

24.04.07

Deliverable DS3.13.1: common Network Information Service Schema Specification



Deliverable DS3.13.1

| | |
|------------------------|---|
| Contractual Date: | 31/01/07 |
| Actual Date: | 24/04/07 |
| Contract Number: | 511082 |
| Instrument type: | Integrated Infrastructure Initiative (I3) |
| Activity: | SA3 |
| Work Item: | 13 |
| Nature of Deliverable: | R (Report) |
| Dissemination Level | PU (Public) |
| Lead Partner | DANTE |
| Document Code | GN2-07-045v4 |

Authors: Marcin Wolski (PSNC), Stanislaw Osinski (PSNC), Pawel Gruszczynski (PSNC), Maciej Labedzki (PSNC), Anand Patil (DANTE), Ian Thomson (DANTE)

Abstract

This paper presents the schema specification of the common Network Information Service (cNIS). It begins with a brief explanation of the background, motivation and scope for cNIS. It then lists the requirements collected from various GN2 activities. The data model of the cNIS schema is presented, explaining its components and naming convention. The Appendix to this document shows the Entity Relationship diagram of the cNIS schema.

Table of Contents

| | | |
|-------|--|----|
| 0 | Executive Summary | v |
| 1 | Introduction | 1 |
| 1.1 | Background | 1 |
| 1.2 | Scope | 2 |
| 1.3 | Design | 3 |
| 1.4 | Requirements | 4 |
| 1.4.1 | Common requirements | 4 |
| 1.4.2 | SA3: Premium IP requirements | 4 |
| 1.4.3 | JRA1: Multi Domain Network Monitoring | 5 |
| 1.4.4 | JRA3: Automated Bandwidth Allocation across Heterogeneous Networks (AutoBAHN) requirements | 5 |
| 1.4.5 | JRA4: Cross Border Fibres Development requirements | 5 |
| 1.5 | Schema Integration | 6 |
| 1.5.1 | Common schema | 7 |
| 1.5.2 | IP schema | 7 |
| 1.5.3 | Ethernet schema requirements | 8 |
| 1.5.4 | SDH technology schema requirements | 8 |
| 1.5.5 | Specific extensions | 9 |
| 1.6 | Database population | 9 |
| 1.6.1 | Initial load | 10 |
| 1.6.2 | Daily maintenance | 10 |
| 1.6.3 | Manual editing | 11 |
| 1.6.4 | Audit trail | 11 |
| 1.7 | Service Interface | 12 |
| 1.7.1 | Operational Interfaces | 12 |
| 1.7.2 | Management Interface | 12 |

| | | |
|------------|---|----|
| 2 | Schema | 15 |
| 2.1 | Design Principles and Naming Convention | 15 |
| 2.1.1 | Tables | 15 |
| 2.1.2 | Columns | 16 |
| 2.1.3 | Indexes | 17 |
| 2.1.4 | Constraints | 17 |
| 2.1.5 | Views | 18 |
| 2.2 | Data Model | 18 |
| 3 | Conclusions and Future Work | 39 |
| 4 | References | 40 |
| 5 | Acronyms | 41 |
| Appendix A | Entity Relationship (ER) Diagram | 43 |

Table of Figures

| | |
|--|----|
| Figure 1. A multiple database scenario | 1 |
| Figure 2. Common database structure | 2 |
| Figure. 3. Conceptual view of the cNIS schema | 6 |
| Figure 4. Database population of the IP layer (semi-automatic) | 11 |
| Figure 5. The prototype of the cMA | 13 |
| Figure 6.ER diagram - common tables | 44 |
| Figure 7.ER Diagram - IP tables | 45 |
| Figure 8.ER diagram - Ethernet tables | 45 |
| Figure 9.ER diagram - SDH tables | 46 |
| Figure 10.ER diagram - technology specific tables | 46 |
| Figure 11.ER diagram - JRA4 specific tables | 47 |

0 Executive Summary

Different activities within GÉANT2 require network information. Currently, these activities (specifically JRA1, JRA3, JRA4 and SA3) develop, populate and use their own unique data store according to their requirements. The use of such multiple repositories results in repetition of work and the replication of data. Also, ensuring that the repositories are all synchronised and up-to-date is time consuming and difficult.

The common Network Information Service (cNIS) is a solution to this problem. cNIS designers have worked closely with representatives of the GN2 activities involved to come up with a common solution that meets each activity's requirements. This deliverable describes this work.

cNIS is an interface to a unified repository of all relevant network information about a single administrative domain for GN2 applications. There will be one instance of cNIS per domain, with data being collected and stored separately for each domain. The collection of cNIS instances work together so that a client application can receive whatever network information it requires, even if this spans several domains. Each NREN will be expected to operate an instance of cNIS themselves.

The GN2 project will develop a cNIS reference implementation to be used within the GÉANT2 network. However, it is recognised that other NRENS may choose to add the cNIS interfaces to existing systems and existing databases rather than deploy an instance of the reference implementation.

Several GN2 applications will have been deployed before cNIS is ready. Therefore, there will be a period when these applications will need to rely on their own individual, private repositories with proprietary interfaces. To ensure that the transition to cNIS is as smooth as possible, the existing interfaces will be honoured. Also, each GN2 application has been asked to specify what data they want cNIS to hold, and how they wish to collect it.

A comprehensive database schema has been produced for cNIS that satisfies the requirements of the relevant GN2 activities, while providing enough flexibility to allow for change. This is described in "Schema" on page 15.

The success of cNIS depends upon the efficiency of automatic data collection, population and verification, and methods for this have also been in development. It must be noted that plans for current cNIS only allow for the automatic collection of Internet Protocol (IP) topology information. This is because the routers that form an IP network make such information readily available using a standardised protocol (Simple Network Management Protocol, SNMP). However, the devices that make up a switching and transmission network tend to use vendor-specific, proprietary protocols, making the required information difficult to acquire.

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

The original concept of cNIS was to only hold data common across multiple applications, such as IP topology information, which is of interest to the GN2 AMPS and perfSONAR applications. However, the current scope also includes holding client-specific information on request. For the remainder of GN2, the project will investigate potential extensions to the scope of the cNIS to possibly include the storage, retrieval and analysis of real-time networking data, advanced visualisation of database contents and storage of multi-domain information, as well as support for a “trouble ticketing system” and multi-domain circuit provisioning.

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

1 Introduction

1.1 Background

Many GN2 applications require network information, and each application either already has or plans to have its own databases. This will result in multiple, near-identical databases, as illustrated in Figure 1.

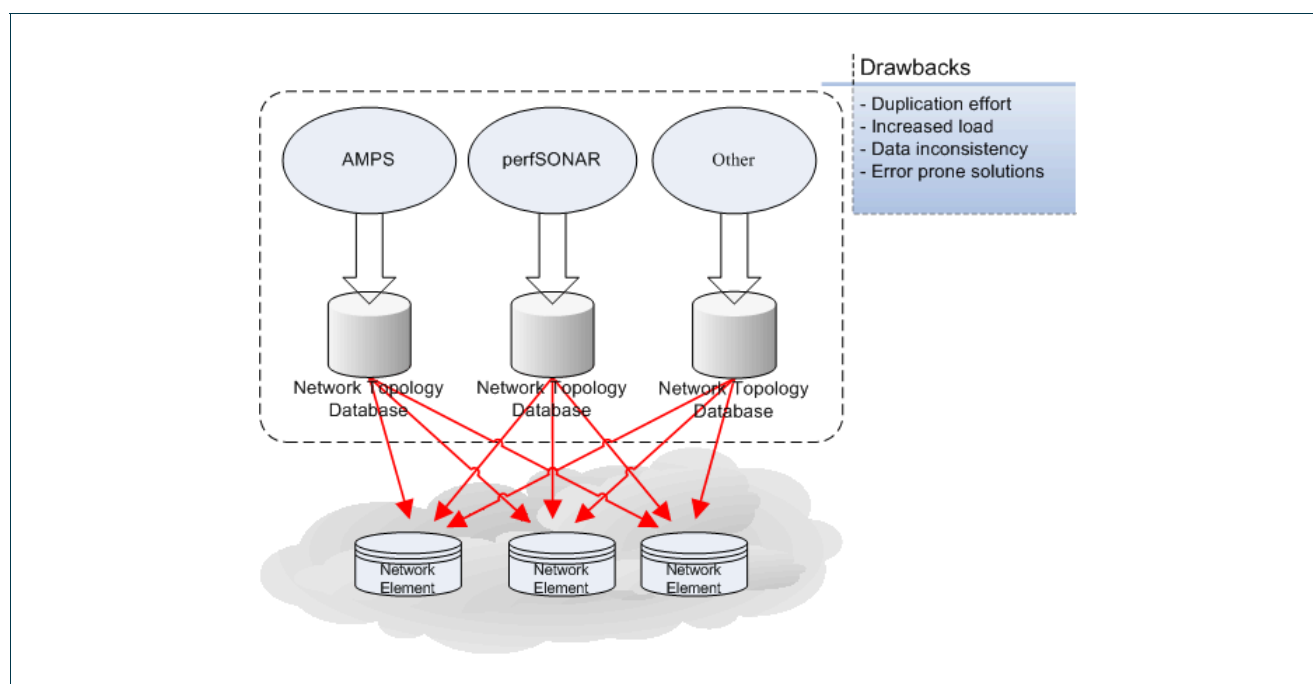


Figure 1. A multiple database scenario

However, running multiple, near-identical databases for the same network is not ideal. Multiple databases increases the maintenance effort required by network administrators, and increases the risk that a database is not updated or checked properly. Also, if there is ever a significant change to the design of the network or its components then the database schema (and therefore the database module) used by each application will have to be independently updated, and populated.

To avoid these problems, GN2 decided to create and deploy a common repository, which would store all relevant information about a given single domain's network infrastructure. This repository would be available to all GN2 as well as other authorised applications requiring network information. The repository, together with the programs required to populate and administer it, and the interfaces to other applications, is to be the Common Network Information Service, or cNIS (see Figure 2).

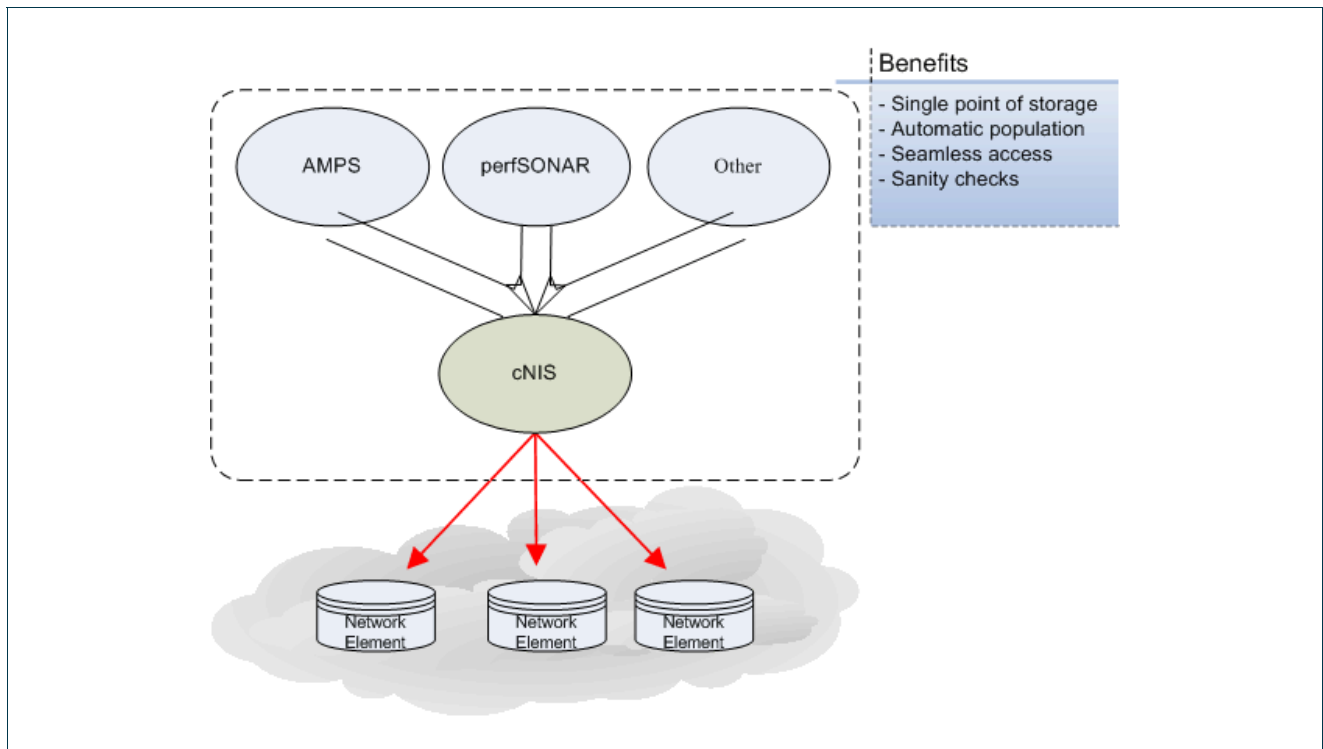


Figure 2. Common database structure

1.2 Scope

cNIS will provide an interface to a unified repository of all relevant network information about a single administrative domain for GN2 applications. Each organisation that deploys cNIS on their network will implement a separate instance of cNIS for that network.

A cNIS instance is considered to be either of:

- A cNIS interface to any existing database implementation. The implementation must support the cNIS interface and the cNIS data model.
- The cNIS Reference Implementation. This is the complete implementation of the schema described in this deliverable. The sections that follow are mostly concerned with the cNIS Reference Implementation.

It is important to note that supporting the cNIS interface and the cNIS data model are mandatory requirements for using cNIS, whereas installing the Reference Implementation is not.

The original concept of cNIS was that only data common across multiple applications would be held by cNIS. However, the current scope allows for client-specific information (which is to say, information that is relevant only to one particular client application) to be held as well.

The following issues are currently outside the scope of cNIS:

- Storage, retrieval and analysis of real-time networking data.
- Advanced visualisation of database contents.
- Storage of multi-domain information.
- Future development may include the above functionality, but is currently outside the scope of this deliverable.

Also, discussion with the interested parties has shown that there may be requirements to expand the cNIS functionality to cover other requirements (for example, to include a “trouble ticketing system” and to support multi-domain circuit provisioning). This is also for future development, and is outside of the scope of this deliverable.

Note that in the discussions that follow, a network that uses cNIS is termed a **cNIS domain**. An application that makes uses of cNIS is termed a **cNIS client application**, or a **cNIS client**.

1.3 Design

So far, cNIS development has consisted mainly of conceptual data modelling. This modelling process cannot be automated (though some tools can support syntax verification and management of the project dictionary tasks).

Conceptual data modelling involves:

- Requirements analysis. The participating GN2 activities specified what network information they would like to have stored in cNIS, concentrating on:
 - Which data are interesting
 - How this data should be represented
 - How this data should be stored
- Schema integration. Combining schemas designed by different activities into one, unified schema.
- Entity Relationship diagram design. This diagram illustrates the data entities derived from the informational model and their relationships.

1.4 Requirements

Common requirements across all activities that cNIS needs to address are described below.

A further requirement is that the resulting design is flexible enough to allow for potential future developments.

A summary of the requirements of GN2 activities (GN2 SA3, JRA1, JRA3 and JRA4) are described in the sections that follow. A separate requirements document is being prepared to describe these requirements in more detail.

1.4.1 Common requirements

cNIS must provide:

- Pointers to cNIS instances running in foreign cNIS domains. In order to concatenate information across multiple domains, cNIS applications will need pointers (for example URLs) to cNIS instances in neighbouring networks through the Lookup Services.
- Definitions of the relationships between interfaces. The server-client (or parent-child) relationship between given logical interfaces must be explicitly stated.
- Control of the relationships allowed between interfaces. cNIS must allow the administrator to define what kinds of relationships between interfaces are allowed (for example, an Ethernet interface can support an IP interface, but an IP interface cannot support an SDH interface). The relationship types permitted are configured using the administrator interface.
- Searching for specific topology elements. To diagnose a specific problem with cNIS or change a specific piece of information, an administrator may need to search for certain elements based on a set of properties; for example, the administrator may need to search for an IP node (router) based on the IP address of one of its interfaces.

1.4.2 SA3: Premium IP requirements

The Advance Multi-domain Provisioning System (AMPS) intra-domain Service will query cNIS to:

- Obtain information about the IP network structure of the local domain
- Find the shortest path between pairs of interfaces in that network.

The AMPS inter-domain service will use cNIS to:

- Obtain pointers to AMPS instances deployed in adjacent domains.
- Find the egress interface for a specific IP destination.

1.4.3 JRA1: Multi Domain Network Monitoring

The JRA1 applications that will make most use of cNIS services are the perfSONAR visualization tools, such as the CNM tool (Customer Network Management) [JRA1-CNM] and perfSONARUI (PerfSONAR User Interface).

These applications need cNIS to perform the following operations:

- Obtaining current IP topology information for a cNIS Domain.
- Obtaining a list of IP topology changes over a period of time.
- Obtaining historical IP topology information for a cNIS Domain.

1.4.4 JRA3: Automated Bandwidth Allocation across Heterogeneous Networks (AutoBAHN) requirements

JRA3 Domain Manager application needs cNIS to perform the following operations:

- Obtaining SDH information for a cNIS Domain
- Obtaining Ethernet information for a cNIS Domain
- Find links or paths between two nodes with defined parameters (for example, minimum throughput).

1.4.5 JRA4: Cross Border Fibres Development requirements

JRA4's End-to-End Monitoring System (E2EMON) requires cNIS to obtain information on the circuits E2E links use, specifically circuit ID and description, and start and end points. The following information is required for each network domain:

- Points of Presence (PoP);
- Monitored Links, which are parts of E2E links. A Monitored Link is currently realized by several interconnected optical channel (OCH) trails but in future this could be any connection-oriented service.

1.5 Schema Integration

A high-level view of the cNIS schema is shown below.

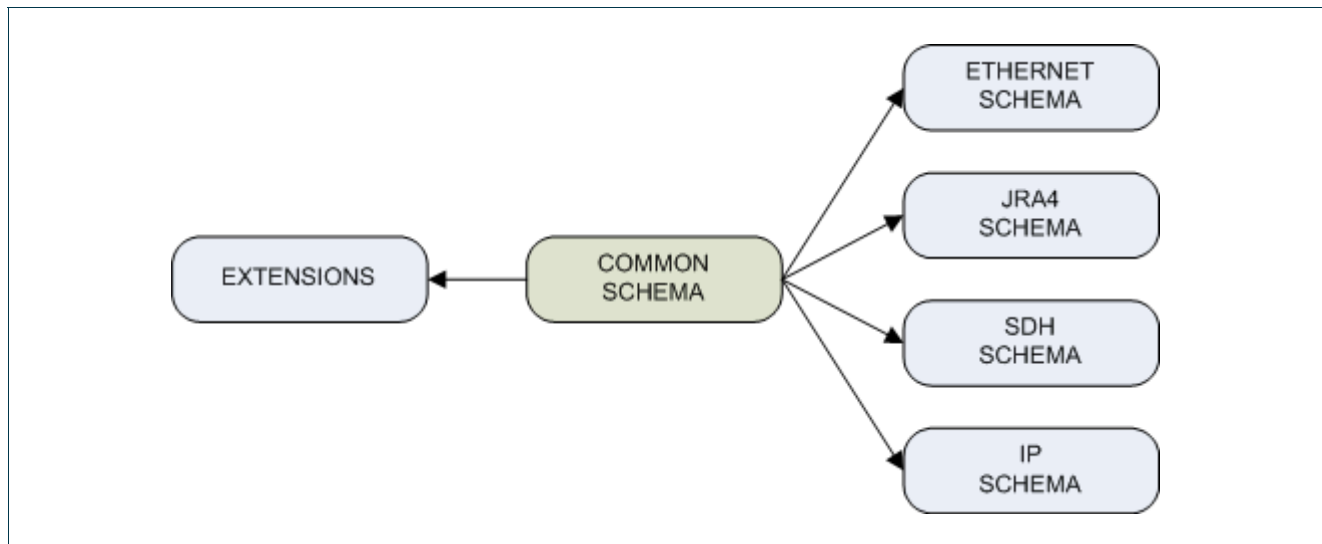


Figure. 3. Conceptual view of the cNIS schema

This design is a result of combining already existing schemas, created for the different activities, into one common schema.

The schema can be divided into:

- Activity and technology specific schemas.
- Common Schema.
- Extensions.

The Common Schema includes generic entities (such as node, interface and link). This means that the model does not contain any direct or hard-coded relationship between technology specific tables, and so the schema remains flexible and open to extensions for any other technologies.

The schema does not make any reference to the standard OSI 7 layer model, for the following reasons:

- Several technologies (including Ethernet and MPLS) do not map cleanly onto an OSI layer, and this may lead to confusion and/or long-running debates as to which table to put these protocols in.
- Techniques (such as tunnelling) may lead to a situation where a lower-layer protocol runs on top of a higher-layer protocol, leading to confusion.

The cNIS database schema allows any protocol to run over any other protocol. However, it is recognized that not all combinations of payload/carrier protocols are permitted in a given cNIS domain, or indeed technically possible (for example, SDH cannot be tunnelled over Ethernet). Therefore, when configuring their cNIS instance, operators have the option to specify which combinations are allowed and which are not. The representation and modelling of technological combinations that is possible and relevant for L1 and L2 technologies is outside the current scope of cNIS. This is a joint responsibility of the JRA3 and JRA1 activities.

1.5.1 Common schema

The common schema is that part of the database that stores characteristics common to all protocols and network technologies (for example **description** and **unique ID**). Information stored in the common schema allows the following:

- Construction of past structures. Being able reconstruct the network's past structure means it is possible to show historical measurement data in the proper context.
- Construction of future expected structures. By including information on the future expected structure of the network it is possible to predict what resources will be available in the future.
- Audit of changes. Each record holds information on who last modified the data.
- Simple enhancements. No direct or hard-coded relationship between technology specific tables so the schema can, if required, be expanded quickly and easily.

The common section is made up of more than 12 inter-related tables (see Figure 6.). The main tables are:

- Node: A node is a network device of any kind. A node is characterised as being at some geographical location and hosting one or more generic interfaces.
- Location: A geographical location including coordinates (which can be very useful for visualization purposes), address, and phone details.
- Interface: A generic interface that holds basic information (extended by reference to the technology specific tables) and has a pointer to indicate its parent interface.
- Link: A link is a connection between two generic interfaces.
- Acceptable mapping: This table specifies the acceptable mappings between interfaces.

1.5.2 IP schema

The IP schema describes the topology of the IP network elements (ISO/OSI Layer 3) in the cNIS Domain. This part of the database model is based on the common JRA1/SA3 schema.

The common JRA1/SA3 schema was a joint effort between JRA1 and SA3, conducted during GN2 Year 2. This merged the IP models that each activity was using into a common design. SA3's AMPS will use cNIS in place of its current native Pathfinder module, and JRA1's perfSONAR applications will use cNIS instead of their originally planned perfSONAR-specific topology service.

The IP schema supports both versions of IP protocol in current use; IPv4 and IPv6. The schema is made up of several entities, connected to one other appropriately. The main tables in the IP schema are:

- Extensions to the generic tables: ip_link, ip_interface and ip_node.
- Further extensions ip_interface and ip_network (for ipv4 and ipv6).
- An ip_network table.
- An AMPS specific table for holding Premium IP parameters.

1.5.3 Ethernet schema requirements

The main tables required by the Ethernet schema are:

- Extensions of generic tables: eth_link, vlan_port, eth_logical_port and eth_physical_port.
- vlan, spanning_tree, vtp_domain, (which hold information on VLANs, Ethernet spanning tree instances, and Cisco's VLAN Trunking Protocol respectively).

1.5.4 SDH technology schema requirements

The SDH technology schema is almost an exact replica of the SDH model designed by JRA3 [JRA3-SDHSchema]. As such the schema meets JRA3's requirements of showing sufficient detail of the SDH network so as to allow configuration changes to be planned (by the AutoBAHN DM), whilst also supporting the network abstraction required by the AutoBAHN IDM. Of particular note are the tables holding attributes for Ethernet VLANs and WDM.

The Alcatel 1678 MCC SDH optical cross-connect and the Alcatel 1626 Light Manager equipment, as used in the GÉANT2 network, provided the basis for the technology schema. However, wherever possible, the final schema has been made vendor independent.

The SDH schema has undergone several iterations and its latest form incorporates Alcatel's Ethernet Virtual Private Line (EVPL) functionality. The EVPL function allows a 10G Ethernet interface to be configured with multiple VLANs. Multiple Virtual Concatenation Groups (VCGs) can be associated with this Ethernet interface and each VCG can be associated with a VLAN, hence each VCG becomes an EVPL. Similar to the other technologies schemes, SDH section contains a number of tables which are extensions to generic tables (sdh_port, sdh_device, stm_link, ops_link, ho_vc_group).

The low order path has been deliberately omitted from the SDH schema since such paths have not been implemented in the GÉANT2 network.

1.5.5 Specific extensions

In addition to the common and technology-specific parts of the schema described, there are a number of specific extensions that do not fit easily into the sections above. These extensions support:

- The End-to-End (E2E) Monitoring System (E2EMON). This is being developed to monitor E2E links that traverse multiple administrative domains. This part of the schema contains two JRA4 specific tables; topology_point and monitored link. These tables are not currently associated with the generic node and link tables, because it has been specifically stated there is not requirement for such a relationship. This may change in the future.
- ATM and Frame Relay specific entities. Although no GN2 activity has specifically requested support for ATM or Frame Relay, appropriate tables have been added to cNIS in order to support these protocols.

1.6 Database population

It is assumed that organisations deploying cNIS will want, as far as possible, to populate their cNIS instance automatically (rather than manually). As such a network discovery program is included in cNIS. After a successful installation and initial set up of the cNIS base software, the cNIS administrator (see section 1.7) can populate the IP network topology database with information gathered from the domain in which cNIS is deployed. Assuming this initial load completes successfully, the cNIS database can subsequently be synchronized with the actual IP network using either a manual, semi-automatic, or fully-automatic mode.

The manual mode is used in those cases where network discovery will not detect changed attributes.

In semi-automatic update mode (which might typically be scheduled to run once a day) the cNIS operator is presented with the differences between the cNIS database and the current state of the network. The cNIS administrator can then accept or reject the changes.

Fully-automatic mode is the same as semi-automatic mode except that the changes are automatically accepted, without any review by the cNIS administrator. For the fully-automatic mode filters can be specified so that only certain characteristics are accepted automatically with all others being queued for administrator approval.

The network discovery described above currently only applies to the IP layer of the network topology. Automatic discovery of the lower layers (for example SDH or DWDM) is much more difficult to perform. The main reason for this is the lack of common standards. Hardware manufacturers typically implement proprietary management solutions for their products. Consequently, the population of the lower layer network must currently be performed manually and independently of the existing data for the IP layer. If there is a need to establish relationships between the L2 and L3 structures, the cNIS administrator must do this manually using the appropriate functions provided by the administrative interface (see section 1.7.2)

1.6.1 Initial load

The initial load of the cNIS database can be done in a number of ways. The method chosen by the cNIS administrator will depend both on the type of access that the cNIS server has to the network, and whether or not there is an existing database that cNIS must work with. The possible methods are:

- SSH and/or SNMP access: population with information automatically inferred from various kinds of data gathered from nodes (for example lists of interfaces and routing tables) using the SNMP and/or SSH protocols (in the case of SSH this means establishing an SSH connection to a node and issuing an appropriate command in order to obtain the required information). The advantage of this method is that it only requires the cNIS administrator to provide a small amount of initial data (the IP addresses of the network routers, an SNMP community name and SSH access details).
- XML file: cNIS can read in data from an XML file in a specified format. This method would normally be used if there is an existing, master database. The advantage of this method is that it does not require the cNIS administrator to arrange for cNIS to have SNMP and SSH access to the network. However, the administrator of the existing database needs to export the topology data in the cNIS specified XML format.
- Manual entry of data using the user interface. This is the most laborious method of populating the cNIS database and is recommended only as a last resort, in the event that there is no existing data and it is not possible to collect network data using SNMP and SSH.

It is theoretically possible that the cNIS database could be populated with data retrieved from an existing database without the use of the intermediate XML file format. This would require the cNIS administrator to define a mapping between the schema of the existing database and the cNIS database schema. This assumes that the cNIS administrator has enough resources and experience to understand both schemas, and is able to make sure that the data imported to the cNIS database meets all consistency criteria required by the cNIS core software. This method does not appear to have any advantages over the XML file method and is not considered any further.

1.6.2 Daily maintenance

It will be the responsibility of the cNIS administrator to ensure that the content of the cNIS IP network database is synchronised with the real state of the network. At specified intervals (for example once a day) cNIS should perform the automatic IP network discovery procedure using SNMP/SSH. If differences are found, cNIS sends a notification to specified e-mail addresses. When the administrator follows the link provided in the notification e-mail, cNIS displays a list of detected changes. cNIS then allows the administrator to select which of the changes should be transferred to the database and additionally it ensures that the choices do not lead to database inconsistencies. When the administrator decides to commit the chosen changes, cNIS applies the changes to the database.

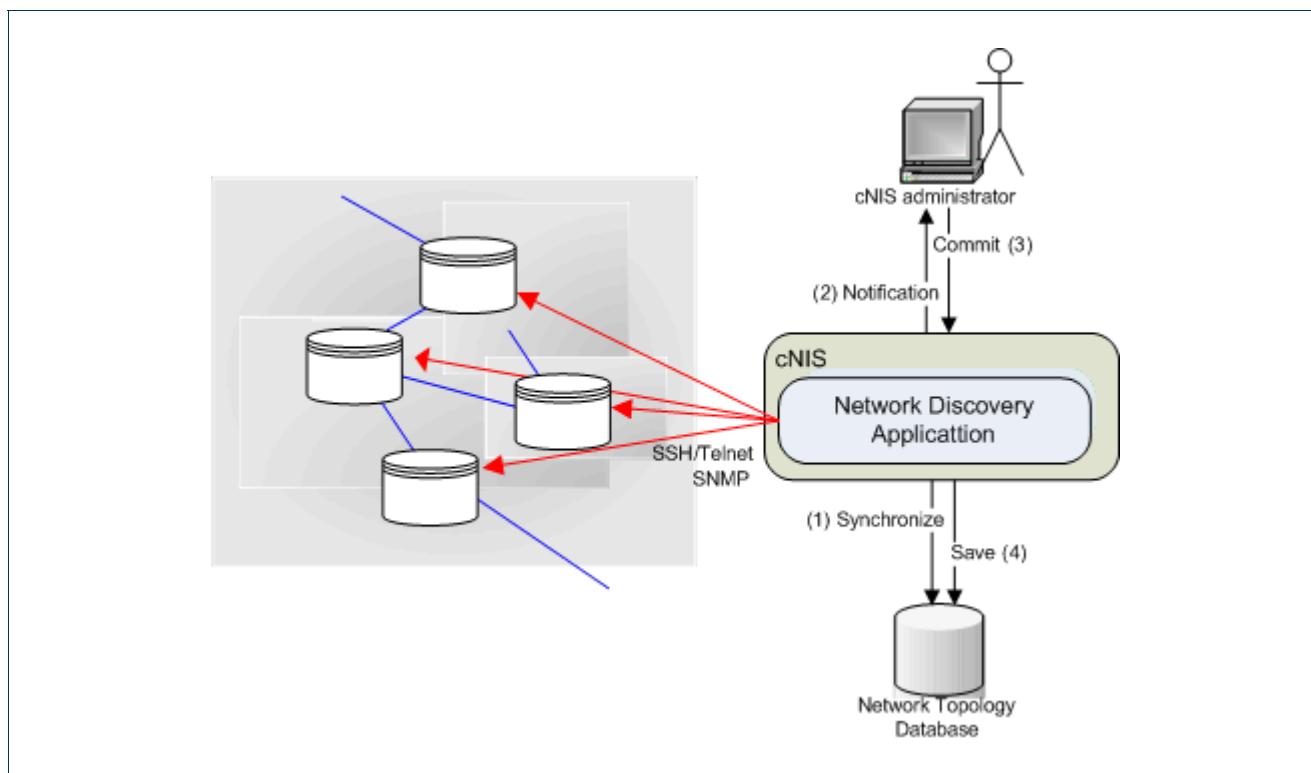


Figure 4. Database population of the IP layer (semi-automatic)

1.6.3 Manual editing

If required, it is possible for the cNIS administrator to make manual modifications to the database using the management interface. Again, as a safety measure, cNIS checks the impact of this action on database consistency. If the changes damage database integrity, cNIS prevents the modified data from being committed.

1.6.4 Audit trail

In order to diagnose and fix problems with cNIS clients, the cNIS administrator may need to inspect the changes that have been made to the database. To that end, cNIS enables the administrator to view a log of changes applied to a specific network element (for example a node or interface) or the network as a whole. The administrator may also browse logs of changes that were made during a specified period of time (between two dates, before a given date, or after a given date).

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

1.7 Service Interface

cNIS provides two different types of interface to the system:

- An operational interface for external applications. For these clients, cNIS will expose a number of communication interfaces in the form of standard APIs or NMWG compliant methods.
- A management interface for system users who connect to cNIS in order to perform various administrative tasks. These tasks will include:
 - Configuring the cNIS instance.
 - Initial population of the cNIS IP topology database using one of the available initialisation schemes.
 - Periodic maintenance of the topology database using the cNIS Management Application (see section 1.7.2).
 - Handling semi-automatic topology change notifications.

1.7.1 Operational Interfaces

In addition to what will be the primary data access interface, cNIS will offer a number of interfaces developed specifically to support existing applications. Ideally these bespoke interfaces will be temporary, such that as cNIS and its clients evolve all applications will be able to work towards one common interface. The bespoke interfaces will be kept available as long as they are required, but the ultimate goal will be to migrate all applications to the primary interface and discontinue support for the bespoke interfaces.

In the first release of cNIS the client applications will be restricted to having read-only access to the database (data modifications will have to be done by the cNIS administrators as described in section 1.7.2). As such, concurrency problems stemming from simultaneous updates are not an issue that needs to be addressed immediately.

There will be a period when existing GN2 applications will need to rely on their own individual, private repositories with proprietary interfaces. To ensure that the transition to cNIS is as smooth as possible, these existing interfaces will be honoured. To allow for this, each GN2 application is to specify what data they want cNIS to hold, and how they wish to collect it.

1.7.2 Management Interface

The user interface for cNIS administrators is called the **cNIS Management Application (cMA)**. It supports the initialization and maintenance (manual and semi-automatic) of the database. cMA does not require installing any client software on the client's workstation, other than a standards-compliant web browser.

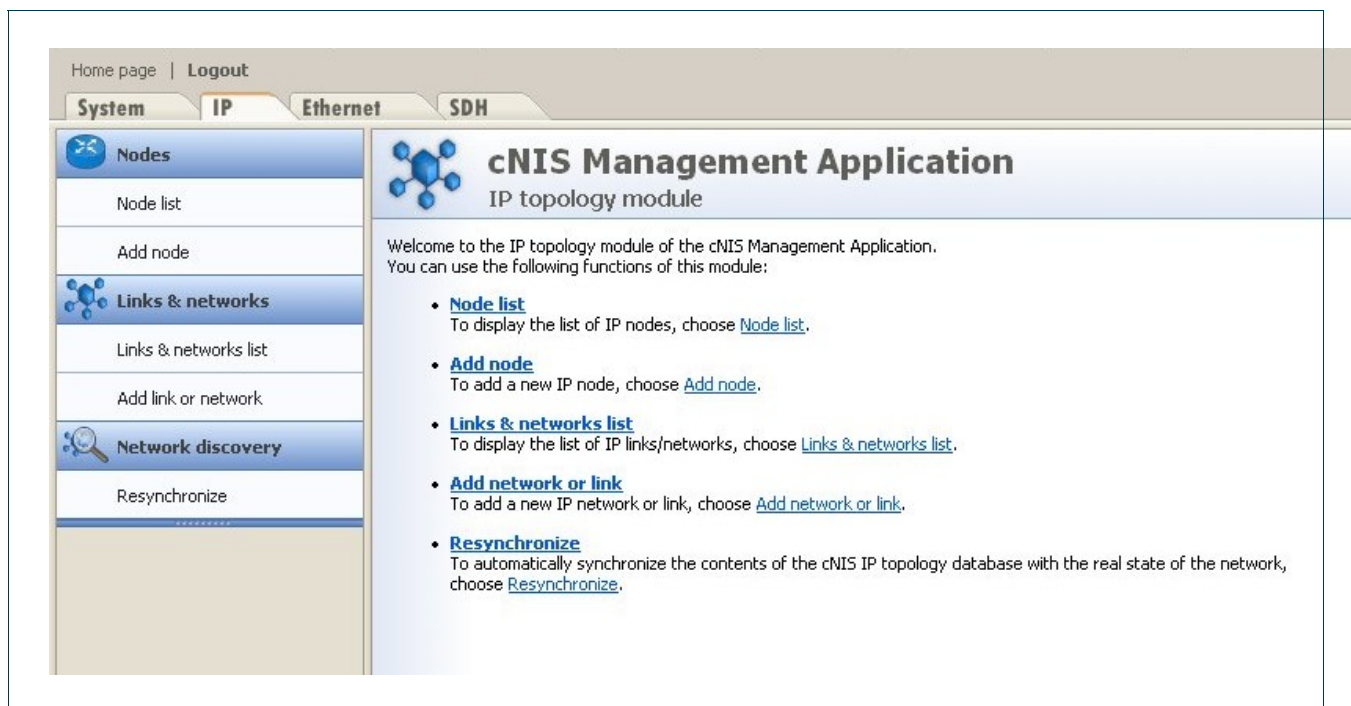


Figure 5. The prototype of the cMA

The main purpose of the cNIS Management Application is to enable network administrators to browse and modify the stored network data. The figure above is a screenshot of the cMA prototype, which was shown at the GN2 Technical Workshop in January 2007. This prototype provides the following functionality:

- System configuration:
 - User management – display, edit, delete and add user profiles (e.g. login name, real name, credentials).
 - Notification method – set details for notifications, e.g. outgoing mail server, sender's e-mail address.
 - User preferences – Set the user's email address to be used for receiving notifications; customization of the UI layout.
- IP topology data changes:
 - add, delete and edit data about Layer 3 nodes (routers), their interfaces and routing tables.
 - add, delete and edit data about adjacent (neighbouring) cNIS domains.
 - initialize the IP topology database.
 - interactively merge the existing (and possibly outdated) cNIS topology information with the most up-to-date topology.
- Ethernet topology changes:
 - add, delete and edit data about Ethernet devices, defining VLANs.
 - add, delete and edit parameters of links (bandwidth, delay).
- SDH topology changes:
 - add, delete and edit data about SDH devices.
 - configuring virtual containers and associating to VLANs.
 - add, delete and edit parameters of links (bandwidth, delay).

Note that cNIS does not create a graphical display of topology data. This is left to client applications such as CNM.

Note also that the cMA design, structure and functionality described above is subject to change, and the prototype may differ from the actual production version that will be released as part of cNIS version 1 at the end of GN2 Year 3.

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

2 Schema

2.1 Design Principles and Naming Convention

The main design principles for the tables, columns, indexes, constraints and views of the cNIS schema are described below. Also presented is the cNIS naming convention, which should be observed by any and all future developers of cNIS in order to ensure consistency in the naming of objects.

Note that the schema does not reference the OSI 7 layer model, since many of the protocols used by NRENs do not fit this model. Instead, generic tables are used for interfaces and links that hold basic information on network entities, and technology-specific extensions are used to add unique information to these generic records.

2.1.1 Tables

2.1.1.1 *Plural Names*

Table names should be plural. For table names with multiple words, only the last word should be plural.

2.1.1.2 *Prefixes*

Table prefixes can help organize tables into related groups or distinguish them from other unrelated tables. Avoid prefixes like "tbl" or "TBL_" as these are just redundant.

2.1.1.3 *Notation*

Table names:

- Should be in lowercase.
- Should use an underscore for separating words and prefixes.
- Should not contain numbers.

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

- Should not contain spaces.

2.1.1.4 Junction (Intersection Tables)

Junction tables, which handle many to many relationships, should be named by concatenating the names of the tables (with an underscore separator) that have 'one to many' relationship with the junction table. Since this convention can result in lengthy table names, abbreviations may sometimes be used (but care should be taken to avoid introducing ambiguity).

2.1.1.5 Other Table Naming Conventions

Use short, unambiguous names, using no more than one or two words. Avoid abbreviated, concatenated, or acronymic names.

2.1.2 Columns

Columns are members of the table, so their names do not need to include any mention of the table's name. The exception to this rule is the primary key field where including the table name is justified so as to have a field name more descriptive than just "id". As with the guidelines for naming tables, avoid using abbreviations, acronyms or special characters. All column names should be in lowercase with underscores as word separators.

2.1.2.1 Identity Primary Key Fields

The primary key should be the table name suffixed with "_id".

2.1.2.2 Foreign Key Fields

Foreign key fields should have the same name as the primary key to which they refer.

2.1.2.3 Prefixes

Fields should not be prefixed with "fld_" or "col_" as it should be obvious in SQL statements which items are columns (before or after the FROM clause).

2.1.2.4 Data Type Specific Naming

Boolean fields should be given names like "is_deleted", "has_permission", or "is_valid" so that the meaning of the data in the field is unambiguous. If the field holds date and/or time information, the word "date" or "time"

should appear somewhere in the field name. It is sometimes appropriate to add the unit of time to the field name, especially if the field holds data as whole numbers.

2.1.3 Indexes

Since indexes are always related to a table or view, it makes the most sense to use the name of the table or view, as well as the column(s) they index, in the index name, along with some characters that specify the type of index it is. This naming convention also allows, if looking at a list of indexes, to see the indexes ordered by table, then column, then index type.

2.1.3.1 Naming Convention

The naming convention for indexes follows this structure: {table_name}_{columns_indexed}{U/N} where "U/N" is for unique or non-unique. This naming convention is unique among database objects, so adding characters to denote it being an index, like "idx" is not necessary. The naming convention alone is self-documenting and identifies it as an index. For indexes that span multiple columns the column names should be concatenated. If a name is too long it should be abbreviated.

2.1.4 Constraints

Constraints are effective at the field/column level so the name of the field the constraint is placed on should be used in the constraint's name. The type of constraint (Check, Referential Integrity i.e. Foreign Key, Primary Key, or Unique) should also be in the name. A constraint is unique to a particular table and field combination, so its name should also include the table name so as to ensure unique constraint names across the set of database tables.

2.1.4.1 Naming Convention

The naming convention syntax for constraints is {constraint_type}_{table name}_{field name}.

2.1.4.2 Prefixes

A two letter prefix should be applied to the constraint name depending on the type:

| Constraint type | Prefix |
|-----------------|--------|
| Primary Key | pk |
| Foreign Key | fk |
| Check | ck |
| Unique | un |

2.1.5 Views

Views follow similar rules to those that apply to naming tables.

2.1.5.1 Prefixes

View name should be prefixed with "vw_".

2.1.5.2 View Types

Views should be named in accordance with the type or purpose of the view. For simple views that just join two or more tables with no selection criteria, the view name should be the combined names of the tables joined, or alternatively just the name of the main table prefixed with "vw_".

2.2 Data Model

The data model of the cNIS schema comprises a set of tables and their accompanying indexes. Each table is characterized with its name, description and constraints. Each column of the table is described with its name, data type, constraints, flags, default value and a comment. This data model also provides additional information about the indexes coupled with particular tables.

version_info

Generic table for storing information about historical and future changes.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-----------------|----------|------------|---------|----------|---------------|----------------------|---------|
| version_info_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| start_date | DATETIME | | | | | Administrative date. | start |
| end_date | DATETIME | | | | | Administrative date. | end |
| created_by | VARCHAR | | | | system | | |
| modified_by | VARCHAR | | | | system | | |
| date_created | DATETIME | | | | today | | |
| date_modified | DATETIME | | | | today | | |
| date_verified | DATETIME | | | | | | |
| change_comment | VARCHAR | | | | | | |

| IndexName | IndexType | Columns |
|-----------|-----------|-----------------|
| PRIMARY | PRIMARY | version_info_id |

location

The table containing location information.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|------------------|----------|------------------|---------|----------------|---------------|--|---------|
| location_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | business name, not unique | |
| description | VARCHAR | | | | | description | |
| country | VARCHAR | | | | | country, using 2-digit country codes, ISO 3166 | |
| city | VARCHAR | | | | | | |
| institution | VARCHAR | | | | | | |
| street | VARCHAR | | | | | | |
| floor | INTEGER | | NN | UNSIGNED | | | |
| room_suite | INTEGER | | | UNSIGNED | | | |
| row_ | INTEGER | | | UNSIGNED | | | |
| cabinet | INTEGER | | | UNSIGNED | | | |
| zip_code | VARCHAR | | | | | | |
| phone_number | VARCHAR | | | | | | |
| e_mail_address | VARCHAR | | | | | | |
| geo_latitude | FLOAT | | | | | Geographic coordinates - latitude, north[-90,90] | |
| geo_longitude | FLOAT | | | | | Geographic coordinates - longitude, east[0,360] | |
| type | VARCHAR | | | | | enum from "geo", "zone", "pixel" | |
| zone | VARCHAR | | | | | UTM zone number | |
| altitude | FLOAT | | | | | decimal metres | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | location_id | | | |

node

Generic node description, so need for separation into layer 1,2, or 3 nodes; a node is not tight to a physical device, but is a logical node.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------------|----------|------------|-----------|-----------------|---------------|---|---------|
| node_id | INTEGER | PK | NN | UNSIGNED | | unique key, given internally by database | |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| location_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | NN | | | business name, noes not have to be unique, can be used to identify predecessors | |
| description | VARCHAR | | | | | | |
| status | VARCHAR | | | | | administrative status (up, down, maintenance) | |
| vendor | VARCHAR | | | | | vendor name (e.g. Juniper) | |
| model | VARCHAR | | | | | equipment model | |
| os_name | VARCHAR | | | | | Operating System | |
| os_version | VARCHAR | | | | | Version of OS | |
| ip_address | VARCHAR | | | | | IP address (IPv4 or IPv6) for management, also for layer 1 and layer 2 nodes | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | node_id | | | |
| fk_node_location_id | | | Index | location_id | | | |
| fk_node_version_id | | | Index | version_info_id | | | |

interface_type

This table keeps interface types according to RFC-3471.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------|----------|------------|-----------|-------------------|---------------|---------|---------|
| interface_type_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| switching_type | VARCHAR | | | | | | |
| data_encoding_type | VARCHAR | | | | | | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | interface_type_id | | | |

interface_permitted_based_on

This table keeps data about permitted "interface based on interface" associations. The meaning of the row in this table is following: interface denoted by child_type can run on top of interface denoted by parent_type. It should contain all physically possible mappings.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--|----------|------------|---------|-----------|---------------------------------|--|---------|
| parent_type_id | INTEGER | PK | NN | UNSIGNED | | | |
| child_type_id | INTEGER | PK | NN | UNSIGNED | | | |
| permitted_in_domain | BOOL | | | | | Indicates if this kind of mapping is possible in local domain. | |
| IndexName | | | | IndexType | Columns | | |
| PRIMARY | | | | PRIMARY | parent_type_id child_type_id | | |
| fk_interface_permitted_based_on_parent_type_id | | | | Index | parent_type_id | | |
| fk_interface_permitted_based_on_child_type_id | | | | Index | child_type_id | | |

generic_interface

Generic interface description, layer 1,2,3 interfaces are derived. This table also keeps data of generic interfaces of neighbour domains. It allows to model connections between domains. Interfaces of other domains would have null foreign keys - details can be retrieved from other domain.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------------|----------|------------|---------|----------|---------------|--|---------|
| generic_interface_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| parent_interface_id | INTEGER | | NN | UNSIGNED | | | |
| interface_type_id | INTEGER | | NN | UNSIGNED | | | |
| node_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | NN | | | unique string for object identification; format node_id.given_by_domain; for layer 3: default: ifDescription, alternatives: ifIndex, ifAlias, ipAddress (depending on stability) | |
| description | VARCHAR | | | | | | |
| bandwidth | INTEGER | | | UNSIGNED | | bits per second | |
| status | VARCHAR | | | | | administrative status (up, down, maintenance) | |
| mtu | VARCHAR | | | | | maximum transfer unit | |
| domain_id | VARCHAR | | | | | Null value means that this interface belongs to local domain, not null means that it belongs to other domain. In case of other | |

| IndexName | IndexType | Columns |
|--|-----------|----------------------|
| PRIMARY | PRIMARY | generic_interface_id |
| fk_generic_interface_node_id | Index | node_id |
| fk_generic_interface_parent_interface_id | Index | parent_interface_id |
| fk_generic_interface_interface_type_id | Index | interface_type_id |
| fk_generic_interface_version_id | Index | version_info_id |

generic_link

Generic link table, layer 1,2,3 links are derived.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------------|----------|------------|---------|----------|---------------|---|---------|
| generic_link_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| generic_interface_id | INTEGER | | NN | UNSIGNED | | | |
| direction | VARCHAR | | | | | one-way, duplex ... | |
| protection | BOOL | | | | | if this link is protected by one or more backup lines | |
| prop_delay | FLOAT | | | | | Propagation delay. | |

| IndexName | IndexType | Columns |
|------------------------------------|-----------|----------------------|
| PRIMARY | PRIMARY | generic_link_id |
| fk_generic_link_start_interface_id | Index | generic_interface_id |
| fk_generic_link_end_interface_id | Index | generic_interface_id |
| fk_generic_link_version_id | Index | version_info_id |

generic_connection

Models reserved bandwidth on top of link.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-----------------------|----------|------------|---------|----------|---------------|---------|---------|
| generic_connection_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| generic_link_id | INTEGER | | NN | UNSIGNED | | | |
| path_id | INTEGER | | NN | UNSIGNED | | | |
| direction | VARCHAR | | | | | | |
| connection_type | VARCHAR | | | | | | |
| bandwidth | FLOAT | | | | | | |

| IndexName | IndexType | Columns |
|-----------|-----------|-----------------------|
| PRIMARY | PRIMARY | generic_connection_id |

| | | |
|----------------------------------|-------|-----------------|
| fk_generic_connection_link_id | Index | generic_link_id |
| fk_generic_connection_path_id | Index | path_id |
| fk_generic_connection_version_id | Index | version_info_id |

path

Path composed of multiple connections.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------|----------|------------|---------|-----------------|---------------|---|---------|
| path_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | | | | | |
| status | INTEGER | | | UNSIGNED | | Status of path: operational, not operational, unknown | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | path_id | | | |
| fk_path_version_id | | Index | | version_info_id | | | |

service

Table for services running on nodes.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------|----------|------------|---------|------------|---------------|---------|---------|
| service_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | | |
| description | VARCHAR | | | | | | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | service_id | | | |

node_service

This table keeps which services are running on which nodes.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------|----------|------------|---------|-----------------------|---------------|---------|---------|
| node_id | INTEGER | PK | NN | UNSIGNED | | | |
| service_id | INTEGER | PK | NN | UNSIGNED | | | |
| configuration | VARCHAR | | | | | | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | node_id service_id | | | |

| | | |
|----------------------------|-------|------------|
| fk_node_service_node_id | Index | node_id |
| fk_node_service_service_id | Index | service_id |

external_system

Table for external systems collaborating directly or indirectly with cNIS.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------|----------|------------|---------|----------|---------------|---------|---------|
| external_system_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | | |

| IndexName | IndexType | Columns |
|-----------|-----------|--------------------|
| PRIMARY | PRIMARY | external_system_id |

location_external_identifier

Mapping of cNIS location to identifier of location in external system.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------------|----------|------------|---------|----------|---------------|---------|---------|
| location_id | INTEGER | PK | NN | UNSIGNED | | | |
| external_system_id | INTEGER | PK | NN | UNSIGNED | | | |
| external_identifier | VARCHAR | | NN | | | | |

| IndexName | IndexType | Columns |
|--|-----------|-----------------------------------|
| PRIMARY | PRIMARY | location_id external_system_id |
| fk_location_external_identifier_location_id | Index | location_id |
| fk_location_external_identifier_external_system_id | Index | external_system_id |

node_external_identifier

Mapping of cNIS node to identifier of node in external system.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------------|----------|------------|---------|----------|---------------|---------|---------|
| node_id | INTEGER | PK | NN | UNSIGNED | | | |
| external_system_id | INTEGER | PK | NN | UNSIGNED | | | |
| external_identifier | VARCHAR | | NN | | | | AI |

| IndexName | IndexType | Columns |
|--|-----------|-------------------------------|
| PRIMARY | PRIMARY | node_id external_system_id |
| fk_node_external_identifier_node_id | Index | node_id |
| fk_node_external_identifier_external_system_id | Index | external_system_id |

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

interface_external_identifier

Mapping of cNIS interface to identifier of interface in external system.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---|----------|------------|---------|-----------|--|---------|---------|
| external_system_id | INTEGER | PK | NN | UNSIGNED | | | |
| generic_interface_id | INTEGER | PK | NN | UNSIGNED | | | |
| external_identifier | VARCHAR | | | | | | |
| IndexName | | | | IndexType | Columns | | |
| PRIMARY | | | | PRIMARY | external_system_id generic_interface_id | | |
| fk_interface_external_identifier_interface_id | | | | Index | generic_interface_id | | |
| fk_interface_external_identifier_external_system_id | | | | Index | external_system_id | | |

ip_node

This table keeps data of IP nodes.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------|----------|------------|---------|-----------|---------------|-------------------|---------|
| ip_node_id | INTEGER | PK | NN | UNSIGNED | | | |
| asn | INTEGER | | | UNSIGNED | | Autonomous number | System |
| IndexName | | | | IndexType | Columns | | |
| PRIMARY | | | | PRIMARY | ip_node_id | | |
| fk_ip_node_node_id | | | | Index | ip_node_id | | |

ip_interface

IP interface description.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|------------------------------|----------|------------|---------|-----------|-----------------|--|---------|
| ip_interface_id | INTEGER | PK | NN | UNSIGNED | | | |
| ip_network_id | INTEGER | | NN | UNSIGNED | | | |
| routing_cost | INTEGER | | | UNSIGNED | | routing protocol (e.g. IS-IS) routing cost | |
| IndexName | | | | IndexType | Columns | | |
| PRIMARY | | | | PRIMARY | ip_interface_id | | |
| fk_ip_interface_interface_id | | | | Index | ip_interface_id | | |
| fk_ip_interface_network_id | | | | Index | ip_network_id | | |

ipv6_interface

Keeps specific data of ipv6 interface.

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------------|----------------|------------|---------|-------------------|---------------|---------|---------|
| ipv6_interface_id | INTEGER | PK | NN | UNSIGNED | | | |
| ipv6_address | VARCHAR | | | | | | |
| ipv6_address_prefix_len | INTEGER | | | UNSIGNED | | | |
| IndexName | IndexType | | | Columns | | | |
| PRIMARY | PRIMARY | | | ipv6_interface_id | | | |
| ipv6_interface_FKIndex1 | Index | | | ipv6_interface_id | | | |

ipv4_interface

Keeps specific data of ipv4 interface.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------------|----------------|------------|---------|-------------------|---------------|---------|---------|
| ipv4_interface_id | INTEGER | PK | NN | UNSIGNED | | | |
| ipv4_address | VARCHAR | | | | | | |
| ipv4_address_prefix_len | INTEGER | | | UNSIGNED | | | |
| IndexName | IndexType | | | Columns | | | |
| PRIMARY | PRIMARY | | | ipv4_interface_id | | | |
| ipv4_interface_FKIndex1 | Index | | | ipv4_interface_id | | | |

SA3_ip_interface

SA3 IP interface extension.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------------------|----------------|------------|---------|---------------------|---------------|--|---------|
| sa3_ip_interface_id | INTEGER | PK | NN | UNSIGNED | | | |
| PIP_percentage | FLOAT | | | | | used to estimate the per link (starting from this interface) total premium IP capacity | |
| IndexName | IndexType | | | Columns | | | |
| PRIMARY | PRIMARY | | | sa3_ip_interface_id | | | |
| SA3_ip_interface_FKIndex1 | Index | | | sa3_ip_interface_id | | | |

ip_link

Table for point-to-point IP links.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------|----------|------------------|---------|----------------|---------------|---------|---------|
| ip_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | ip_link_id | | | |
| fk_ip_link_link_id | | Index | | ip_link_id | | | |

ip_network

Network consists of multiple interconnected interfaces (covers point-to-point and multipoint links).

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--------------------------|----------|------------------|---------|-----------------|---------------|---------|---------|
| ip_network_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | NN | | | | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | ip_network_id | | | |
| fk_ip_network_version_id | | Index | | version_info_id | | | |

ipv4_network

Keeps specific data of ipv4 network.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------------------------|----------|------------------|---------|-----------------|---------------|---------|---------|
| ipv4_network_id | INTEGER | PK | NN | UNSIGNED | | | |
| ipv4_network_address_prefix | VARCHAR | | | | | | |
| ipv4_network_address_prefix_len | INTEGER | | | UNSIGNED | | | |
| IndexName | | IndexType | | Columns | | | |
| PRIMARY | | PRIMARY | | ipv4_network_id | | | |
| fk_ipv4_network_network_id | | Index | | ipv4_network_id | | | |

ipv6_network

Keeps specific data of ipv6 network.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------------------------|-----------|-----------------|---------|----------|---------------|---------|---------|
| ipv6_network_id | INTEGER | PK | NN | UNSIGNED | | | |
| ipv6_network_address_prefix | VARCHAR | | | | | | |
| ipv6_network_address_prefix_len | INTEGER | | | UNSIGNED | | | |
| IndexName | IndexType | Columns | | | | | |
| PRIMARY | PRIMARY | ipv6_network_id | | | | | |
| fk_ipv6_network_network_id | Index | ipv6_network_id | | | | | |

eth_physical_port

Represents an Ethernet port on a switch.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------------------|-----------|----------------------|---------|----------|---------------|--|---------|
| eth_physical_port_id | INTEGER | PK | NN | UNSIGNED | | | |
| interface_name | VARCHAR | | | | | port's internal name (i.e. GigabitEthernet0/1) | |
| mac_address | VARCHAR | | | | | | |
| duplex | VARCHAR | | | | | Full/Half | |
| medium_dependent_interface | VARCHAR | | | | | as per 802.3 e.g. 100BASE-SX, 100BASE-T | |
| is_tagged | BOOL | | | | | is supporting trunking | |
| IndexName | IndexType | Columns | | | | | |
| PRIMARY | PRIMARY | eth_physical_port_id | | | | | |
| fk_ethernet_port_port_id | Index | eth_physical_port_id | | | | | |

eth_logical_port

Table for ethernet logical interfaces configured on one physical Ethernet interface.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-----------------------------|-----------|---------------------|---------|----------|---------------|---------|---------|
| eth_logical_port_id | INTEGER | PK | NN | UNSIGNED | | | |
| IndexName | IndexType | Columns | | | | | |
| PRIMARY | PRIMARY | eth_logical_port_id | | | | | |
| fk_eth_logical_port_port_id | Index | eth_logical_port_id | | | | | |

eth_link

Represents an adjacency between two ethernet nodes or between an ethernet node and a router.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------------|----------|------------|---------|-----------|---------------|--|---------|
| eth_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| discovery_protocol | VARCHAR | | | | | Can be 'None', 'NK', 'CDP', LLDP etc | |
| is_trunk | BOOL | | | | | TRUE if the I2_adjcy is a 802.1Q trunk else FALSE. | |
| is_I2_bndry | BOOL | | | | | TRUE if the I2_adjcy is a switching boundary . | |
| native_vlan | INTEGER | | | UNSIGNED | | native VLAN in case of trunk adjcies | |
| IndexName | | | | IndexType | Columns | | |
| PRIMARY | | | | PRIMARY | eth_link_id | | |
| fk_eth_link_link_id | | | | Index | eth_link_id | | |

vtp_domain

Represents a “switching domain” of I2_nodes that share their VLANs. This would typically be a set of interconnected switches in the same physical location, providing network access to clients. All switches in that domain are assumed to use the same VTP domain name, which should be unique in the network.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------|----------|------------|---------|-----------|---------------|---------------------------------------|---------|
| vtp_domain_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | | |
| srv_ipv4_addr | VARCHAR | | | | | IP address of the domain's VTP server | |
| IndexName | | | | IndexType | Columns | | |
| PRIMARY | | | | PRIMARY | vtp_domain_id | | |

vlan

Represents a VLAN in a vtp_domain.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-----------------------|----------|------------|-----------|---------------|---------------|---------|---------|
| vlan_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| vtp_domain_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | | | | | |
| number | INTEGER | | | UNSIGNED | | | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | vlan_id | | | |
| fk_vlan_vtp_domain_id | | | Index | vtp_domain_id | | | |

vlan_port

Represents VLAN port.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------------|----------|------------|-----------|--------------|---------------|---------|---------|
| vlan_port_id | INTEGER | PK | NN | UNSIGNED | | | |
| vlan_id | INTEGER | | NN | UNSIGNED | | | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | vlan_port_id | | | |
| fk_vlan_port_port_id | | | Index | vlan_port_id | | | |
| fk_vlan_port_vlan_id | | | Index | vlan_id | | | |

spanning_tree

Represents a spanning tree instance for VTP domains and their VLANs. There will be multiple entries in this table if the I2_adjcy is a trunk, one per VLAN exported.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------|----------|------------|---------|------------|---------------|-------------------------------------|---------|
| vlan_id | INTEGER | PK | NN | UNSIGNED | | | |
| eth_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| state | VARCHAR | | | | | ENUM('FWD', 'BLOCK', 'L3-BOUNDARY') | |
| cost | INTEGER | | | UNSIGNED 0 | | | |

| IndexName | IndexType | Columns |
|--------------------------|-----------|------------------------|
| PRIMARY | PRIMARY | vlan_id eth_link_id |
| fk_spanning_tree_vlan_id | Index | vlan_id |
| spanning_tree_FKIndex2 | Index | eth_link_id |

sdh_domain

This table stores information on the available SDH domains. In GÉANT2 there is a single SDH domain.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------------|----------|------------|---------|----------|---------------|---|---------|
| sdh_domain_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | | |
| provMethod | VARCHAR | | | | | Technology/s available for provisioning services in this domain. For GÉANT2 this will be SDH. | |
| equipment_provder | VARCHAR | | | | | This information provided to aid with stitching. For GÉANT2 this will be Alcatel. | |
| date_modified | DATETIME | | | | | | |

| IndexName | IndexType | Columns |
|-----------|-----------|---------------|
| PRIMARY | PRIMARY | sdh_domain_id |

sdh_device

This table stores information on the available SDH devices.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------|----------|------------|---------|----------|---------------|---------------------------|---------|
| sdh_device_id | INTEGER | PK | NN | UNSIGNED | | | |
| sdh_domain_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | | | | | |
| nsap | INTEGER | | | UNSIGNED | | Unique NSAP address of NE | |

| IndexName | IndexType | Columns |
|-----------------------------|-----------|---------------|
| PRIMARY | PRIMARY | sdh_device_id |
| fk_sdh_device_node_id | Index | sdh_device_id |
| fk_sdh_device_sdh_domain_id | Index | sdh_domain_id |

sdh_port

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

This table stores information on the available ports. This may include GE/10GE as well as STM-N or WDM ports.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------|----------|------------|---------|----------|---------------|---|---------|
| sdh_port_id | INTEGER | PK | NN | UNSIGNED | | | |
| address | VARCHAR | | | | | Address of port. Alcatel equipments addresses ports by the rack/shelf/card/port numbers e.g. 1-1-17-2 | |
| phy_port_type | VARCHAR | | | | | Physical port type: I-64.1, L-64.2, S-16.1, GE, 10GE VLAN, etc. | |

| IndexName | IndexType | Columns |
|---------------------|-----------|-------------|
| PRIMARY | PRIMARY | sdh_port_id |
| fk_sdh_port_port_id | Index | sdh_port_id |

stm_type

Type of STM link 0: STM-1, 1: STM-4, 2: STM-16, 3: STM-64, 4 STM-256 .

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------|----------|------------|---------|----------|---------------|---------|---------|
| stm_type_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | | |
| bandwith | FLOAT | | | | | | |

| IndexName | IndexType | Columns |
|-----------|-----------|-------------|
| PRIMARY | PRIMARY | stm_type_id |

stm_link

This table stores information on the available STM_links (i.e., links that are defined between a pair of ports on different devices).

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------|----------|------------|---------|----------|---------------|---------|---------|
| stm_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| och_id | INTEGER | | NN | UNSIGNED | | | |
| stm_type_id | INTEGER | | NN | UNSIGNED | | | |
| status | VARCHAR | | | | | | |

| IndexName | IndexType | Columns |
|-------------------------|-----------|-------------|
| PRIMARY | PRIMARY | stm_link_id |
| fk_stm_link_stm_type_id | Index | stm_type_id |
| fk_stm_link_och_id | Index | och_id |
| fk_stm_link_stm_link_id | Index | stm_link_id |

vlan_tag

This table stores information on the available VLANs.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------|----------|------------|-----------|-------------|---------------|---------|---------|
| vlan_tag_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | VARCHAR | | | | | | |
| date_modified | DATETIME | | | | | | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | vlan_tag_id | | | |

ho_vc_group

This table stores information on the available VC groups (i.e., virtual concatenation of VC links).

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------------------------|----------|------------|-----------|----------------|---------------|---------|---------|
| ho_vc_group_id | INTEGER | PK | NN | UNSIGNED | | | |
| vlan_tag_id | INTEGER | | NN | UNSIGNED | | | |
| name | VARCHAR | | | | | | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | ho_vc_group_id | | | |
| fk_ho_vc_group_vlan_tag_id | | | Index | vlan_tag_id | | | |
| fk_ho_vc_group_ho_vc_group_id | | | Index | ho_vc_group_id | | | |

ho_vc_type

This table stores information on the available HO-VC types.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|---------------|----------|------------|-----------|---------------|---------------|--|---------|
| ho_vc_type_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| name | INTEGER | | | UNSIGNED | | 0: VC-3, 1: VC-4, 2: VC-4-4c, 3: VC-4-16c, 4: VC-4-64c, 5: VC-4-256c | |
| bandwidth | FLOAT | | | | | bandwidth of the VC type (in Mbits/sec) | |
| payload | FLOAT | | | | | bandwidth with no overhead (in Mbits/sec) | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | ho_vc_type_id | | | |

ho_vc_link

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

This table stores information on the available HO-VC_links (VC-3/4 links that are configured on the available STM_links).

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------|----------|------------|---------|----------|---------------|---|---------|
| ho_vc_link_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| stm_link_id | INTEGER | | NN | UNSIGNED | | | |
| ho_vc_group_id | INTEGER | | NN | UNSIGNED | | | |
| ho_vc_type_id | INTEGER | | NN | UNSIGNED | | | |
| time_slot | INTEGER | | | UNSIGNED | | | |
| date_modified | DATETIME | | | | | | |
| group_sequence | INTEGER | | | UNSIGNED | | Indicates the sequence number of the VC in the group | |
| status | VARCHAR | | | | | Status of link: operational, not operational, unknown | |

| IndexName | IndexType | Columns |
|------------------------------|-----------|----------------|
| PRIMARY | PRIMARY | ho_vc_link_id |
| fk_ho_vc_link_ho_vc_type_id | Index | ho_vc_type_id |
| fk_ho_vc_link_ho_vc_group_id | Index | ho_vc_group_id |
| ho_vc_link_FKIndex3 | Index | stm_link_id |

och

This table stores information on the available optical channels. Note that this channel inherits the start and end ports from the OCh links that make up the channel.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|------------|----------|------------|---------|----------|---------------|---|---------|
| och_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| payload | VARCHAR | | | | | Format of the OCh payload | |
| status | VARCHAR | | | | | Status of path: operational, not operational, unknown | |

| IndexName | IndexType | Columns |
|-----------|-----------|---------|
| PRIMARY | PRIMARY | och_id |

ops_link

This table allows for future expansion to WDM interfaces. The OPS stores information on the available Optical Physical Section links (i.e., links that are defined between a pair of ports on different OTN devices) Note that the OMS and OTS are included in the OPS.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|----------------|----------|------------|-----------|-------------|---------------|--|---------|
| ops_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| max_no_lambdas | INTEGER | | | UNSIGNED | | Contains the number of wavelength on this optical section if the section is WDM (equivalent to n in OTM-n.m) | |
| bitrate | INTEGER | | | UNSIGNED | | OTM bit rate (equivalent to m in OTM-n.m) | |
| status | VARCHAR | | | | | This table stores information on the available optical channels. Note that this channel inherits the start and end ports from the optical sections that make up the channel. | |
| IndexName | | | IndexType | Columns | | | |
| PRIMARY | | | PRIMARY | ops_link_id | | | |
| fk_ops_ops_id | | | Index | ops_link_id | | | |

och_link

This table stores information on the available optical channels. Note that this channel inherits the start and end ports from the optical sections that make up the channel.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------|----------|------------|-----------|----------|---------------|---|---------|
| och_link_id | INTEGER | PK | NN | UNSIGNED | | | AI |
| ops_link_id | INTEGER | | NN | UNSIGNED | | | |
| och_id | INTEGER | | NN | UNSIGNED | | | |
| frequency | FLOAT | | | | | Contains the frequency (wavelength) of the channel (where the section is WDM) | |
| status | VARCHAR | | | | | Status of path: operational, not operational, unknown | |
| IndexName | | | IndexType | Columns | | | |

| | | |
|--------------------|---------|-------------|
| PRIMARY | PRIMARY | och_link_id |
| fk_och_link_och_id | Index | och_id |
| och_link_FKIndex2 | Index | ops_link_id |

generic_link_generic_link

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--|-----------|------------|---------------------------------|----------|---------------|---------|---------|
| parent_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| child_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| IndexName | IndexType | | Columns | | | | |
| PRIMARY | PRIMARY | | parent_link_id child_link_id | | | | |
| generic_link_has_generic_link_FKIndex1 | Index | | parent_link_id | | | | |
| generic_link_has_generic_link_FKIndex2 | Index | | child_link_id | | | | |

monitored_link_generic_link

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|--|-----------|------------|--------------------------------------|----------|---------------|---------|---------|
| monitored_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| generic_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| IndexName | IndexType | | Columns | | | | |
| PRIMARY | PRIMARY | | monitored_link_id generic_link_id | | | | |
| monitored_link_has_generic_link_FKIndex1 | Index | | monitored_link_id | | | | |
| monitored_link_has_generic_link_FKIndex2 | Index | | generic_link_id | | | | |

JRA3_interface_ATM

Layer 2 interface extension for ATM.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-----------------------|-----------|------------|---------|----------|---------------|-------------------------------------|---------|
| JRA3_interface_ATM_id | INTEGER | PK | NN | UNSIGNED | | | |
| VCPI_local | VARCHAR | | | | | virtual path identifier (local) | |
| VCPI_remote | VARCHAR | | | | | virtual path identifier (remote) | |
| VCI_local | VARCHAR | | | | | virtual channel identifier (local) | |
| VCI_remote | VARCHAR | | | | | virtual channel identifier (remote) | |
| IndexName | IndexType | | Columns | | | | |

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

| | | |
|------------------------------------|---------|-----------------------|
| PRIMARY | PRIMARY | JRA3_interface_ATM_id |
| FK_JRA3_interface_ATM_interface_id | Index | JRA3_interface_ATM_id |

JRA3_interface_framerelay

Layer 2 interface extension for frame relay.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|------------------------------|----------|------------|---------|----------|---------------|--|---------|
| JRA3_interface_framerelay_id | INTEGER | PK | NN | UNSIGNED | | | |
| NUA | VARCHAR | | | | | network user address | |
| DCLI_local | VARCHAR | | | | | data link connection identifier (local) | |
| DCLI_remote | VARCHAR | | | | | data link connection identifier (remote) | |

| IndexName | IndexType | Columns |
|--|-----------|------------------------------|
| PRIMARY | PRIMARY | JRA3_interface_framerelay_id |
| FK_JRA3_interface_ramerelay_interface_id | Index | JRA3_interface_framerelay_id |

topology_point

A TopologyPoint is a logical representation of either an end point of an E2E link or a demarcation point. This table also keeps data of topology points of neighbour domains. It allows to model connections between domains.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|-------------------|----------|------------|---------|----------|---------------|---------|---------|
| topology_point_id | VARCHAR | PK | NN | | | | |
| location_id | INTEGER | | | UNSIGNED | | | |
| domain_id | VARCHAR | | | | | | |

| IndexName | IndexType | Columns |
|-------------------------------|-----------|-------------------|
| PRIMARY | PRIMARY | topology_point_id |
| fk_topology_point_location_id | Index | location_id |

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

monitored_link

A MonitoredLink is delimited by two TopologyPoints.

| ColumnName | DataType | PrimaryKey | NotNull | Flags | Default Value | Comment | AutoInc |
|------------------------------|------------------|------------|-------------------------|----------|---------------|---|---------|
| monitored_link_id | INTEGER | PK | NN | UNSIGNED | | | |
| version_info_id | INTEGER | | NN | UNSIGNED | | | |
| end_topology_point_id | VARCHAR | | NN | | | | |
| start_topology_point_id | VARCHAR | | NN | | | | |
| e2e_link_id | VARCHAR | | | | | This field contains the "globally unique" E2E Link ID. This information is needed to correlate the Monitored Links which belong to the same E2E Link. | |
| local_name | VARCHAR | | | | | | |
| start_topology_point_role | VARCHAR | | | | | Role of the TopologyPoint (either Demarcation Point or Endpoint) | |
| end_topology_point_role | VARCHAR | | | | | Role of the TopologyPoint (either Demarcation Point or Endpoint) | |
| link_type | VARCHAR | | | | | This field can be one of DomainLink, IDLink (for an Interdomain Link) and PartIDLink (for an Interdomain Link Partial Info) | |
| IndexName | IndexType | | Columns | | | | |
| PRIMARY | PRIMARY | | monitored_link_id | | | | |
| fk_monitored_link_version_id | Index | | version_info_id | | | | |
| monitored_link_FKIndex2 | Index | | start_topology_point_id | | | | |
| monitored_link_FKIndex3 | Index | | end_topology_point_id | | | | |

3 Conclusions and Future Work

cNIS has been developed with a very specific purpose in mind, namely to provide a single network topology database for use by the new GN2-developed applications, such as AMPS and perfSONAR. Before the conception of cNIS, each of these GN2 applications had plans to collect, store and maintain their own individual topology databases, and this would represent a waste of resources, and an unnecessary burden on network operations staff who would have to administer each of the database modules independently.

Since the goal of cNIS is to serve GN2 applications, its designers and developers have worked closely with the developers of the other GN2 activities to ensure that cNIS is able to hold all the data required by those applications. In the case of IP topology data, cNIS will use network discovery techniques in order to populate the appropriate tables – for other, lower –layer protocols such as Ethernet and SDH such auto-discovery is not possible, but as and when methods are developed that do allow the automated collection of this information then cNIS will be modified appropriately.

Whilst it is hoped that in time there will be a common interface through which any authorised application accesses cNIS’ topology data, for now cNIS will honour specific, bespoke web service interfaces for each of the currently planned GN2 applications (AMPS, AutoBAHN and the perfSONAR compliant monitoring tools). The exact format of these interfaces, and their operation, is the next part of cNIS’ development. The design of these interfaces will be critical to the overall success of the new GN2 services, since it is perfectly possible that some NRENs may already use cNIS-like topology databases, in which case rather than deploying the cNIS reference model those legacy databases can be adapted to expose the appropriate interfaces.

The original concept of cNIS was to only hold data common across multiple applications. However, the current scope also includes holding client-specific information on request. Future effort for the remainder of GN2 will investigate extensions to the scope of the cNIS to possibly include the storage, retrieval and analysis of real-time networking data, advanced visualisation of database contents and storage

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

4 References

| | |
|----------------------------|--|
| [GGF NM-WG] | http://nmwg.internet2.edu/ |
| [SA3-AMPS] | GN2-04-153v4 AMPS – Design Specification.doc |
| [SA3-PERT] | GN2-06-016v5: DS3.7.3 Report on Performance Monitoring Tools |
| [JRA1-perfSONAR] | GN2-04-150v14 (DJ1.1.1): Requirements Report on the Design of the Measurement System |
| [JRA3-AutoBAHN] | GN2-05-208v7 (DJ3.3.1): Definition of Bandwidth on Demand Framework and General Architecture |
| [JRA4-E2E] | E2E Monitoring System Design, accessed from http://wiki.geant2.net/pub/JRA4/DmsDocE2ELinkDataModel/GN2-JRA4-06-010v121.pdf |
| [JRA1-CNM] | http://wiki.perfsonar.net/jra1-wiki/index.php/CNM |
| [JRA3-EthSchema] | http://wiki.geant2.net/pub/JRA3/JRA3TopologyDB/EthernetTopologyDBschema.doc |
| [JRA3-SDHSchema] | http://wiki.geant2.net/pub/JRA3/JRA3TopologyDB/sdh_schema_20060224_v5.doc |
| [SA3-JRA1-IPSchema] | http://wiki.geant2.net/bin/view/SA3/Cnis_Schema |
| [JRA4-Schema] | http://wiki.geant2.net/pub/JRA4/DmsDocE2ELinkDataModel/GN2-JRA4-06-010v121.pdf |

5 Acronyms

| | |
|-----------------|--|
| AMPS | Advance Multi-domain Provisioning System |
| AutoBAHN | Automated Bandwidth Allocation across Heterogeneous Networks |
| CBF | Cross-Border Fibre |
| cMA | cNIS Management Application |
| cNIS | Common Network Information Service |
| CNM | Customer Network Management |
| DBMS | Database Management System |
| DM | Domain Manager |
| E2E | End-to-End |
| E2EMON | End-to-End Monitoring System |
| ER | Entity Relationship |
| GGF | Global Grid Forum |
| IDM | Inter Domain Manager |
| IP | Internet Protocol |
| JRA | Joint Research Activity |
| LAN | Local Area Network |
| LHC | Large Hadron Collider |
| MA | Measurement Archive |
| MP | Measurement Point |
| MPLS | Multi-Protocol Label Switching |
| NMWG | Network Monitoring Working Group |
| NOC | Network Operations Centre |
| NREN | National Research and Education Network |
| OCH | Optical Channel |
| OPN | Optical Private Network |
| OTN | Optical Transport Network |
| PERT | Performance Enhancement and Response Team |
| POP | Point of Presence |
| QoS | Quality of Service |
| SDH | Synchronous Digital Hierarchy |
| SNMP | Simple Network Management Protocol |
| SSH | Secure Shell |
| UI | User Interface |

| | |
|-------------|----------------------------------|
| VCG | Virtual Concatenation Group |
| VLAN | Virtual LAN |
| WDM | Wavelength-division multiplexing |
| WI | Work Item |
| XML | Extensible Markup Language |

Appendix A Entity Relationship (ER) Diagram

The Entity Relationship diagram (ER diagram) completes the formal description of the cNIS schema and constitutes a final part of the conceptual data modelling phase. It presents data entities and their relationships in a graphical notation.

The ER model can be transformed into the relational model, which presents a logical schema of the relational database.

Due to size of the figure presenting the cNIS schema, it has been split into several parts for better understanding. Each part refers to the specific functionality of the database model (see section 1.5)

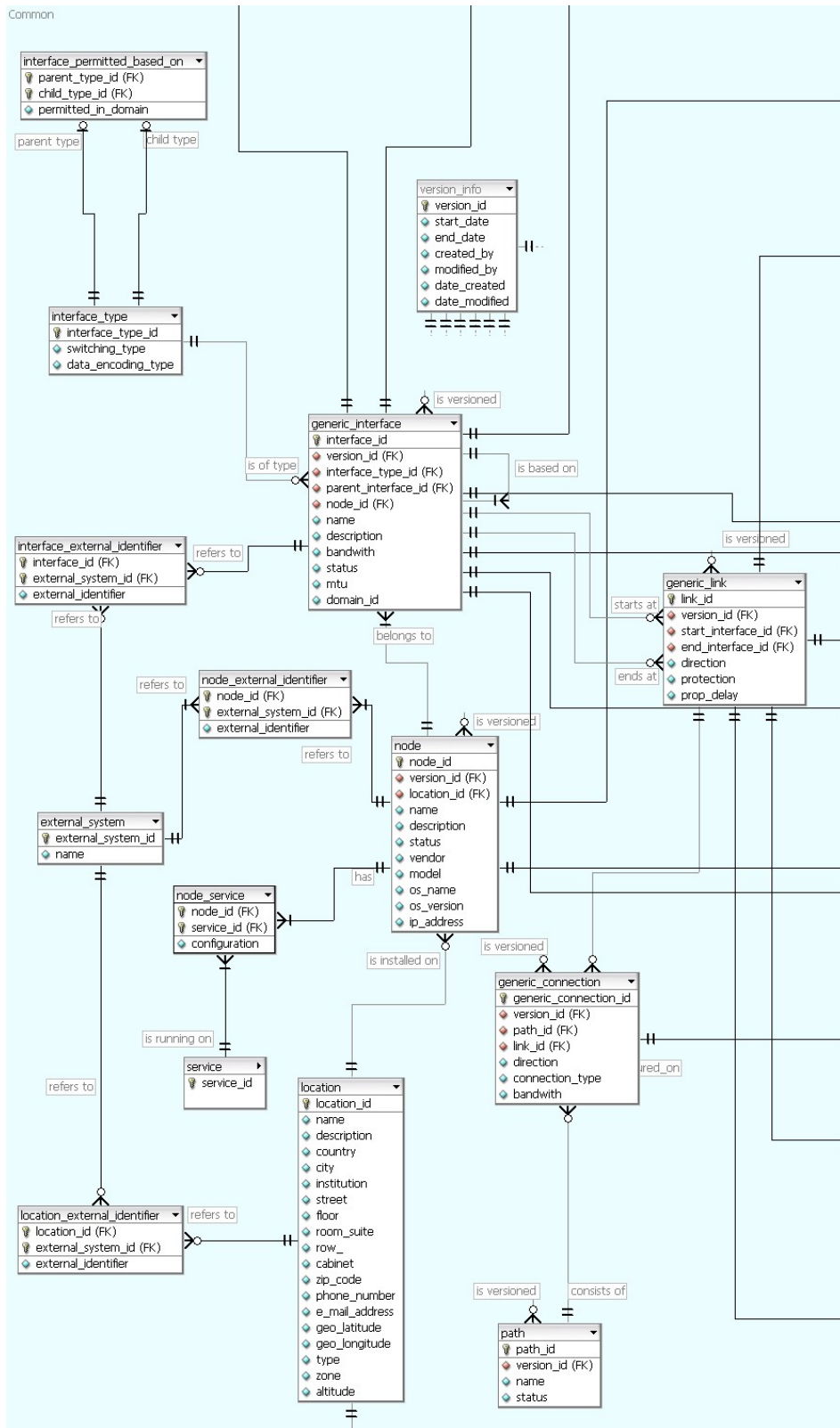


Figure 6.ER diagram - common tables

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

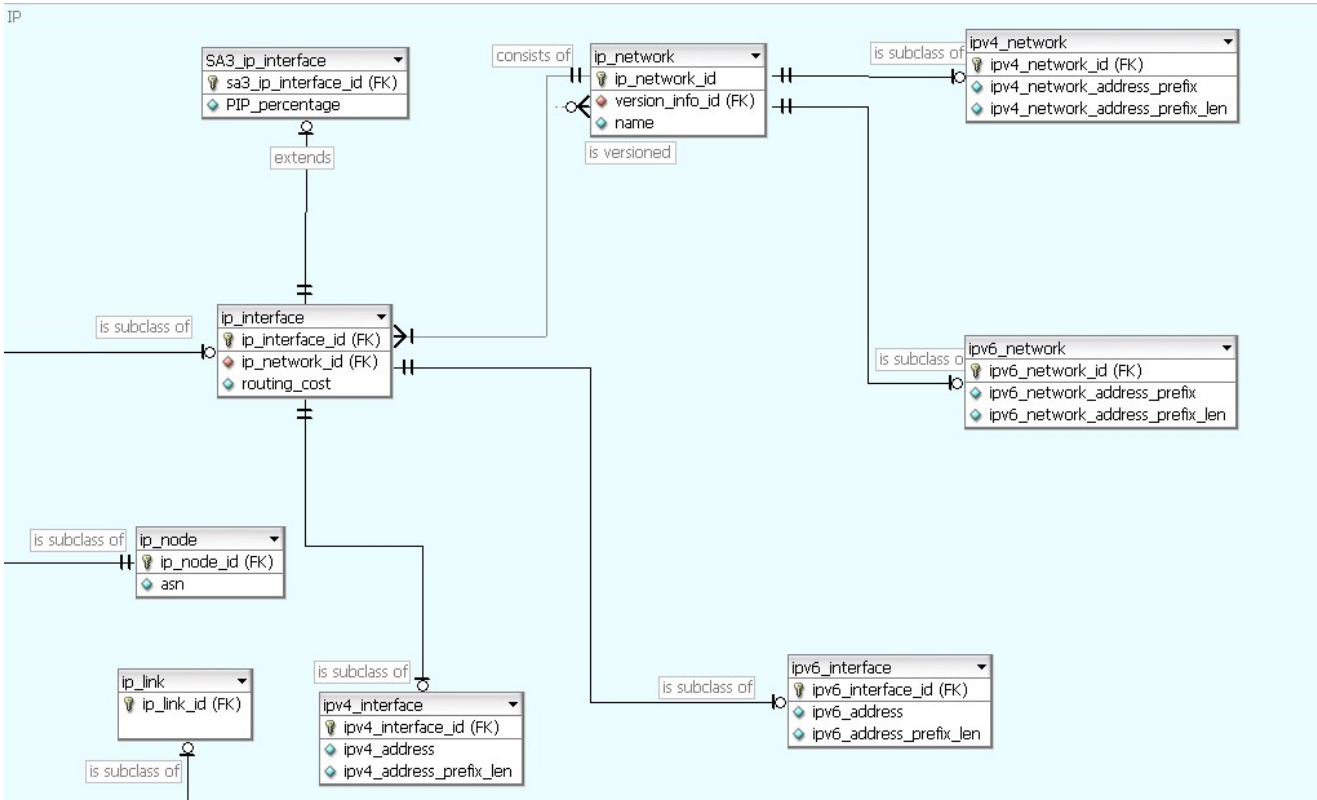


Figure 7.ER Diagram - IP tables

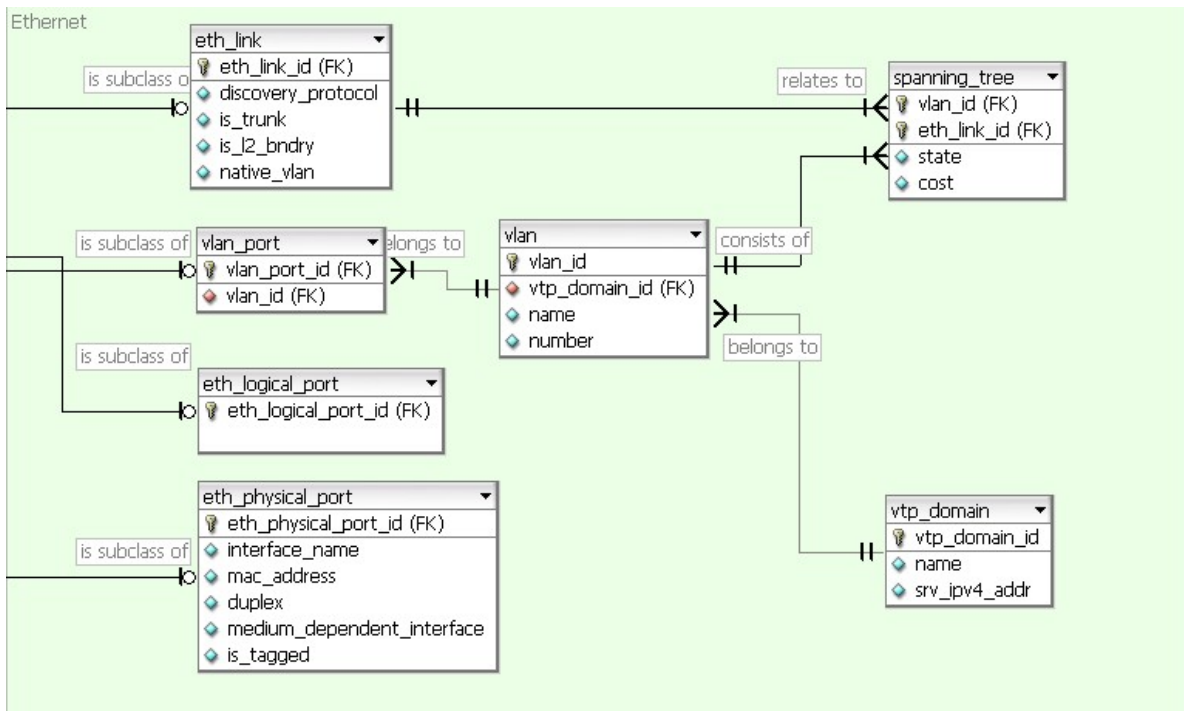


Figure 8.ER diagram - Ethernet tables

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

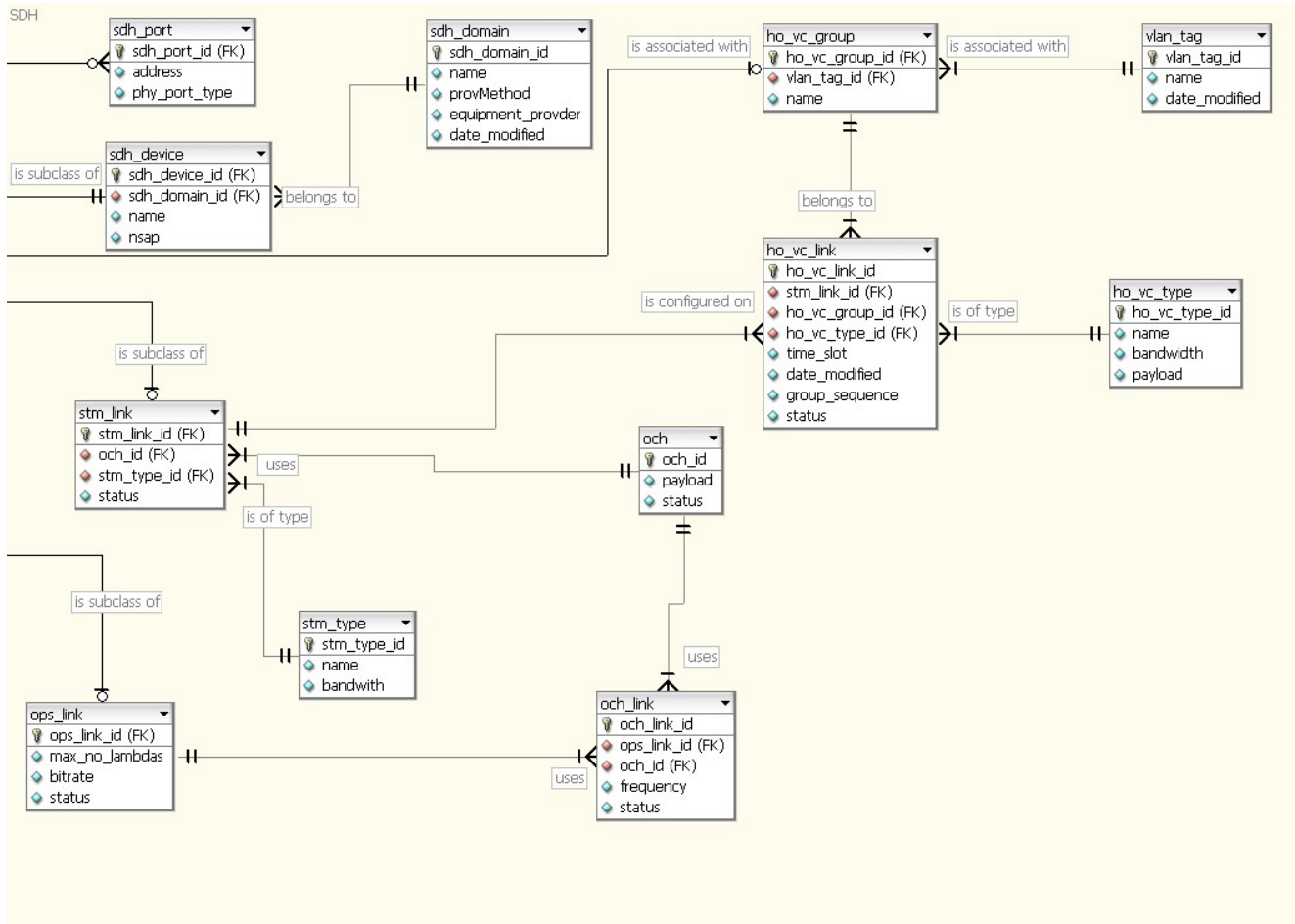


Figure 9.ER diagram - SDH tables

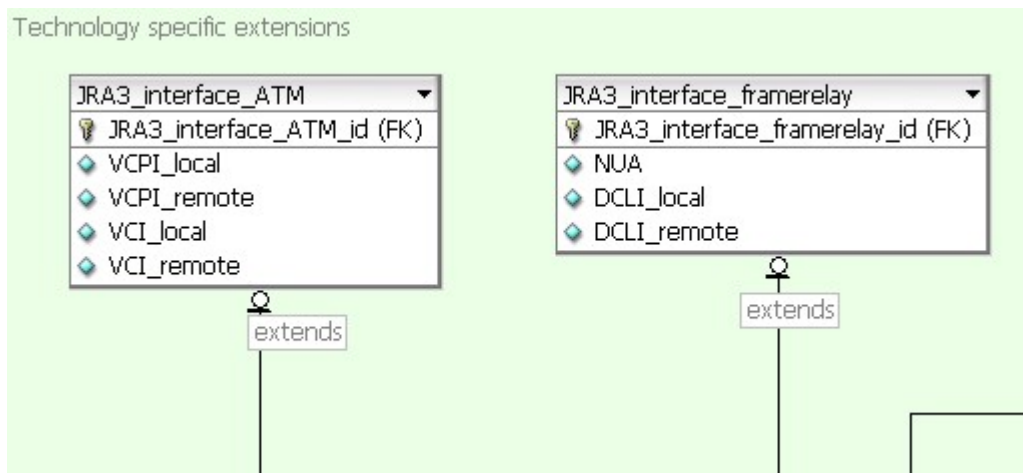


Figure 10.ER diagram - technology specific tables

| | |
|---------------------|--------------|
| Project: | GN2 |
| Deliverable Number: | DS3.13.1 |
| Date of Issue: | 24/04/07 |
| EC Contract No.: | 511082 |
| Document Code: | GN2-07-045v4 |

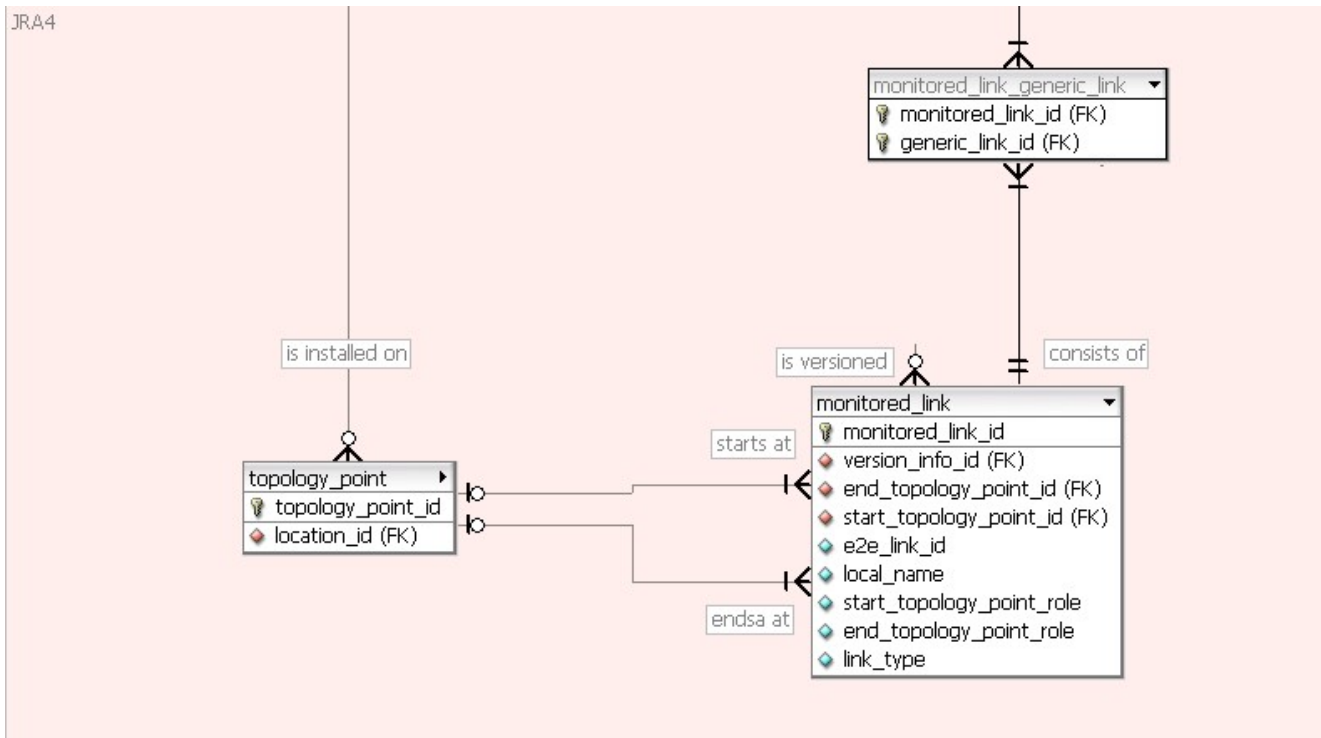


Figure 11.ER diagram - JRA4 specific tables