTP used without any form of security is only possible in a highly secure environment and is not recommended for telecommunications network management.

8.4.2 Basic authentication

The basic authentication mechanism provided with HTTP is a challenge-response based authentication. This means that the server challenges the client to provide a username and password before being allowed to receive information from the server. The user id and password are base64-encoded before being transmitted, which means that it is not directly human-readable, but can be simply decoded with pen and paper in a couple of minutes. Due to this, the basic authentication scheme demands a trusted carrier if it is to be used. No form of content encryption is provided in this scheme. Basic authentication will protect from accidental access to information, but will not stand up to even the simplest form of attack.

8.4.3 Digest authentication

The digest authentication scheme is described in RFC2069 [6]. This is a stronger version of the basic authentication, and in the same way, this scheme is also

restricted to authentication and does not support encrypted message content. The specific authentication method to use is provided by the server who will give a 'nonce' to the client. This nonce is recommended to be a combination of the client id, a timestamp and a private key, but can be anything the server decides. The server will then, with the help of the timestamp, be able to decide whether the nonce is to be one-time or be time-limited. The client should then respond with username, password, nonce and an URI, all protected by a MD5 checksum (the specific checksum algorithm to be used can be specified in a header option, but MD5 is the default). This way the server can authenticate the client by decrypting the MD5 checksum and compare to the data stored in the server.

The password is not sent in the clear as it is in the basic scheme, it will therefore be much harder for an eavesdropper to get hold of the password. This scheme is vulnerable to replay attacks unless the nonce uses the timestamp of the nonce to make all requests into one-time. The digest authentication is also weak to 'Man-in-the-middle' attacks as it is (for example) easy for a compromised proxy to fool a client into using basic authentication with the purpose of stealing the password.

8.4.4 SSL

SSL stands for Secure Socket Layer and is a protocol used between TCP and HTTP (when SSL is used, the URL-prefix https is used). Apart from authentication, SSL also provides support for encryption of the message contents. The main components of SSL are the SSL Record protocol and the SSL Handshake protocol. The Handshake protocol is used to authenticate the users in a session, while the Record protocol is the lower level protocol that is used to protect the network layer.

The different parts of the security offered by SSL can be negotiated for each session and any algorithm can be used as long as both parties in the conversation have implemented this algorithm. SSL provides public key encryption (for example RSA or DSS) for the authentication, while the data encryption is performed with symmetric encryption such as DES or RC4. The integrity of messages is checked by using hash functions such as SHA or MD5.

An SSL session is initiated by performing the handshake and after that the hosts will exchange 'change cipher spec'-messages where the hosts can negotiate which encryption methods are to be used. After this, the shared secrets can be exchanged and at this stage the session is ready to start sending the encrypted message data. A feature of SSL is that it can optionally provide data compression before the encryption so that the bandwidth usage can be reduced. One weakness in SSL is that it does not provide any non-repudiation facility. Non-repudiation means that the receiver can prove that the sender actually sent a message even though he/she might deny sending the message afterwards.

A downside with using SSL is that it will give a quite long startup delay as it completes the handshake and negotiation of encryption methods.

8.4.5 SHTTP

SHTTP is defined in [7] where the interested reader can find the intricate workings of this protocol. SHTTP means Secure HTTP and is built so that SHTTP will encapsulate HTTP information. To accomplish this, an HTTP message (with its header) is encapsulated inside an SHTTP header that in principle is a normal HTTP header with fields stating that it is an SHTTP transmission. Everything that can provide any

kind of sensitive information about the message is stored in the encapsulated HTTP header. This is so because nothing important should be shown in the SHTTP header, which is the only part of the message that is sent unencrypted. SHTTP provides great flexibility in its security features, including a choice of encryption algorithms to use for each session. Examples of algorithms supported for different parts of SHTTP are RSA, DSS, MD5, SHA, DES-CBC, RC2-CBC and DES-EDE-CBC. The DES-EDE-CBC is triple-DES with 2 keys that encrypts, decrypts and then encrypts again in the DES.

To further strengthen encryption, it is possible to choose different encryption algorithms for the message header and the contents. The security for the session is decided by three different parameters: signature, authentication and encryption. All these parameters can be set to a number of different algorithms or even to none if a less secure transmission is required.

Another feature provided to ensure the freshness of a message is a simple challenge-response mechanism using a 'nonce', similar to that of basic authentication. All information about the security modes used for this transmission is provided in HTTP header fields and is not required to be part of the content itself. SHTTP provides non-repudiation when digital signatures are used. This feature might not be very important in the situation of telecommunications management, but if it is needed, then SHTTP is a better choice than SSL.

SHTTP is generally considered to be a stronger form of protection than SSL due to the slightly more general structure of SHTTP. A wider range of choices will make it harder for the attacker to know what method to use for attacking the system. A downside with SHTTP compared to SSL is that it requires more overhead both in processing and in size.

8.4.6 Basic/Digest vs. SSL/SHTTP

The basic and digest authentications are usually included in any implementation of HTTP. This means that they are easy to add to a WBEM server or client application. If the user requires higher security than those two can provide, SSL or SHTTP needs to be added to the application. These are not standard in an implementation of HTTP and either needs to be implemented or an implementation of them that can fit in with the HTTP used needs to be found. As SSL is an added layer between TCP and HTTP it can be added without being integrated in the application itself, which is an advantage compared to SHTTP. If the applications do not support SSL or SHTTP, then it will not be possible to encrypt the message contents which could be dangerous in certain environments.

8.5 Naming in CIM

The different existing standards for representing management information uses different ways of naming their objects. The scheme for naming used in CIM is that each class in CIM that can be instantiated, contains one (or more) key properties. The value of this property is used to identify the various instances of this class. When a class inherits from another class, all key values are inherited and must be used for identifying the unit.

In this scheme, the key values are not dependant on its surrounding objects, so that for example a hardware card is not named after the slot it is located in, it only uses its own key value for identification. This means that the unit will keep its name even

if it is moved to another slot in another computer. To find the card situated in a specified slot in the computer above, this can still be done by following an association from the specified slot that will lead to the card itself.

8.6 Generality

The decision to discard the attempts at declaring the HMMP in favor of using HTTP extensions will assure a larger spread of WBEM applications as the industry is already familiar with HTTP and does not have to learn a new protocol. In the same way, the use of XML is spreading quickly and many applications supporting XML already exist. This will ease the development of new XML tools for WBEM. The largest advantage of WBEM compared to previously existing management standards is the information model provided by CIM. As all WBEM units will use a single information model to describe every part of the managed system, the interoperability of all WBEM units is practically guaranteed.

Another aspect that seems to add momentum to the WBEM initiative is that mapping from most other existing standards are being developed. This will mean that these standards can be unified into WBEM by just adding a WBEM provider to the existing NEs instead of having to replace these NEs with completely new ones.

9.0 What is SNMP?

SNMP stands for Simple Network Management Protocol and is a standard widely spread in all areas of network management. As its name suggest, it is meant to be simple, efficient and easy to use/understand but does not place very much emphasis on being complete or offer lots of different operations. The versions that this report will look in to are SNMPv2c and SNMPv3. SNMPv2c is currently the most used version but SNMPv3 has more features and might slowly be inching out the older version. SNMP can be run over different transport protocols, but this report will focus on SNMP using UDP. Specifications for SNMP can be found in [4] and [5].

10.0 WBEM compared to SNMP

In this comparison, WBEM will be compared against SNMPv2c and SNMPv3 as these are the versions of SNMP that are considered most important to the network managers of today. SNMPv1 is still used, but no new network elements are created using this version and will therefore not be considered in this evaluation. The same goes for SNMPv2 and SNMPv2* which are not used very much at all.

10.1 Performance

In SNMP, a client is usually required to have prior knowledge of the object model used by the managed NE. To do this, the NE must have access to a MIB-file describing this model. The client can find out its structure by repeated use of the get-next operation. Get-next calls will get the value of a requested attribute and return the address of the next one. A WBEM client on the other hand does not need to have any knowledge of the object model in the NE, because it can be acquired by performing enumeration calls to the server. This group of operations will return the classes (or instances) that match a specific criteria.

Therefore, a WBEM client can issue an enumeration call for all the instances in the CIMOM by specifying the right argument and all matching instances will be returned. This has the advantage of the client only having to perform one call across the network, while the SNMP equivalent would have to issue numerous calls to get all the requested information. The SNMP versions investigated in this report support a get-bulk call that can get multiple attributes in a single call, but this call needs a parameter to say how many attributes are to be fetched and will not fetch the entire instance unless this parameter is set high enough.

All SNMP messages are encoded in BER, which is a standard encoding representation that is built around a Tag, Length, Value scheme (TLV). This means that each value will be encoded with a tag telling what kind of data is represented. After that, a length field telling the size of the data and finally the data itself. This is a very compact encoding scheme, and the tag and length fields will usually be one byte each in length yielding only two bytes of overhead to each value.

10.2 Complexity

As the S in SNMP stands for 'Simple', this is an area where it is supposed to have one of its largest strengths. Therefore, this area is one where WBEM might have trouble competing against SNMP. This section will compare such topics as protocol operations, complexity of applications and encodings.

10.2.1 Protocol operations and message types

There are seven different protocol operations in SNMPv2c: Get, GetNext, Set, Trap, GetBulk, Inform and Reply, while SNMPv3 adds a new Trap type and a Report message taking the total number of protocol operations to nine.

WBEM has only four different protocol operations: SIMPLEREQ, SIMPLERSP, MULTIREQ and MULTIRSP (with possibly two new elements for event transmissions to be added). Considering this, WBEM is less complex in terms of the number of different protocol operations that have to be implemented. But on the other hand, HTTP is far more complex than UDP, so the communication will still be more complex in WBEM than it is in SNMP. When you move to the message types, SNMP only has the same messages as the protocol operations above, while WBEM supports 23 intrinsic methods plus an arbitrary number of extrinsic methods. This will mean that both the client and server applications for WBEM will have to be able to understand a much larger number of message types than SNMP. In turn, this will make the development of applications more complicated and the applications will usually require more processing for WBEM than for its SNMP equivalents.

10.2.2 Message encodings

The BER encoding for SNMP messages is quite simple to perform as you only need to know the type, size and value of an object to encode it. Encoding CIM information in XML, though, is a much larger project that requires good knowledge of both CIM, XML and [2]. Also, the encoding of WBEM messages will give yield more processing overhead than SNMP.

10.3 Security

SNMP v2c provides the same security as SNMPv1, which is based on the community concept. The community concept binds each unit to a community name that is represented by an arbitrary string. Access to the unit is restricted to those that pro-

vide this string in their messages to the unit. Therefore the community string works as a password for SNMPv2c units. This scheme gives only about the same level of security as the Basic authentication used in HTTP, that is, it protects on a secure network environment, but does not have much to offer against an attacker with any form of access to the network.

A large part of the security in SNMPv2c is instead placed in the units in the way that attributes needing protection are given write-protection in the SNMP agent itself. This will mean that these attributes only can be changed locally from the unit and not by a remote manager. Some really sensitive attributes might even be given read-protection, forcing the manager to locally access the NE to even view the status of this attribute. This kind of security does not adapt well to a dynamically changing network environment and will force some parts of the management to take place locally at the units. Compared to WBEM, SNMPv2c is very weak in the security area, where WBEM can offer a wide variety of security options ranging from insecure to very strong security.

SNMPv3 offers great improvements in the security area of SNMP by providing a User-based Security Model (USM) [5]. It can support both authentication and message encryption. SNMPv3 offers the option to select a security strength that is suitable for a specific operation. SNMPv3 can be used with no security, with authentication and no encryption or with both. For the authentication, an SNMPv3 implementation supports the HMAC-MD5-96 and the HMAC-SHA-96 algorithms, which both are a so-called secure hash algorithms.

For the encryption, SNMPv3 currently only offers support for CBC-DES which is a symmetric encryption form, but it shall be possible to define support for other encryption standards in the future. Comparing the SNMPv3 USM to SSL and SHTTP, the latter two offers more flexibility in the choice of algorithms to use than the former. This flexibility makes it much harder for an attacker to break the encryption as the method for attacking must be changed for each possibility. The algorithms that are supported in the SNMPv3 USM, though, are also available in the other schemes, so when the algorithms used are known to the attacker, the protection offered by SNMPv3 should be of similar strength as in SSL/SHTTP.

SNMPv3 does not offer support for digital signatures, which is offered in both SSL and SHTTP. This further reduces the options to choose and again simplifies for the attacker. Another thing is that the SNMPv3 USM has no support for data compression, which is supported in SSL to reduce the bandwidth usage. On the other hand, by using the BER encoding, the data in SNMP is already quite compact and is unlikely to gain much from using data compression.

The security model in SNMPv3 also provides an access control mechanism called the VACM (View-based Access Control Model), that allows the setting of different access rights for users. For example the VACM makes it possible to give one manager write access to a property, while a group of others only get read access to that same property. WBEM does not provide any standard model for different levels of access. This is left to the person implementing the CIMOM to decide how differentiated access for different users should be modelled.

Another difference in the protection strength between the HTTP security schemes and the SNMPv3 schemes is that SNMP runs over UDP, while HTTP lies on top of TCP. This means that the SNMPv3 cannot guarantee reliable delivery of the messages sent, while TCP is built for exactly this reason. In a scheme running over

UDP, it will be harder to detect certain types of attacks as it is harder to detect lost packages.

10.4 Naming

SNMP uses a naming system based on OIDs (Object Identifiers), that gives a name to an object depending on its class hierarchy. This system makes the OIDs into large strings of the form “1.3.6.1.1057.2.1.35.5” etc. To access an instance of this object type, both the OID and an instance identifier specific for the instance must be specified. This system is similar to the one used in CIM, where the OID can be said to correspond to the CIM class name, while the instance identifier corresponds to the key values.

The OID naming system can quickly become illegible as each of the integers in the string stand for some subtree in the naming hierarchy. The OID strings are very hard to remember for a human, while a simple classname is much easier to keep in mind. Also, 99% of all SNMP OIDs start with the same four numbers, “1.3.6.1”, which represent the Internet group of variables, and it seems like a waste not to put the Internet group at the root of the OID tree. An advantage with the OID system is that it guarantees a global uniqueness to each object model, but similar uniqueness with CIM classes is planned as developers strongly encouraged to register extension schemas with the DMTF.

10.5 Generality

SNMP has information models to describe practically everything as does CIM, but the difference is that the SNMP information model is spread out into thousands of different MIBs while CIM is collected into a single model. This is where the biggest advantage of using WBEM lies as CIM is a much more dynamic model than the SNMP MIBs. The MIBs also do not assure compatibility between similar equipment from different manufacturers which means it is hard work to adapt SNMP management applications to different network environments. WBEM also has the advantage of supporting mappings from other standards and it is meant that SNMP units could be used by just adding a WBEM provider on top of the SNMP agent. As SNMP does not have an information model such as CIM, it would not be able to map all the operations from WBEM if it was interested in handling WBEM elements.

11.0 WBEM in NMS

11.1 General design criteria for WBEM AMs

For the implementation of an AM for TeMIP, the Visual TeMIP library must be used. Contrary to its name, Visual TeMIP is not a visual development environment, it is a c++ class library for development of TeMIP management modules.

When designing an AM for WBEM, there are lots of different design criteria to be considered. Some of the main parts will be how to develop the communication with the NE (in this case HTTP communication). A problem with integrating WBEM into TeMIP will be that WBEM wants the object models to be dynamic, while the TeMIP prefers to have as much as possible of this added during compile-time. Therefore the WBEM functions concerning adding, deleting and editing classes will

not be possible to implement in the TeMIP AM. This will also force the developer to code new object models for the TeMIP for each new type of NE that it wants to manage.

This goes completely against the concept of WBEM as its main goal is the vendor-independent management with specialized interpretations of data, but such are the constraints placed upon us by the TeMIP.

Some other important design criteria is to decide how many of the standard WBEM methods will be supported in the AM. The way NMS is used today, very little of the functionality offered by WBEM is necessary, but in the future, NEs might well make use of a larger part of these functions. As stated earlier, the functions for dynamical model handling has to be discarded when TeMIP is unable to handle these. Another problem with making a WBEM AM is that TeMIP uses a management language called MSL to describe managed information. This means that the MOF files for each CIM class must be mapped to MSL before the AM can be enrolled into the TeMIP framework. To develop a WBEM AM framework for TeMIP will mean that a standardized mapping of all CIM elements to MSL will have to be found. Among the problems with developing such a mapping is the fact that MSL is not as rich a language as CIM and decisions on what to discard will have to be taken.

Design options:

1) Make an implementation of CIM from scratch. To do this, an HTTP implementation that supports header extensions needs to be found. The CIM operations over HTTP [1] needs to be implemented by hand as well as the encoding of CIM in XML [2]. Decisions on how to represent CIM inside the AM must also be taken. This solution would produce an AM that is coded exclusively in C/C++ and it will most likely open possibilities for higher efficiency than other solutions, but will be much work. For example, the SNIA CIMOM (see design option 2) is built on something like 120 Java classes, which means that a C++ version of this will probably need even more classes than that. A problem with finding implementation parts to use is that they have to work on a Tru64 UNIX, which is not exactly the most widespread operating system around.

2) Use SNIA CIMOM (or Sun WBEM SDK) to build the WBEM client application. The SNIA CIMOM is a Java API that implements CIM and makes for easy development of both client and provider applications. This CIMOM provides many parts of WBEM ready to use, like the CIM over HTTP operations and XML transfer encodings. In an implementation using SNIA CIMOM, the AM starts the Java process (as a UNIX process) and uses some primitive communication form between these processes. This design option will be fast to implement but probably not very fast during run-time. As the AM will be built on two separate processes (Visual TeMIP and Java), there is a need for communication between these and most likely some mappings when information is passed between them. This option is very useful for testing of the WBEM interface before deciding on how to build a more formal framework for WBEM AMs.

11.2 The prototype built for this thesis

The prototype developed for this thesis is made to test the conclusions drawn in this report and to find out how hard/easy it is to develop future WBEM AMs. Therefore, the AM is not built to be efficient or complete in its functionality. The prototype is

built around design principle 2 above as there is not time to implement CIM as a whole.

For the communication between the Java part and the Visual TeMIP part of the AM, the choice has been made to use regular UNIX pipes. When the AM is started, the Java program's input stream and output stream are redirected to one pipe/fifo each. This way, both parts can constantly read from the pipe/fifo directed towards it and have other threads/processes write to the pipe when this is necessary. For reasons explained later, there is also need for another UNIX primitive, a message queue, that is used to pass information between the different threads in the TeMIP part of the AM.

Due to lack of time and the fact that this is just a prototype to test conclusions from the investigation performed, the AM will only support a very limited subset of the WBEM operations. The TeMIP has standard directives CREATE, SHOW, SET and DELETE and all these except SET have been implemented. The SHOW will use the getInstance-call so that all attributes of the NE can be fetched in one single call. Another feature that has been added to the WBEM AM is the possibility to scan for network elements at a specified location (The SCAN directive). This is a feature that is often requested by NMS customers, but is not really suitable to implement in other network management standards as the NEs seldom support this kind of functionality. This is though very suitable for WBEM as a CIMOM supports the enumerateInstanceNames call that will return the names (key values) of all the CIMOM's NEs of a specified class. This way the AM can scan for CIMOMs and when found ask them to return their NEs in a single call.

11.2.1 The Fictional network element used by the prototype

As it would be complicated to handle existing NEs supporting WBEM, it has been decided to use a completely faked class to represent the NE. This class is called Fictional_NE and the MOF file describing that class is supplied in an appendix to this report. The class inherits the CIM_ManagedSystemElement class from the CIM Core schema so that it has a connection to real CIM classes. The class is very simple and has only a few different data types: strings, integers and booleans (it also contains a DateTime attribute, but in the prototype, this is handled as any other string). This is to show that different data types can be handled in this prototype design, but it is not complete in handling all the different data types defined in CIM. The addition of a few lines of code will be required when adding support for more data types, but no re-writes are expected to be needed.

11.2.2 Description of the communication in the prototype

The following description is quite technical and is directed to the reader that wants a more in-depth description of how the prototype works. The names listed below are used in the description:

- **Pipe_In:** This is a regular UNIX pipe that is used for communication between two threads. The one used here is single-directional and the "In" in the name signifies direction is toward TeMIP.
- **Fifo_Out:** This is a UNIX Fifo that is used for passing messages between processes that do not share a common ancestry.
- **MQ:** Message queues are yet another UNIX primitive that are used for communication between an arbitrary number of threads (or processes). The MQ used here is positioned between the call threads and the reader thread on the Visual TeMIP part of the code. It is used to return call results to the correct caller.

- **VTL:** Main thread running during the entire life of the AM. Its principal function is to read messages from Pipe_In and writing these on to the MQ.
- **VT_A:** A thread created by a call request from the TeMIP. Writes the call messages to Fifo_Out and reads the replies from MQ. As the AM is meant to be able to handle multiple simultaneous calls, there can be any number of this thread type.
- **JL:** The main thread of the Java program. Reads from Fifo_Out and creates a new thread to handle the message read.
- **ST_A:** This is a service thread in the Java part that performs all the communication with the NE. The reply received from the NE is written to Pipe_In. There can be any number of service threads at any one time.

Detailed description of the communication:

The numbers in the following text refer to figure 4 which gives a graphical overview of the AM.

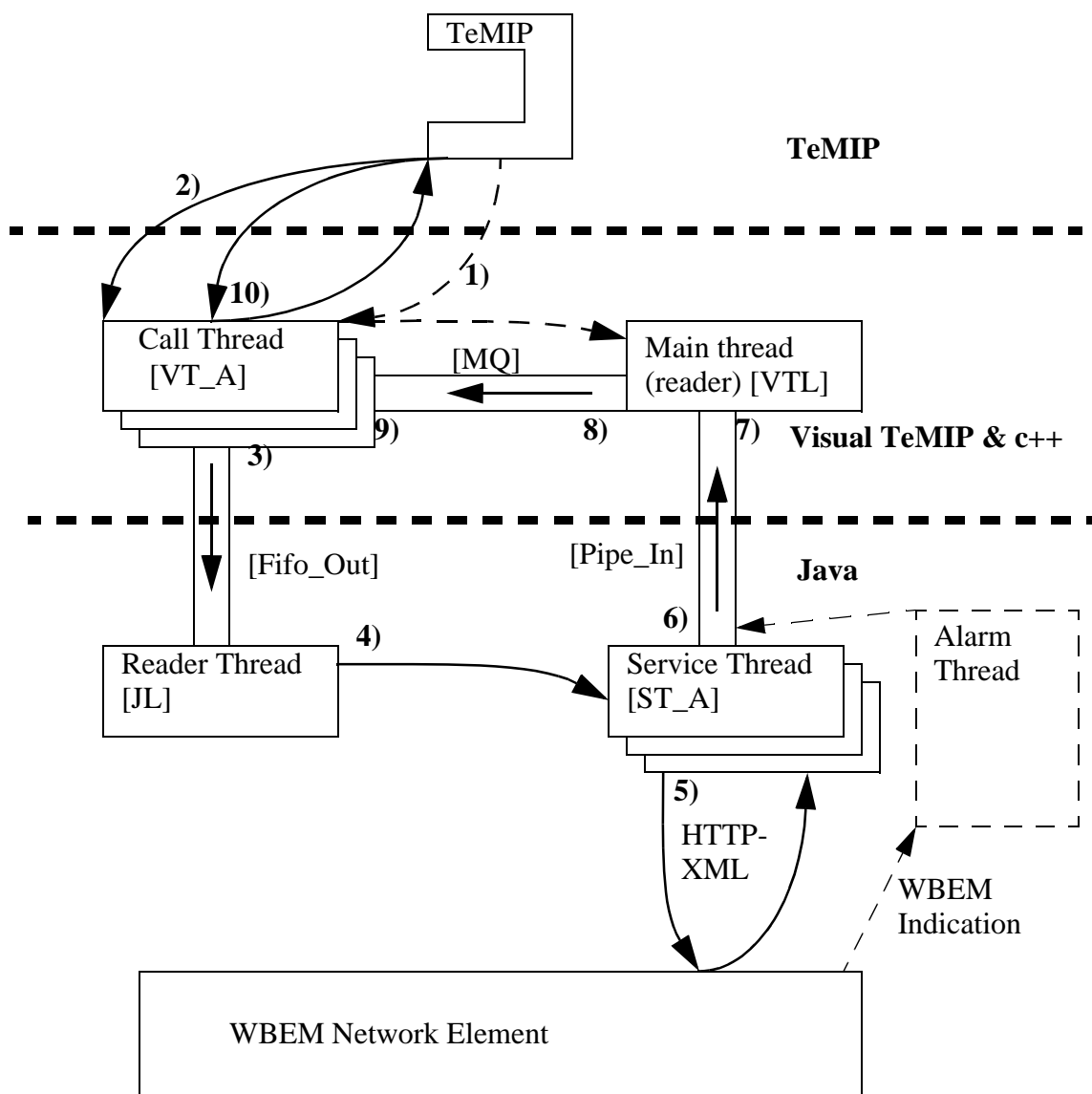
1. The TeMIP starts the AM by performing a Create-call. This call will start up the main thread (VTL) that will read from Pipe_In for the AMs entire lifetime. It will also start up a child thread that will execute the Java process (by starting up JL). After JL is started it will receive a message (on Fifo_Out) that tells JL of the connection information of the CIMOM (the NE). Once this message is received, JL will establish a connection with the NE or report failure to do so.
2. TeMIP performs any form of call request on the AM. This call will automatically be performed in a new thread (here called VT_A).
3. VT_A constructs a message out of the call information received from the TeMIP and writes it to Fifo_Out. After sending the message, VT_A will wait for a reply from MQ.
4. JL receives the messages from Fifo_Out and creates a service thread (ST_A) that takes over the processing of the message. JL waits for the next message.
5. ST_A sends a HTTP-XML message to the NE and receives a reply.
6. ST_A writes the reply to Pipe_In. After this writing, ST_A will stop its execution and be removed from the system.
7. CL reads from Pipe_In and can by looking at an ID# determine which thread is waiting for this reply (In this case VT_A).
8. CL writes the message to MQ. After this, CL waits for the next message from Pipe_In.
9. VT_A reads the message from MQ (Due to the ID# set on the message, no other threads will accidentally access this message). The message is translated to a format suitable for the TeMIP and the result of the operation set as return parameters to the TeMIP. VT_A will then finish its execution and be removed by the TeMIP.
10. The TeMIP will now receive the reply from the call.

Events in the prototype AM:

The prototype implementation will not contain any form of event handling as the WBEM event specification is not finished at the time of implementation. WBEM refers to events as indications, this document will use both those notations but still be referring to the same thing. Figure 4 contains a sketch on how such indication handling could be implemented in the prototype. There would be one extra thread in

the Java part of the code for listening to events sent from the NE. This thread would most likely have to be running an HTTP server to receive these indications. After receiving an event, the Alarm thread would write this to Pipe_In just as any other message from the NE. The difference here is that the indication would not have anyone waiting for it and the CL thread would have to handle these itself without using MQ. These additions will not add much neither in complexity nor processing time to the prototype and should also be easy to integrate if one wants to test indications.

FIGURE 4. The structure of the prototype AM



Communication ‘protocol’ in Pipes/FIFOs and Message Queues

The protocol used in these is very simple and not built for robustness or efficiency. For the Pipe/FIFO, the protocols are identical and a message is structured as in the figure below. The msgID field is used so that the message is received by the right

process, the msgSize is currently unused but is left as a possibility for future versions. The callType is a constant that identifies the type of message that is being sent (for example CONNECT, CLOSE, GET_INST, SET_PROP) so that the receiver knows how to process the message. The nrParams field says how many parameters are included in this message and after that there follows a number of parameters equal to nrParams, each preceded by an integer (paramSize) telling the size of the next parameter.

The protocol used in the message queue is similar, with the main difference that the message is split into a header message and multiple parameter messages as the message queues have a very limited maximum size of a message. The header message consists of msgID, callType and nrParams, while a parameter message consists of msgID, paramSize and paramValue. For decoding of different datatypes by receivers, a property is sent as 3 separate parameters: one for the property name, one for the property datatype and one for the property value (represented as an ASCII string).

FIGURE 5. The Pipe/FIFO protocol message

| | | | | |
|-----------|----------------|---|---------------|---------------|
| | msgID(int) | msgSize(int) | callType(int) | nrParams(int) |
| nrParams* | paramSize(int) | paramValue (ASCII string with length = paramSize) | | |
| | | | | |

Examples of how the message format is specified for the most usual calls are:

CONNECT (adds a new connection to a NE) contains 6 parameters: hostname, port number, namespace, url text, username and password as required by the SNIA CIMOM during connection. For the prototype, only the first two are used while the others are set to default values.

DISCONNECT (removes a connection to an NE) consists of two parameters: hostname and port number.

CLOSE (closes down the entire java process) does not take any parameters at all.

The GET_INST call (retrieves a specified instance containing all its attributes and its classname) is used for the show-command in the prototype uses 3+3*x parameters where x is the number of Key values used to identify the NE. The first 3 parameters are hostname, port number and class name, while each key value is encoded by three parameters: key name, key datatype and key value. This system is used to ensure that the Java program can be made general to all different NEs.

11.2.3 Strengths and weaknesses of the Prototype

Due to time constraints, the prototype does not contain any form of process synchronization and cannot guarantee that data read or written might not be corrupted by simultaneous calls to the AM. To ensure correctness of the AM, locks and semaphores should be used to protect data from being destroyed. The Java process,

though, does synchronize its writes and reads to/from pipe/fifo, but the c++ parts of the code does no such thing.

Another thing that the AM does not handle well is failed requests. If a connection is attempted to a host that is not running a CIMOM at the specified location, the AM will retry the request for quite a long time before returning an error. This delay exists because of the way SNIA CIMOM handles its HTTP connections. To ease the operation of the AM, timeouts should be set after something like 20 seconds, so that the wait time can be reduced. Timeouts could/should also be used when a call to the AM encounters an error of some kind and never receives a reply to this request. In the current implementation, when this occurs, the request might hang forever without returning. Such a feature would be quite easy to add if time was allocated to solve this task.

A big strength of the prototype is that large parts of the code is independent of the object model used for the network element. All the Java code and also the c++ code handling the communication between the AM and the Java process are fully reusable for an NE using another object model. This means that in principle, this design could be used to form the basis of a framework for developing WBEM access modules. The only parts that need to be changed are those handling the specific behaviour of the TeMIP for handling attributes and calls. But such is the way TeMIP works, and existing development frameworks for the TeMIP usually require a bit of coding for each new object model.

11.3 Performance testing on the prototype

Testing of the prototype has been performed to try to see both the efficiency of the implementation and also to test the bandwidth usage of WBEM. Due to the limited functionality of the prototype, these tests cannot be very thorough, the intention is to see a tendency of how well WBEM will perform. To have something to compare the results to, a SNMP simulator was used to simulate an NE with an object model corresponding to Fictional_NE.

11.3.1 The testing setup

For these tests, it has been decided to use SNIA CIMOMs equipped with static instances of the Fictional_NE class as the CIM Servers. The instances being static means that no values or states in these instances can be changed, but as the SET directive for the prototype is not implemented, this will not matter. The simple tool to evaluate the tests is the UNIX program 'tcpdump', that can be used to print the traffic on the network into standard out or to a file. The printouts from tcpdump will give the size of all packets sent and also the timestamps for when they were sent. This data can then easily be used to calculate bandwidth usage for different calls to the Fictional_NE instances.

For the SNMP tests, the Simple Agent Pro simulator was used for the NE, and a MIB browser used to perform the SNMP method calls.

11.3.2 The results of the tests

The abbreviations listed below are used in the table showing the test results. All calls marked with * means that this call has no SNMP equivalent and the test is only preformed on WBEM.

#Packets = The total number of TCP (UDP for SNMP) packets sent back and forth between the agent and the client programs.

BW = bandwidth usage.

Req BW = request BW, equals total payload of request packets.

Rep BW = reply BW, equals total payload of reply packets.

Tot BW = total BW usage, including TCP and IP headers and also includes packets with no payload.

Get Prop = Get Property call, retrieves a single property value from NE. In SNMP, this is done with the Get call.

Get Inst = Get Instance call, retrieves an entire instance in a single call. In SNMP, this is done by using the GetBulk command listing all attributes.

Scan = Scans a CIMOM for all units of the specified class, in this case, two instances are located.

Class Nm = Enumerate class names, retrieves all classes known to the CIMOM, in this case 64 class names are fetched (that is, all the classes of the CIM Core model).

The numbers listed in the table below represent a typical result out of four test runs, where two have been at the local host computer and two against a remote host. The variations between these different runs were very small or none at all, and the only difference might be an extra packet for the connection set-up.

TABLE 1. Test results

| Call type | WBEM | | | | SNMP | | | |
|------------|----------|--------|--------|--------|----------|--------|--------|--------|
| | #Packets | Req BW | Rep BW | Tot BW | #Packets | Req BW | Rep BW | Tot BW |
| Get Prop | 8 | 1014 | 494 | 1828 | 2 | 62 | 67 | 185 |
| Get Inst | 9 | 1083 | 4376 | 5819 | 2 | 60 | 354 | 470 |
| Scan * | 34 | 3441 | 2199 | 7000 | +++ | +++ | +++ | +++ |
| Class Nm * | 12 | 939 | 3202 | 4521 | +++ | +++ | +++ | +++ |

As the table shows, the WBEM requests require quite a lot of bandwidth. For example, a simple request for one property value for a specified NE means that almost 1kB (1014 bytes) of data has to be sent. The reply data size of almost 500 bytes is also quite large as the data we are really interested in is a string of about 20 bytes length. From the table, one can also see that a call to get an entire instance has about the same amount of request data size (as Get Prop), but the reply is almost tenfold of that for Get Prop. As there are 8 properties to fetch in this call plus some other information on the instance, it seems as the reply size grows proportionally to the number of properties returned.

The reason that Scan calls require such a large number of packets sent is that it first has to list all the available NEs and then perform a simple operation on the instances found to check that they really do exist.

Comparing the WBEM size figures to those of SNMP, one can see that the two different calls both have about 15 times as large amount of request data in WBEM, and almost 5 times as many packets are sent. For the replies, here WBEM has about 7 times the data size for getting one property, while it reaches 12 times the SNMP data size when getting the entire instance. The figures for total bandwidth usage point towards WBEM using something in the range of 10 times the bandwidth of equivalent SNMP calls. With an bandwidth usage increase of around 1000%, the question is if WBEM will steal too much bandwidth to be used in many systems. The answer is that this will depend both on the network and on how the network is managed. As for configurations management (i.e. all the examples above), this will

only be used at irregular intervals and will put such a small load on the network that even a 1000% increase will hardly be noticeable.

As this report is unable to test alarm management in WBEM, one can only guess at the impact on the load. Assuming the same load increase as the commands above, this could force a capacity increase on many networks. This is due to the fact that the alarms often arrive in bursts, as a single failure on an NE could generate tens of new alarms reporting failures that are dependent on that first failure. When the size of these bursts is increased by a factor of ten, we could experience lots of congestion in the network. WBEM will (as discussed earlier) try to reduce the number of alarms sent by the setting of filters on the NE itself, so hopefully the problem anticipated here can be avoided.

12.0 Conclusions

12.1 The future of WBEM

The WBEM standard seems like something that will have a large impact on network management. Many other standards have tried to become The standard for network management, but the only one who can even claim to have succeeded is SNMP. The first reason why WBEM looks set to become the next giant in the industry is the backing it receives from all major industry players. When companies like Microsoft, Cisco and Sun all are a part of developing the standard and implementing support for it in their products, it is hard to see a reason why WBEM would not become widely spread. The second reason for WBEM's growing popularity is of course the use of CIM. CIM seems to be able to model just about any manageable object, and doing this in a standardized, easy to understand fashion. The richness of CIM combined with the large amount of standard operations offered by WBEM, it looks like WBEM could be applied in almost any area of network management.

A third important reason for believing in the future of WBEM is that it does not try to replace the current giants (such as SNMP), but instead offers mappings and integrations of these standards to bring them into WBEM. This choice will not force a replacement of millions of network elements already running around the globe, all that needs to be done to keep these is to add a WBEM interface on top of its current interface.

Yet another strength of WBEM is the security features offered by HTTP. The stronger modes of security (SSL/SHTTP) are the same security used in e-business. They are thoroughly tested and have already proven their worth in these situations.

Turning to the downsides of WBEM, the first objection that everyone has about it is the performance penalties in using HTTP and XML for transport encoding. This objection certainly holds true as the comparisons in this report show. When it comes to configuration management, these penalties will not make much of a difference, as only a very small part of the network traffic will consist of commands to configure the units. When turning to alarms management, this problem could become much more severe as there is really no limit to the amount of alarms a unit can generate. As the alarm specification is not yet finished at the time of this report, the severity of this problem is yet to be tested.

Another problem with WBEM will initially be the number of operations and the dynamical nature of object modelling. CIM might simply be too rich and the opera-

tions offered might be too many for the people implementing WBEM solutions. A large part of the operations offered will be a complete overkill for most cases and developers might shy away from a standard that offers too much work to implement. As WBEM spreads and becomes more accepted, this should no longer pose a problem. The fact that there already are SDKs for development of WBEM applications also promises to facilitate this problem.

It is important to remember that the success of WBEM will not hinge primarily on how well it is defined, the clincher will be to get users to accept the standard and integrate it into their systems. The reason that earlier standards in this area have failed is that the makers have not managed to convince the would-be users that their standard is the solution to network management problems. Having the support of the companies in DMTF, the WBEM initiative has come far towards this difficult goal. The future will have to tell if it can be fully realized.

12.2 WBEM in NMS

The implementation of the prototype has proven that it is possible for NMS/TeMIP to make use of WBEM to manage network elements. It has also shown that the design criteria placed upon the developer by TeMIP will very much limit the usefulness of WBEM, as TeMIP needs the object model to be handled at compile-time and not during runtime. But even a restricted version of WBEM will be useful for the NMS as the current usage of NEs is not meant to handle changes in the object model during runtime.

The prototype has also shown that it is possible to develop a framework for WBEM AMs, without having to code every new NE from scratch. This conclusion could be most valuable if/when many different kinds of WBEM units start entering the market.

13.0 References

- [1] Distributed Management Task Force, Inc. (August 11th, 1999). Specification for CIM operations over HTTP v1.0, URL: http://www.dmtf.org/download/spec/xmls/CIM_HTTP_Mapping10.htm (2000-11-20).
- [2] Distributed Management Task Force, Inc. (July 20th, 1999). Specification for the representation of CIM in XML v2.0, URL: http://www.dmtf.org/download/spec/xmls/CIM_XML_Mapping20.htm (2000-11-20).
- [3] Distributed Management Task Force, Inc. CIM XML DTD v2.0, URL: http://www.dmtf.org/download/spec/xmls/CIM_DTD_V20.txt (2000-11-20).
- [4] J. Case, K. McCloghrie, M. Rose, S. Waldbusser (January 1996), Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC1905, URL: <http://www.rfc-editor.org/rfc/rfc1905.txt> (2000-11-20).
- [5] U. Blumenthal, B. Wijnen (April 1999), User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), RFC2574, URL: <http://www.rfc-editor.org/rfc/rfc2574.txt> (2000-11-20).
- [6] J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, L. Stewart (January 1997), An Extension to HTTP: Digest Access Authentication, RFC2069, URL: <http://www.rfc-editor.org/rfc/rfc2069.txt> (2000-11-20).
- [7] E. Rescorla, A. Schiffman (August 1999), The Secure HyperText Transfer Protocol , RFC2660, URL: <http://www.rfc-editor.org/rfc/rfc2660.txt> (2000-11-20).
- [8] R. Fielding, UC Irvine, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee (June 1999), Hypertext Transfer Protocol -- HTTP/1.1, RFC2616, URL: <http://www.rfc-editor.org/rfc/rfc2616.txt> (2000-11-20).

Appendix A List of WBEM standard methods

- Associators
- AssociatorNames
- CreateClass
- CreateInstance
- DeleteClass
- DeleteInstance
- DeleteQualifier
- EnumerateClasses
- EnumerateClassNames
- EnumerateInstances
- EnumerateInstanceNames
- EnumerateQualifiers
- ExecQuery
- GetClass
- GetInstance
- GetProperty
- GetQualifier
- InvokeMethod
- ModifyClass
- ModifyInstance
- References
- ReferenceNames
- SetProperty
- SetQualifier

In addition to these, a few more methods will most likely be added for the handling of WBEM indications.

Appendix B The Fictional_NE MOFs

The Fictional_NE MOF

```
class Fictional_NE : CIM_ManagedSystemElement {
    [key]
    string Identity;
    sint32 NrAccesses;
    boolean isRunning;
};
```

CIM_ManagedSystemElement from CIM Core Schema 2.4

```
[Abstract, Description (
    "CIM_ManagedSystemElement is the base class for the System "
    "Element hierarchy. Membership Criteria: Any distinguishable "
    "component of a System is a candidate for inclusion in this "
    "class. Examples: software components, such as files; and "
    "devices, such as disk drives and controllers, and physical "
    "components such as chips and cards.") ]
class CIM_ManagedSystemElement : CIM_ManagedElement
{
    [Description (
        "A datetime value indicating when the object was installed. "
        "A lack of a value does not indicate that the object is not "
        "installed."),
        MappingStrings {"MIF.DMTF|ComponentID|001.5"} ]
    datetime InstallDate;
    [MaxLen (256), Description (
        "The Name property defines the label by which the object is "
        "known. When subclassed, the Name property can be overridden "
        "to be a Key property.") ]
    string Name;
    [MaxLen (10), Description (
        " A string indicating the current status of the object. "
        "Various operational and non-operational statuses are "
        "defined. Operational statuses are \"OK\", \"Degraded\", "
        "\"Stressed\" and \"Pred Fail\". \"Stressed\" indicates that "
        "the Element is functioning, but needs attention. Examples "
        "of \"Stressed\" states are overload, overheated, etc. The "
        "condition \"Pred Fail\" (failure predicted) indicates that "
        "an Element is functioning properly but predicting a failure "
        "in the near future. An example is a SMART-enabled hard "
        "drive. \n"
        " Non-operational statuses can also be specified. These "
        "are \"Error\", \"NonRecover\", \"Starting\", \"Stopping\", "
        "\"Service\", \"No Contact\" and \"Lost Comm\". \"NonRecover\" "
        "indicates that a non-recoverable error has occurred. "
        "\"Service\" describes an Element being configured, maintained, "
        "cleaned, or otherwise administered. This status could apply "
        "during mirror-resilvering of a disk, reload of a user "
        "permissions list, or other administrative task. Not all such "
        "work is on-line, yet the Element is neither \"OK\" nor in "
        "one of the other states. \"No Contact\" indicates that the "
        "current instance of the monitoring system has knowledge of "
        "this Element but has never been able to establish "
        "communications with it. \"Lost Comm\" indicates that "
        "the ManagedSystemElement is known to exist and has been "
        "contacted successfully in the past, but is currently unreachable."),
        ValueMap {"OK", "Error", "Degraded", "Unknwn", "Pred Fail",
            "Starting", "Stopping", "Service", "Stressed",
            "NonRecover", "No Contact", "Lost Comm"} ]
    string Status;
};
```