Developing a Fore Technology Adapter for the Integrated Network Management Services

Antonio Rivera
Ludwig-Maximilians-Universitaet

June 15, 1999

Contents

1 Integrated Network Management Services

1.1 INMS and network management

INMS (actually a family of products) is an umbrella network management system developed by Siemens for the management of heterogeneous networks. It integrates various network technologies and provides the user with a uniform global view, which can encompass either the entire network or a part of it as defined by the network administrator.

The design of the INMS system takes into account the growing complexity and heterogeneity of corporate and carrier networks. In its current version, it supports a large number of different networking technologies, types of network elements, and network services. The spectrum of supported network scenarios ranges from LAN, MAN, and WAN architectures through to voice transmission networks.

In order to understand the motivation behind the design of the INMS family of products, it is important to have some knowledge of today's networking landscape. Corporate networks are made up of subnetworks with different architectures managed by different element management systems. Since element managers are designed for subnetwork-specific management tasks, they cannot cooperate with the management systems of other subnetworks. In general, they use different management protocols and different information models, and operate on different management platforms. To enable communication throughout the entire network, the management activities of the subnetworks must be coordinated and the subnetwork-specific management information must be correlated.

INMS integrates the configuration, status, and performance information of the element management systems in homogeneous networks and combines it into a uniform, superordinate network management system using a standardized network management protocol (SNMPv1), a global network model, and a common user interface.

Because it uses a global network model, INMS is open to the integration of new technologies. Moreover, since the definition of the INMS data model is protocol-independent, it is not limited to SNMP and can, if required, be upgraded to a bilingual SNMP/CMIP version.

INMS is based on a standard management platform, HP OpenView Network Node Manager, and is currently available for SUN Solaris and Windows NT Workstations. Its graphical user interface is based on UNIX/Motif and the HP OpenView mamagement platform.

1.2 INMS Gate

Also known as GateNM or just Gate, it provides a distributed front-end component for INMS with the purpose of monitoring a set of heterogeneous network elements. In order to perform this task, Gate implements the following features:

Integration of third party agents. Gate provides a framework for the integration of new network technologies into the INMS, without having to modify other INMS product components. Third party agent integration has to be done by a developer (also known as a customizer in INMS documentation). This customizer makes use of the Prolog API of Gate to implement an interface to the new technology.

Distributed Network Management. Gate is implemented as a mid-level manager, able to monitor a set of (possibly heterogeneous) network elements assigned to it by the network administrator.

Configurable Alarm Filtering and Correlation. Gate's Prolog API allows the customizer to define (optional) alarm filtering and correlation rules.

Graphical User Interface. Gate includes a GUI for maintenance purposes —the Gate MAI— that can be helpful for the developer to test the Technology Adapter's functionality.

The INMS Gate allows the integration of SNMPv1 agents and proxy agents, even though the respective MIB might be in SNMPv2 format. The integration of agents and proxy agents providing an ASCII-based interface is also possible with the help of the Mediation Device component, which transforms incoming data streams into generic data structures. Thus, Gate acts as a SNMPv1/SNMPv2 agent, delivered with a MIB in SNMPv2 format.

Only the Gate Server needs to be modified in case of third party agent integration. The Gate Server is responsible for agent monitoring, agent autodiscovery and the retrieval of service data, although the latter has not yet been implemented in version 1.0. Monitoring and configuration data are stored in a cache that is maintained and updated independently of SNMP get and set requests of upper-layer applications.

1.2.1 Architecture of GateNM

Gate is based on a client-server architecture. The clients are realized as Emanate subagents communicating with the Emanate Master Agent via an Emanate proprietary protocol. It is the Master Agent that provides the SNMP interface of Gate to the INMS products CoreNM (INMS Server), ViewNM and HelpNM. The Master Agent has been compiled in the trilingual mode in order to support SNMPv1 as well as SNMPv2 and SNMPv2c. Two clients have been foreseen for GateNM, namely the Monitoring Subagent and the Service Subagent. The Monitoring Subagent implements the mechanisms for retrieving monitoring and configuration data, while the Service Subagent is responsible for retrieving service data.

The version 1.0 of GateNM only implements the Monitoring Subagent. The MIB-II Agent is an option to additionally handle MIB-II SNMP requests for TCP/IP management, and is delivered by Emanate. It is not necessary for the Gate functionality and needs not to be described here.

The Gate functionality relies on the following types of interprocess communication:

- Communication between INMS Database Export and Gate Server. Reconfiguration of all Gate instances (i.e. Gate Server instances) takes place after an INMS database export. Communication takes place via rcp of the respective files and the subsequent request to reconfigure.
- Communication between Gate MAI and Emanate Master Agent. For example: call start/stop scripts, Emanate command utilities (e.g. getone), modify configuration files.
- Communication between Gate MAI and Emanate Subagents (MIB-II, Monitoring, Service), like calling start/stop scripts, modifying trace configuration, and so forth.
- Communication between Gate MAI and Gate Server, like starting/stopping a process, Gate maintenance, etc.

Communication between the Gate MAI and other processes is based on TCP/IP.

1.2.2 Implementation of GateNM

The management tasks of Gate can be subdivided in tasks concerning configuration management (the retrieval of configuration data for INMS autodiscovery), agent monitoring (retrieval of monitoring data and trap handling), and those tasks concerning performance management (standard MIB access to service data).

The Emanate Master Agent forwards incoming get requests via an internal dispatcher table to the subagent responsible for the respective MIB objects. Since get requests on different subagents do not mutually affect each other, two individual subagents have been realized: A monitoring subagent for the retrieval of monitoring and configuration data, and a service subagent for the retrieval of service data.

This procedure is necessary because get and set requests on monitoring, configuration and service data will be performed synchronously between a subagent and the Gate Server. Thus, processing a set or get operation would block the processing of any other management task when realized with just one subagent. Using two different subagents guarantees that a get request on service data won't block agent monitoring tasks and viceversa. Both subagents act as clients of the Gate Server.

1.3 Technology integration

INMS's CoreNM component provides the functionality that allows different technologies to be integrated via their ASCII, bitstream, or SNMP interfaces. In this case, CoreNM implements an open gateway for existing and new network management systems and network elements on the market. To this end, CoreNM provides both a runtime environment for technology-specific adaptation modules and a powerful development environment for their creation.

In principle, the entire INMS functionality, from topology and configuration recording through to status monitoring and help desk, is available for technologies integrated in INMS in this way. However, the level of integration that can be achieved for a specific technology is essentially determined by the functionality and information content of its management interfaces. Moreover, the number of alarms forwarded by the respective technology can be effectively reduced with the aid of intelligent filtering and correlation mechanisms in accordance with individual operating requirements.

1.3.1 Prolog API for Technology Adapters

The Prolog API for technology adapters is a library of Prolog predicates and modules built into the Gate Server that constitutes the Gate framework. The INMS designers provided this API with the purpose of making it simpler for the developer to implement complex control flows and help him/her focus on the essential relationships between the agent's MIB and the INMS information model.

It is of course difficult to agree with or contest the previous claim based only on the experience acquired developing the Technology Adapter described here. If this experience is to be used as a measure of the amount of work required for the task of technology integration, it is certainly true that defining Prolog rules is a much less complex task than implementing the same functionality in a procedural language like C. However, a customizer without any background on logic programming but seasoned in, say, object oriented software development might have disagreeed on this point.

1.3.2 Module concept for Technology Adapters

Each technology adapter is an IF-Prolog module with a common external interface used by the Gate framework. The module concept for the technology adapters is similar to the concept of class used in object oriented programming. A specialized or extended technology adapter module inherits methods from the derived base technology adapter, much in the way of virtual methods in C++.

The Basic technology adapter is what can be considered as the root of this hierarchy of modules. It exports "callback predicates" for cyclic polling, poll responses and traps to be called by the "callback" module of Gate. The callback predicates in turn call control, optional and mandatory methods. Methods are predicates used like virtual C++ methods that are inherited to extended agent handlers.

The Fore technology adapter presented in this paper extends the MIB-II module, which in turn extends the Basic SNMP module. In the sample code presented below, the Prolog fact `extends('MIB-II')` is used to indicate this inheritance relationship.

```
% NAME: 'Fore' - GateNM Customized Agent Handler 'Fore'
% DESCRIPTION:
% Frame generated by createAgentHandlerTemplate('Fore', 'MIB-II').
:- module 'Fore'.
:- begin_module 'Fore'.
:- import(framework).
:- import('Basic').
extends('MIB-II').
```

1.3.3 The INMS Data Model

The first step in the process of third party integration is to model the network entities of the new network technology within the framework of the INMS data model. Thus, the very first thing the developer has to do is understand the INMS data model, which is the basis for the Gate Monitoring MIB. The essential parts of the INMS data model are:

Proxy Agent. A proxy agent is modelled as a software module that is installed on a workstation. It manages a definite number of assigned physical nodes, physical modules, physical and logical ports, physical links and logical links. The workstation the proxy agent is installed on is modelled as a physical node.

Physical Node. For example, switches, routers and workstations are modelled as physical nodes. A physical node may contain physical modules. It can be managed directly by Gate or via a proxy agent.

Physical Module. For example, I/O cards, slots, units, and modules are modelled as physical modules. A physical module must be contained by a physical node and may contain physical ports. If the network equipment in question doesn't support physical modules but does support physical ports, a dummy physical module must be defined and the physical ports assigned to it.

Physical Port. For example, ports and interfaces are modelled as physical ports. A physical port must be contained by a physical module and must contain at least one logical port. If the network equipment in question doesn't support such logical entities, a dummy logical port must be assigned to each physical port.

Logical Port. For example, channels and interfaces are modelled as logical ports. A logical port must be contained by a physical port.

Physical Link. A physical link connects two physical ports on two separate physical nodes.

NodeToNodeLink. A nodeToNodeLink is defined as a physical link connecting two nodes where the respective end physical ports are unknown.

NodeToPortLink. A nodeToPortLink is defined as a physical link connecting two nodes where the physical port of one end point of the link is not known.

LogicalLink. A logical link connects two logical ports. It doesn't contain information about the intermediate nodes and links in the path connecting the two end points of the connection object. For example, FrameRelay PVCs and ATM VPCs and VCCs are modelled as logical links.

In general, all these objects have an alarm state and an operational state. The TA developer must define rules in order to model the network technology within the INMS data model, as well as rules to assess the alarm and operational states of the network components supported by the new network technology. A prerequisite to develop a reasonable technology adapter is that the network technology to be integrated provides MIB objects for:

- The modelling of the network technology within the INMS data model as described above. For example, if the new network technology supports ATM PVCs, which is the case with Fore's technology, the corresponding agent MIB must provide objects that allow the definition of the corresponding Gate MIB objects, e.g. gateAtmPvcSrcVci (source VCI) and gateAtmPvcDstVci (destination VCI).
- The definition of Gate operational and alarm status MIB objects for those network entities supported by the new network technology. For example, if the new network technology supports ATM PVCs, the corresponding agent MIB must provide MIB objects that allow the definition of the operational status of the entire PVC, i.e., the gateE2eConnOperStatus MIB object.

2 Fore's ATM technology

2.1 About Fore's network element model

This section describes the hardware characteristics of Fore's ATM switches in some detail as they form the basis for the network element model of the Fore-Switch-MIB. The correspondence between Fore's and the INMS network element model is explained when appropriate. The current version of the Fore Technology Adapter supports autodiscovery and monitoring of five types of Fore ATM switch: the ASX-200WG switch for LAN workgroups, and the four models designed for LAN backbone and LAN/WAN internetworking: ASX-200BX, ASX-1000, ASX-1200 and ASX-4000. Since the logical model of a switch is maintained by the different physical configurations, the Fore-Switch-MIB applies to all switch types.

2.2 ATM switch models

ASX-200WG. Single board, single SCP (switch control processor), up to four network modules (i.e. modules with network functionality). Only one PS (power supply) module and one set of internal fans. The network modules support up to eight ports.

ASX-200BX. Single board, single or dual SCP, up to four network modules, two redundant PS modules and one set of internal fans. The network modules support up to eight ports.

ASX-1000. Up to four switch boards in a common enclosure. Single or dual SCP and up to four network modules on each board. The network modules support up to eight ports. Two redundant PS modules and a hot-swappable fan tray are shared by all boards in the enclosure.

ASX-1200. An improved version of the ASX-1000.

ASX-4000. Up to four switch fabrics, functionally equivalent to the boards in the other models. Up to four network modules controlled by each fabric. Network modules support a maximum of four ports. The entire array of fabrics is centrally controlled by a single or dual SCP configuration. The enclosure holds either four AC of five DC power supply modules and two fan banks.

2.2.1 Special characteristics of the ASX-1000 and ASX-1200

These switch types share the remarkable feature of being able to hold multiple SNMP network nodes in a single physical enclosure. They can contain up to four boards, each with its own SCP module and SNMP agent, which makes each board appear to the management system as a stand-alone ATM switch.

Of course, all boards populating an ASX-1x00 share certain hardware resources (the most important being ventilation and power supply units), but considering their independent networking and management functionality, the Fore Technolology Adapter handles each board as a logical ATM network node functionally equivalent to an ASX-200BX.

2.2.2 Special characteristics of the ASX-4000

The physical arrangement of network modules and switch fabrics (boards) in the ASX-4000 differs from that of the other switch models. The network modules are not physically embedded in the boards: they are inserted in separate slots and comunicate with the fabric over the switch backplane. Another difference lies in the physical arrangement of network modules: in the ASX-4000, each fabric supports two port cards, which in turn are logically divided into two interface groups, each containing up to four physical ports.

However, considering each interface group as a logical network module preserves the model used in the other switch types: each port card can be considered as a hardware unit consisting of one or two physical modules, which results in each fabric containing up to four network modules which in turn can contain up to four physical ports (as opposed to a maximum of eight ports in each module supported by the other switch types). The same applies to fault management: failure of a switch fabric results in its "contained" modules being out of operation.

The ASX-4000 is equipped with four AC or five DC power supply modules. Three AC or four DC PS modules are required to provide power to an ASX-4000. The fourth AC or fifth DC power supply is provided for redundancy.

2.2.3 The Switch Control Processor

The switch control processor (SCP) provides the distributed connection setup for a network of ATM switches. The SCP primarily provides management access through SNMP and is responsible for storing and updating all SNMP management information. The SCP is also responsible for monitoring the environmental conditions of the switch, and for reporting events such as malfunctioning fans, overheated power supplies, etc. It should be noted that the SCP is not involved in the actual cell switching: this task is up to the switch fabric(s) (boards). The ASX-200WG is the only switch type that does not support a dual SCP configuration. When two SCPs are installed in a switch board, the switch recognizes their presence and automatically runs in dual SCP mode, providing for failover support. In this case the SCPs are hot-swappable. In the event of a failure on the controlling SCP, the standby SCP takes over. If PVP/PVC connection preservation is enabled, all PVPs and PVCs listed in the configuration database (according to the last synchronized status) and found to be intact in the hardware are maintained without disruption of cell flow.

2.2.4 Fore's "smart" PVCs

Permanent virtual ATM connections are designated by Fore as either SPVCs (Smart Permanent Virtual Circuit connections) or SPVPCs (Smart Permanent Virtual Path Connections). The Fore TA adopts the same naming convention as it will be consistent with Fore's documentation. The use of the word smart refers to the ability of Fore's internetworking software to dynamically configure the path followed by a permanent virtual connection, which means that the administrator only needs to configure the ATM PVC/PVP at the switches interacting directly with the end points of the connection.

2.3 SNMP Interface

Fore network elements are integrated into the INMS through the SNMP agent included in the switch control software running on every Fore ATM switch. The SNMP agent enables the remote monitoring and configuration of network nodes and is configurable via Fore's ATM Management Interface (AMI). Fore's ATM Switch Network Configuration Manual contains detailed SNMP configuration instructions and is avaliable at Fore's web site.

2.4 Configuration of Fore network elements

Fore's ATM Switch Network Configuration Manual also offers detailed configuration instructions for all types of switch hardware, as well as general ATM information in a tutorial fashion.

2.5 ForeView

ForeView Network Management is the proprietary software package provided by the vendor for the management of Fore ATM networks. It is a graphical application that integrates with HP OpenView on the Solaris and Windows NT platforms. The ForeView Network Management User's Manual includes full installation and configuration instructions and is available at Fore's web site.

3 Fore Technology Adapter

3.1 Features

Upon release of the Technology Adapter described in this paper, the INMS will support in its version 3.0 the distributed monitoring of Fore networks via its Gate component. The INMS will offer, through the use of the Fore TA, the following functionality for the management of Fore backbone and workgroup switches:

- Autodiscovery of network modules for all types of Fore backbone switches
- Autodiscovery of their physical ports and configured ATM PVCs
- Autodiscovery of power supply, fan (for ventilation), and switch control processor (SCP) physical modules for all types of Fore backbone switches
- Monitoring of the alarm state of Fore nodes and of their physical modules (network modules and other types) as well as their physical ports
- Monitoring of the operational state of Fore nodes and of their physical modules (network modules and other types) as well as their ports and configured PVCs
- Alarm display of fault management relevant alarms in the HP OpenView browser with a Fore specific alarm category

Fore element	INMS element
Fore node	physNode of type ASX-200WG, ASX-200BX, ASX-1000, ASX-1200 or ASX-4000
switch board / fabric	not modelled - mapped either to contained physical modules for the ASX-4000, or to physical node for all other node types
module / port card	physModule
power supply module	physModule
SCP module	physModule
fan bank / fan tray	physModule
physical port	physPort (with 1 dummy logical port)
SPVPC	atmPVC with VCI set equal to 0 by the Fore TA
SPVC	atmPVC

Table 1. Mapping of object classes

- Display of alarm state and operational state for nodes, modules, physical ports, logical ports and PVCs in the Network and Service Browser

3.2 Hardware and software requirements

Deployment of the Fore Technology Adapter will be possible on systems fulfilling the following requirements:

- Sun Sparc/Ultra Sparc Workstations with Solaris version 2.6
- HP-Workstations with HP-UX version 10.20
- ForeView Software version 5.x.x
- Fore-Switch-MIB released on May 1997
- Fore-Common-MIB released on November 1996
- Fore-TrapLog-MIB released on October 1996
- INMS V3.0 - B and upwards

3.3 Modelling of the Fore node configuration within INMS

Table 1. shows the correlation of Fore switch components with the INMS network element model.

3.4 Configuration in the INMS database

The Fore TA supports the automatic configuration in the INMS database of nodes, modules, ports and PVCs in a Fore network, by means of the INMS autodiscovery function. As an alternative to INMS autodiscovery, it is also possible for the network administrator to manually configure the physical nodes, modules and ports through the INMS DB editor.

3.4.1 Configuring Fore nodes via the INMS DB editor

The following steps have to be performed in order to define a Fore node with a given name in the DB editor:

1. Generation of a node. A Fore node is specified in the DB editor by indicating the type of node (ASX-200WG, ASX-200BX, ASX-1000, ASX-1200 or ASX-4000) along with the given name, and activating the create button in the Physical Nodes menu.

2. Specifying properties of the node. After creating the node, the properties window is displayed automatically. The node attributes to be defined in this window are the following:

- Location. The location of the node (optional)
- Polltimer. The time interval for explicitly retrieving actual state information from the node by INMS (mandatory)
- ManagedStatus. Attribute to specify if the node should be monitored. By setting this attribute to unmanaged the network administrator can disable the monitoring for nodes currently not in operation (mandatory)
- ManagementType. This attribute has to be set to Fore and specifies the name of the Technology Adapter in Gate (mandatory)

3. Creating a Gate instance. If not already done, the Gate instances responsible for accessing the monitored Fore nodes have to be configured in the database as well. A Gate instance is created as a Software Module object of the host where it is operating (the corresponding physical node instance must have been previously created in the DB editor). In the property windows of Gate, the network administrator has to specify the corresponding PollTimer intervall and the IP address of the Gate host. The Fore nodes to be monitored by the Gate instance in question are assigned to it by choosing the Managed Nodes card in the properties window of Gate, moving the desired Fore nodes from the left to the right selection list and choosing the apply button.

4. Configuration of Modules and Ports. Physical modules and physical as well as logical ports can be entered manually in the INMS database via the DB editor. For Fore nodes, INMS alternatively supports automatic configuration of modules and ports in the INMS database via INMS autodiscovery. By default all components of a node are monitored: If Fore ports are not being used in the configuration the administrator must explicitly set the port to the disabled state, i.e. administratively down as opposed to the port's own derived down state.

5. Configuration of logical links. ATM PVCs can be configured manually in ForeView, Fore's proprietary management system. For a description of manual configuration procedures please refer to the ATM Switch Network Configuration Manual. However, through the features implemented in the Fore TA, the INMS alternatively offers the automatic configuration of PVCs via INMS autodiscovery.

Note: After entering configuration information in the DB editor a commit action is necessary in order to make the entered information permanent in the database.

3.5 Autodiscovery

INMS supports the autodiscovery mechanism for the configuration of Fore network nodes based on the MIB information present in the monitored elements. In this section, the mapping rules between Fore's MIB data and the INMS network element model followed by the autodiscovery mechanism are explained in more detail.

Physical Nodes. For the automatic configuration of Fore nodes, the network administrator has to select each node in HP OpenView before starting the INMS autodiscovery. It is also up to the administrator to assign each node to a Gate agent. The managed switch model is determined by accesing the MIB variable switchType.

```
autodiscovery_mapping(
N, % N is unified with the name given to this node
[[switchType, 0] = STypeInt],
[]
):-
s2enum(switchType, Model, STypeInt),
(switchModel(N, _) ->
retract(switchModel(N, _))
; true), !,
assertz(switchModel(N, Model)).
```

Physical Modules. The Fore-Switch-MIB implements a moduleTable containing the information necessary for the discovery of configured physical modules with network functionality.

```
autodiscovery_mapping(
N,
[[moduleName, Board, Mod] = MType],
[physModule(N, M, MType)]
) :-
concat_atom(['Board', Board, '/', Mod], M),
% map module to board number for board alarms
(module2board(N, Board, M) ->
retract(module2board(N, Board, M))
; true), !,
assertz(module2board(N, Board, M)).
```

Besides, the Fore-Switch-MIB includes additional tables containing information relative to Power supply, CPU and Fan modules. For instance, on-board CPUs are discovered with the help of the envCPUsTable.

```
autodiscovery_mapping(
N,
[[envCpuType, Board, CpuSlot] = CT],
[physModule(N, M, CpuType)]
) :-
% case of asx1000:  select CPUs on same board as agent
(( switchModel(N, asx1000) ; switchModel(N, asx1200) ) ->
physBoard(N, Board, _BType) % only if board already in database
; true
), !,
concat_atom(['CPU', CpuSlot], M),
s2enum(envCpuType, CpuType, CT).
```

Just as important as the physical presence of CPU modules is the SCP configuration of the switch, that is, the configuration of a second CPU as backup in the event of failure of the primary module. This information is gathered from the dualScpConfTable.

```
% discover if node has dual SCP configuration
user_mapping(getSCPconfig(N, Board),
[[dualScpSlot, Board] = Slot,
[dualScpState, Board] = St,
[dualScpPrimary, Board] = Prim,
[dualScpFailover, Board] = FO],
[]
):-
(St == 2 -> % switch operates with dual SCPs
FO == 1 -> % failover enabled
Config = dual
; Config = alone),
(scpConfig(N, Board, Slot, _, _) ->
retract(scpConfig(N, Board, Slot, _, _))
; true), !,
assertz(scpConfig(N, Board, Slot, Prim, Config)).
```

Physical Ports. The Fore-Switch-MIB implements a hwPortTable containing the information necessary for the discovery of configured physical ports. This table contains two different entries for the port number: one for the "local" port number, i.e. the number of the port within its network module, and the other one for the "global" port number, i.e. the number of the port within the ATM switch. Although the physPort object uses the global number for uniqueness, the local number is used by some traps, so a mapping between both numberings is performed by this rule through the predicate pPortGlobal/5. This autodiscovery rule assigns a dummy logPort object with index 1 to each discovered physical port.

```
% hwPortModule is local module number
autodiscovery_mapping(
N,
[[hwPortModel, Board, Mod, P] = PT,
[hwPortIfIndex, Board, Mod, P] = I,
[hwPortGlobalIndex, Board, Mod, P] = GI],
MappedObjects
) :-
Board ≡ 4, % not a dummy physPort for CPU
s2enum(hwPortModel, Ptype, PT),
% case of asx1000:  make sure N refers to its assigned Board only
physBoard(N, Board, _BType),
concat_atom(['Board', Board, '/', Mod], M),
MappedObjects =
[physPort(N, M, GI, Ptype), % P is local port number
logPort(N, M, GI, 1),
userPoll(mapPortNumber(N, M, P, GI, I))], !
;
MappedObjects = []
.
```

Logical Ports. The Fore-Switch-MIB does not implement logical ports. However, it does have an entry in the hwPortTable (hwPortIfIndex) with the purpose of correlating the physical port numbers to the ifIndex used in the RFC 1213 ifTable. Thus, the Fore TA assigns to each physical port a dummy logical port with index 1.

Logical Links. The Fore-Switch-MIB implements several tables containing information about configured ATM PVCs and PVPs. The pnniSpvpcSrcTable contains information about SPVPCs (Smart Permanent Virtual Path Connections) that have the managed switch as their source, and is used by the Fore TA to retrieve information about PVPs configured at the current node. Similarly, the pnniSpvcSrcTable contains information about SPVCs (Smart Permanent Virtual Circuits) that have the managed switch as their source, and is used by the Fore TA to retrieve information about PVCs. The current version of the Fore TA includes ATM PVCs and PVPs in the INMS database only when both end points of the switched part of the virtual connection are Fore ATM switches managed by GateNM.

```
% discover SPVPCs
autodiscovery_mapping(
SrcN,
[[pnniSpvpcSrcCallingPort, I] = SrcPort,
[pnniSpvpcSrcCallingVPI, I] = SrcVpi,
[pnniSpvpcSrcCalledAtmAddr, I] = DstAddr, % NSAP address
[pnniSpvpcSrcCalledPort, I] = DstPort,
[pnniSpvpcSrcCalledAssignedVPI, I] = DstVpi,
[pnniSpvpcSrcName, I] = SPVPCname,
% [pnniSpvpcSrcQosIndex, I] = QosIndex,
% [pnniSpvpcSrcFwdQoSClass, I] = FwdQos,
% [pnniSpvpcSrcBckQoSClass, I] = BckQos,
% can read qosClassName in QosClassExpansionTable
[pnniSpvpcSrcRowStatus, I] = RowStat],
MappedObject
) :-
RowStat == 1, % only entries with status = valid
% unify DstN. Succeds only if destination node in database
atmAddress(DstN, _DstBoard, DstAddr),
% nsapPrefix defined as leading 13 octects of NSAP address:
% atom_prefix(DstAddr, 13, NsapPre),
% nsapPrefix(DstN, _DstBoard, NsapPre),
% SrcPort and DstPort are global port numbers
% retrieve source and destination module numbers:
pPortGlobal(SrcN, SrcM, _SrcP, SrcPort, _),
pPortGlobal(DstN, DstM, _DstP, DstPort, _),
MappedObject =
[userPoll(getAtmPvc(I, SPVPCname, SrcN, SrcM, SrcPort, 1, DstN, DstM, DstPort,
1, 'SPVPC', SrcVpi, 0, DstVpi, 0))], !
;
MappedObject = []
.
```

PVP connections are stored in the database with a value of 0 for the VCI parameter required by the INMS atmPvc object. In the case of (smart) PVC connections, source and destination VCIs are read from the correspondig MIB table.

```
% discover SPVCCs
autodiscovery_mapping(
SrcN,
[[pnniSpvcSrcCallingPort, I] = SrcPort,
[pnniSpvcSrcCallingVPI, I] = SrcVpi,
[pnniSpvcSrcCallingVCI, I] = SrcVci,
[pnniSpvcSrcCalledAtmAddr, I] = DstAddr, % NSAP address
[pnniSpvcSrcCalledPort, I] = DstPort,
```

```
[pnniSpvcSrcCalledAssignedVPI, I] = DstVpi,
[pnniSpvcSrcCalledAssignedVCI] = DstVci,
[pnniSpvcSrcName, I] = SPVCname,
% [pnniSpvcSrcQosIndex, I] = QosIndex,
% [pnniSpvcSrcFwdQoSClass, I] = FwdQos,
% [pnniSpvcSrcBckQoSClass, I] = BckQos,
% can read qosClassName in QosClassExpansionTable
[pnniSpvcSrcEntryStatus, I] = RowStat],
MappedObject
) :-
RowStat == 1, % only entries with status = valid
% unify DstN. Succeds only if destination node in database
atmAddress(DstN, _DstBoard, DstAddr),
% nsapPrefix defined as leading 13 octects of NSAP address:
% atom_prefix(DstAddr, 13, NsapPre),
% nsapPrefix(DstN, _DstBoard, NsapPre),
% SrcPort and DstPort are global port numbers
% retrieve source and destination module numbers:
pPortGlobal(SrcN, SrcM, _SrcP, SrcPort, _),
pPortGlobal(DstN, DstM, _DstP, DstPort, _),
MappedObject =
[userPoll(getAtmPvc(I, SPVCname, SrcN, SrcM, SrcPort, 1, DstN, DstM, DstPort,
1,
'SPVCC', SrcVpi, SrcVci, DstVpi, DstVci))], !
;
MappedObject = []
.
```

3.6 Technology dependent mapping rules

3.6.1 The switch board as a container object

Since the INMS object model does not include boards as a container object, it is necessary to map board-related SNMP information to the object placed immediately above or below in the containment hierarchy, that is either to the containing SNMP node or to the contained physical modules, according to switch type.

Switch types ASX-200WG and ASX-200BX. Since these switch models contain only one board, it is natural to consider the switch board as the node itself.

Switch types ASX-1000 and ASX-1200. This switch type can contain up to four boards, each with an own SCP board and SNMP agent. Since each agent is reachable through an IP address of its own, thus making each switch board appear as an independent management entity, the Fore TA considers each board as a separate SNMP node. However, it is necessary to take into account that all boards contained in an ASX-1x00 share the same power supply modules and fan bank module. Considering that every agent will necessarily keep management information related to these modules, the Fore TA follows the vendor's MIB design and maps information about them in a redundant manner: each node (i.e. board) contained in an ASX-1x00 enclosure will appear to have an own fan bank and an own set of power supply modules.

Switch type ASX-4000. This switch type can also contain up to four boards, but unlike the ASX-1x00, all of them are managed by a single SNMP agent running on a separate SCP board in the switch enclosure. A second SCP board can be included for redundancy, but this one will only act as a backup SCP, so that the entire compound is managed at any given point by a single agent. In this case it is necessary to map all board-related management information to the physical modules controlled by each board. An advantage of this switch type in relation to an ASX-1x00 is that there is no replication of management information related to the power supply modules (up to five units) and fan banks (two units).

3.6.2 ATM Addressing

This version of the Fore TA uses the addressing scheme defined by Fore's proprietary SPANS signalling protocol in order to determine the ATM address of a managed switch. Other addressing schemes as well as end system addresses are beyond the scope of the TA's mapping rules. However, Fore switches support both private and public ATM addressing schemes, making it possible to incorporate other forms of network node addressing in future versions of the TA. The following address types are supported by Fore's management information base:

SPANS ATM address:

SpansAddress ::= OCTET STRING (SIZE (8))

ATM Forum ATM address (allows E.164 addresses):

AtmAddress ::= OCTET STRING (SIZE (8 — 20))

NSAP (Network Service Access Point) address:

NsapAddr ::= OCTET STRING (SIZE (20))

In both the AtmAddress and NSAP conventions, native E.164 addresses are represented as 8 octets using the format specified in section 3.1.1.3 of the ATM Forum UNI Signalling 4.0 specification. In contrast, an NSAP-encoded address is 20 octets, and an NSAP-encoded network prefix is 13 octets long.

3.7 Implementation

The Fore TA makes use of the following dynamic predicates in order to correlate technology specific information to the INMS data model:

physBoard/3. To implement mapping of board to node.

module2board/3. To implement mapping of board to contained modules.

pPortGlobal/5. To correlate local (i.e. within its module) and global (i.e. within the switch) physical port numbers, as well as logical indexes as defined in the RFC 1213 ifTable.

switchModel/2. Used by mapping rules based on switch type.

atmAddress/3. ATM address of each managed node. Necessary to determine end points of an ATM PVC/PVP.

atmPvcID/4. Unique identifier for each configured ATM PVC/PVP.

scpConfig/5. To store information about primary CPU module and dual/stand-alone configuration.

4 Fault management

Fore's fault management is based on an operational status model which can be almost directly mapped into the INMS network element model. Hereafter has each monitored component (node, module, port) an operational state with possible values equivalent to up, down and testing. The alarm state of Fore nodes is determined by INMS on the basis of the actual operational state of the monitored Fore node components. The monitoring of the Fore network is performed by INMS by way of polling status variables in the MIB of the Fore nodes and processing relevant traps generated by the managed entities. The relevant Fore traps are mapped into Gate alarms and are also forwarded to the View component. The next sections describe the monitoring of the operational and alarm states as well as trap mapping for Fore networks. The administrator may set individual Fore node components to the unmanaged or deleted state from the management table. In this case relevant traps take into account the administrative status of the object.

4.1 Monitoring of operational states

This section describes the determination of the operational state for nodes, modules and ports in terms of the respective MIB variables in the Fore managed entities.

4.1.1 Node level

Within the INMS network element model the operational state of the node is determined by the reaction of the node to an SNMP GetRequest: the node is up if it responds, otherwise it is down. If the node is down its alarm state is Critical.

4.1.2 Module level

The Fore TA considers four different types of physical module.

Network module. The moduleTable (moduleGroup 1), indexed by moduleBoard and moduleNumber, contains the necessary information:

- moduleState (INTEGER reset(1), inService(2), outOfService(3)), which indicates the administrative state of a given module, and
- moduleAttachState (INTEGER inService(1), outOfService(2)), which indicates its actual (operational) state

The Fore TA applies the following mapping scheme for the operational state:

- reset −> testing
- inService −> up
- outOfService −> down

```
status_mapping(N,
[[moduleAttachState, Board, Mod] = IntOp,
[moduleState, Board, Mod] = IntAdm],
[nodeAlarm(N, physModule(N, M), Severity, Reason,
physModule(N, M), '', OpStatus)]
) :-
concat_atom(['Board', Board, '/', Mod], M),
physBoard(N, Board, _BType), %if asx1000 -> only this agent's board
% some enumerated states not defined in INMS:
% s2enum(moduleState, AdmStat, IntAdm),
```

```
(IntOp == 1 ->
(IntAdm == 2 -> Status = up
;
Status = testing)
;
Status = down), !,
(Status == down ->
(IntAdm == 3 -> % AdmStat == down
OpStatus = disabled,
Reason = moduleOutOfServiceByManager,
Severity = 'Normal'
;
OpStatus = Status, % i.e.  'down'
Reason = moduleOutOfService,
Severity = 'Major')
;
OpStatus = Status,
Reason = agentPolled,
Severity = 'Normal'
).
```

Power supply module. This information is provided in the envPowerSupplyTable (powerGroup 3), indexed by envPowerSupplyIndex:

- envPowerSupplyInputState (INTEGER normal(1), fail(2)), i.e. the state of the input voltage to the power supply module
- envPowerSupplyOutputState (INTEGER normal(1), fail(2)), i.e. the state of the output voltage from this power supply
- envPowerSupplyCurrentState (INTEGER normal(1), fail(2)), i.e. the state of the current on the input return path of this power supply
- envPowerSupply5VoltState (INTEGER normal(1), fail(2)), i.e. the state of the +5V output of this power supply

Given the degree of sophistication of the technology involved, the Fore TA follows a very strict policy when assessing the operational state of a monitored node: only if all four variables have the value 'normal' does the TA map the operational state of a power supply module to the INMS value 'up', otherwise the operational state will be 'down'.

Fan bank module. The necessary information is retrieved from the envFanBanksTable (fansGroup 2), indexed by envFanBankIndex:

- envFanBankState (INTEGER normal(1), fail(2))

The status mapping is straightforward in this case.

Switch control processor module. This information is provided by the envCPUsTable (cpuGroup 3), indexed by envCpuBoard and envCpuSlot:

- envCPUState (INTEGER normal(1), fail(2), standby(3), boot(4))

The TA applies the following mapping scheme for the operational state:

- normal $->$ up

- fail $->$ down
- standby $->$ (administratively) down
- boot $->$ testing

The operational state 'standby' is only possible with a dual SCP configuration, and means that the SCP in question is playing the role of backup SCP. In case of a dual configuration, a 'down' condition in the primary SCP should be mapped as a nodeAlarm with severity 'Major', whereas the same condition with a single SCP configuration should have the severity 'Critical'.

```
status_mapping(N,
[[envCPUState, Board, CpuSlot] = IntState],
[nodeAlarm(N, physModule(N, M), Severity, Reason,
physModule(N, M), '', OpStatus)]
) :-
% case of asx1000:  select CPUs on same board as agent
( ( switchModel(N, asx1000) ; switchModel(N, asx1200) ) ->
physBoard(N, Board, _BType) % only if board already in database
; true
), !,
concat_atom(['CPU', CpuSlot], M),
% some enumerated states not defined in INMS, but useful as 'reason':
s2enum(envCPUState, State, IntState),
(
IntState == 1 ->
OpStatus = up,
Reason = agentPolled,
Severity = 'Normal'
;
OpStatus = down,
concat_atom(['State of CPU is ', State], Reason),
(scpConfig(N, Board, CpuSlot, CpuSlot, alone) ->
Severity = 'Critical' % stand-alone SCP is down
; Severity = 'Warning') % Fore:  should second CPU on board be up ?
).
```

4.1.3 Physical port level

The status information is retrieved from the hwPortTable (moduleGroup 3), indexed by hwPortBoard, hwPortModule and hwPortNumber:

- hwPortOperStatus (INTEGER other(1), up(2), down(3), unused(4))
- hwPortAdminStatus (INTEGER up(1), down(2))

The operational state 'other' is considered by the Fore TA as 'testing' for lack of an alternative in the INMS model.

4.1.4 Logical port level

Given that each physical port contains only one dummy logical port, the operational state of the latter is derived directly from the operational state of the containing port.

4.1.5 PVCs

Information about the operational state of an ATM PVC is contained in the same tables used for autodiscovery purposes. For SPVPCs, their status is retrieved from the pnniSpvpcSrcTable (q2931Group 9), indexed by pnniSpvpcSrcIndex:

- pnniSpvpcSrcStatus (INTEGER up(1), down(2)), indicating the operational state, and
- pnniSpvpcSrcRowStatus (INTEGER active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)), indicating the status of the entry, i.e. the status assigned to this SPVPC by administrative action.

Only three of these values will be returned in response to a management protocol retrieval operation: 'notReady', notInService' or 'active'. That is, when queried, an existing row can have only one of three states:

- it is either available for use by the device (active),
- it is not available for use by the device, though the agent has sufficient information to make it so (notInService), or
- it is not available for use, and an attempt to make it so would fail because the agent has insuficient information (notReady).

The Fore TA applies the following mapping scheme for the operational state:

- up and active $->$ up
- up and notReady $->$ testing
- down and notInService $->$ administratively down
- down and notReady $->$ testing
- down and active $->$ down

4.2 Implementation

The operational state of Fore nodes can be maintained using the standard algorithms supported by Gate. However, several technology specific dynamic predicates are necessary to correlate some topology and state information between the INMS data model and Fore's. These predicates are listed in section 3.7.

4.3 Monitoring of alarm states

The alarm state mapping of some hardware components of Fore nodes necessarily has to be based on switch type and/or actual configuration.

4.3.1 Nodes

ASX-200WG alarm status mapping. Since the ASX-200WG is the only switch type that supports only one SCP and only one power supply module, failure of one of them can only result in a Critical alarm state.

ASX-200BX alarm status mapping. This switch type supports a dual SCP configuration, so that the presence of a backup SCP can be determined via autodiscovery. A second (redundant) power supply module is shipped as standard configuration.

ASX-200WG	HP OV
Everything up	Normal
SCP module down	Critical
Power supply module down	Critical
Fan bank module down	Major
Any physical module down	Major
Any physical port down	Minor
Any ATM PVC down	Warning

Table 2. Alarm status mapping for the ASX-200WG

ASX-200BX	HP OV
Everything up	Normal
SCP module down	Major for dual SCP configurationCritical for single SCP
Power supply module down	Major
Fan bank module down	Major
Any physical module down	Major
Any physical port down	Minor
Any ATM PVC down	Warning

Table 3. Alarm status mapping for the ASX-200BX

ASX-1x00 alarm status mapping. As described in section 2.2, each switch board contained in an ASX-1x00 enclosure can be roughly considered as equivalent to an ASX-200BX node. As with the latter model, the presence of a backup SCP on each board can be determined via autodiscovery, and a second power supply module is always provided for redundancy. It is important for the network administrator as well as for the operator to keep in mind that all nodes contained in a single ASX-1x00 enclosure share the same PS modules and fan bank, with the effect that a single event affecting any one of these components will be reported by all on-board SNMP agents.

ASX-4000 alarm status mapping. As already mentioned in section 2.2, board alarms are mapped to their contained modules. The presence of a backup SCP in this switch type can also be determined through autodiscovery. A redundant PS module belongs to the standard configuration. Besides, this is the only model equipped with two fan banks.

ASX-1x00	HP OV
Everything up	Normal
SCP module down	Major for dual SCP configurationCritical for single SCP
Power supply module down	Major
Fan bank module down	Major
Any physical module down	Major
Any physical port down	Minor
Any ATM PVC down	Warning

Table 4. Alarm status mapping for the ASX-1x00

ASX-4000	HP OV
Everything up	Normal
SCP module down	Major for dual SCP configurationCritical for single SCP
Power supply module down	Major
Fan bank module down	Major
Any physical module down	Major
Any physical port down	Minor
Any ATM PVC down	Warning

Table 5. Alarm status mapping for the ASX-4000

4.3.2 Modules, physical and logical ports

The monitoring of the alarm states of modules, physical and logical ports is done using the standard algorithms supported by Gate. For a description of these algorithms please refer to the INMS documentation.

4.4 Trap mapping

The Fore TA processes only Fore traps signalizing an operational state change in a network component (node, module, port) monitored by INMS. Besides updating its internal MIB, Gate generates alarms containing information about the received Fore trap and forwards them to the View components in the network. The mapped Gate alarms are displayed within the View component under the alarm category "Fore".

Let us consider trap number 1093 for instance. The Prolog rule must include control structures to handle the special case of switch model ASX-4000, in which a logical node can contain more than one physical board.

```
trap_mapping(N, 6, 1093 /*asxFabricDown*/,
[[boardIndex, Board] = Board,
[trapLogIndex,0] = _],
Events
) :-
% if switchModel == asx4000:  can't report a board alarm
% since boards not modelled -> map trap to modules
switchModel(N, asx4000),
concat_atom(['Board number ', Board, ' down or hot-swapped'], Details),
% build list of alarms for all contained modules
setof(Mod, module2board(N, Board, Mod), Modules),
setof(nodeAlarm(N, physModule(N, M), 'Major', fabricDownTrapReceived,
physModule(N, M), Details, down),
member(M, Modules), Events), !
;
% if switchModel ≡ asx4000 -> board == node.
not switchModel(N, asx4000),
Events = [nodeAlarm(N, physNode(N), 'Critical', fabricDownTrapReceived,
physNode(N), 'Physical node down or hot-swapped', down)],
!.
```

Table 6 lists the mapping between relevant Fore traps and the operational state of the respective components in the INMS network element model. Table 7 lists the mapping between relevant Fore traps and Gate alarms. The Reason and EventDetails fields in the Gate alarms correspond to the reason given in Table 6.

Event (Trap Nr.)	**Object**	**OpState**	**Reason**
asxSwLinkDown (0)	physPort	down	asxSwLinkDown TrapReceived
asxSwLinkUp (1)	physPort	up	asxSwLinkUp TrapReceived
asxHostLinkDown (2)	physPort	down	asxHostLinkDown TrapReceived
asxHostLinkUp (3)	physPort	up	asxHostLinkUp TrapReceived
asxNetModuleDown (4)	physModule	down	moduleDown TrapReceived
asxNetModuleUp (5)	physModule	up	moduleUp TrapReceived
asxPsInputDown (6)	physModule (PS)	down	powerSupplyDown TrapReceived
asxPsInputUp (7)	physModule (PS)	up	powerSupplyUp TrapReceived
asxPsOutputDown (9)	physModule (PS)	down	powerSupplyDown TrapReceived
asxPsOutputUp (10)	physModule (PS)	up	powerSupplyUp TrapReceived
asxFanBankDown (22)	physModule (Fan)	down	fanBankDown TrapReceived
asxFanBankUp (23)	physModule (Fan)	up	fanBankUp TrapReceived
asxLinkDown (28)	physPort	down	asxLinkDown TrapReceived
asxLinkUp (29)	physPort	up	asxLinkUp TrapReceived
asxTempSensorOverTemp (32) (depends on location of temperature sensor)	• physNode (enclosure) • physModule (PS)	• restricted • restricted	overTemp TrapReceived
asxTempSensorRegularTemp (33) (see previous trap)	• physNode (enclosure) • physModule (PS)	• up • up	regularTemp TrapReceived
asxFabricTemperatureOverTemp (34) (mapping depends on switch type)	• physNode • physModule	• restricted • restricted	overTemp TrapReceived

Table 6. Mapping between relevant Fore traps and INMS objects

Event (Trap Nr.)	Object	OpState	Reason
asxFabricTemperatureRegularTemp (35) (see previous trap)	• physNode • physModule	• up • up	regularTemp TrapReceived
asxDualScpSyncFailure (1034)	physNode	up, but send 'Warning' alarm	dualScpSyncFail TrapReceived
asxDualScpSwitchOver (1035)	physNode	restricted	dualScpSwitchOver TrapReceived
asxDualScpHotSwap (1036)	physNode	up, but send 'Warning' alarm	dualScpHotSwap TrapReceived
asxPsCurrentDown (1068)	physModule (PS)	down	psCurrentDown TrapReceived
asxPsCurrentUp (1069)	physModule (PS)	up	psCurrentUp TrapReceived
asxPs5VoltDown (1070)	physModule (PS)	down	ps5VoltDown TrapReceived
asxPs5VoltUp (1071)	physModule (PS)	up	ps5VoltUp TrapReceived
asxFabricDown (1093) (mapping depends on switch type)	• physNode • physModule	• down • down	fabricDown TrapReceived
asxFabricUp (1094) (node might have been reconfigured)	physNode	unknown	starts node_autodiscovery
ifLinkDown (2001)	physPort	down	linkDown TrapReceived
ifLinkUp (2002)	physPort	up	linkUp TrapReceived

Table 6. Mapping between relevant Fore traps and INMS objects

Event (Trap Nr.)	Gate trap	Severity	Object
asxSwLinkDown (0)	gatePportTrap	Minor	physPort
asxSwLinkUp (1)	gatePportTrap	Normal	physPort
asxHostLinkDown (2)	gatePportTrap	Minor	physPort
asxHostLinkUp (3)	gatePportTrap	Normal	physPort
asxNetModuleDown (4)	gateModuleTrap	Major	physModule
asxNetModuleUp (5)	gateModuleTrap	Normal	physModule
asxPsInputDown (6)	gateModuleTrap	Major	physModule (PS)
asxPsInputUp (7)	gateModuleTrap	Normal	physModule (PS)
asxPsOutputDown (9)	gateModuleTrap	Major	physModule (PS)
asxPsOutputUp (10)	gateModuleTrap	Normal	physModule (PS)
asxFanBankDown (22)	gateModuleTrap	Major	physModule (Fan)
asxFanBankUp (23)	gateModuleTrap	Normal	physModule (Fan)
asxLinkDown (28)	gatePportTrap	Minor	physPort
asxLinkUp (29)	gatePportTrap	Normal	physPort
asxTempSensorOverTemp (32) (depends on location of temperature sensor)	• gateNode Trap • gateModule Trap	• Minor • Warning	• physNode (enclosure) • physModule (PS)
asxTempSensorRegularTemp (33) (see previous trap)	• gateNode Trap • gateModule Trap	• Normal • Normal	• physNode (enclosure) • physModule (PS)
asxFabricTemperatureOverTemp (34) (mapping depends on switch type)	• gateNode Trap • gateModule Trap	• Minor • Minor	• physNode • physModule
asxFabricTemperatureRegularTemp (35) (see previous trap)	• gateNode Trap • gateModule Trap	• Normal • Normal	• physNode • physModule
asxDualScpSyncFailure (1034)	gateNodeTrap	Warning	physNode
asxDualScpSwitchOver (1035)	gateNodeTrap	Minor	physNode
asxDualScpHotSwap (1036)	gateNodeTrap	Warning	physNode
asxPsCurrentDown (1068)	gateModuleTrap	Major	physModule (PS)
asxPsCurrentUp (1069)	gateModuleTrap	Warning	physModule (PS)

Table 7. Mapping between relevant Fore and Gate traps

Event (Trap Nr.)	Gate trap	Severity	Object
asxPs5VoltDown (1070)	gateModuleTrap	Major	physModule (PS)
asxPs5VoltUp (1071)	gateModuleTrap	Normal	physModule (PS)
asxFabricDown (1093) (mapping depends on switch type)	• gateNode Trap • gateModule Trap	• Critical • Major	• physNode • physModule
asxFabricUp (1094) (node might have been reconfigured)	TA starts node_autodiscovery	unknown	physNode
ifLinkDown (2001)	gatePportTrap	Minor	physPort
ifLinkUp (2002)	gatePportTrap	Normal	physPort

Table 7. Mapping between relevant Fore and Gate traps

5 Conclusion

5.1 Anecdotal

It seems appropriate to finish this paper with an account of some practical aspects of this project. A key resource for the development of an application intended to manage network nodes is (definitely) technical documentation. It is only fair to acknowledge Fore Systems for the extensive technical as well as tutorial documentation they make freely available at their web site. On the other hand, it was very difficult to obtain additional technical assistance through the channels set up with this purpose. One can't help but to ask himself if these channels are provided by the company with only customers in mind.

Anybody who has ever read MIBs knows that they are not always clear enough on their descriptions of tables and variables. Tests are often necessary in order to figure out the relationships between variables in different MIB tables, among other things. Given the fact that ATM backbone switches are hard to come by, it was a blissful event to find Fore's most advanced switch types at the Leibniz Supercomputing Center (LRZ).

While the people at the LRZ were very helpful, the workstation set apart for network management test purposes was running a Solaris 2.5 version, so that it has still not been possible to test the Technology Adapter on the LRZ's switches. An upgrade to Solaris 2.6 on the test machine —necessary to run the new version of the INMS— should be completed any time soon.

Finally, an unexpected development in the ATM market took place at the time of writing this paper. Fore Systems announced in April that it has reached an agreement with General Electric to become a fully owned subsidiary of the electrical industry giant. The news that a direct competitor of Siemens takes over the vendor of the technology intended to be integrated in the INMS product line upon completion of this study project raises interesting questions. Whatever the outcome, this paper will not handle such non-technical matters.

5.2 To-do list

As already explained above, the Technology Adapter described here has not yet been fully tested. Some (minor) corrections might prove necessary. On the other hand, Fore released a new version of the switch MIB early this year. Whereas the present version of the TA should suffice for the management of every currently available switch type, the new MIB contains some interesting extensions that might justify an improved version of the TA.

Another open question refers to the use of icons to represent Fore nodes on HP OpenView maps. While it its possible to design new icons for OV maps, the author considered that using Fore's own icons would preserve the look and feel of Fore network elements to the advantage of network operators. Thus, Fore's german branch was contacted for their permission to use these icons, with possitive results. However, Fore's representative has so far failed to fulfill his offer to provide the icons via e-mail. In the meantime, the INMS is using standard OpenView network node icons to render Fore switches on the maps.

ATM technology and implementations thereof are still under development. As standards evolve, vendors include new features and improve on the previous ones. The issue of ATM traffic monitoring alone would be stuff for another semester project in its own right. Since QOS monitoring has not yet been implemented in the INMS, traffic parameters are not handled by the current version of the Technology Adapter. But then, this one does not claim to be more than a first version.

6 References

1 Siemens AG *INMS - Top Level Design for INMS Gate V3.0.*

2 Siemens AG *INMS User Manual.*

3 Fore Systems Inc. *ForeView Network Management User's Manual.*

4 Fore Systems Inc. *ATM Switch Network Configuration Manual.*

5 Fore Systems Inc. *ForeRunner ASX-200WG, ASX-200BX, ASX-1000, ASX-1200 ATM Switch Installation and Maintenance Manual.*

6 Fore Systems Inc. *ForeRunner ASX-4000 ATM Switch Installation and Maintenance Manual.*

7 Appendix

Some INMS slides —see next pages.