NATview
User Manual

Version 3.01

Table of Contents

Table of Figures

List of Tables

Document History

Revision	Date	Author	Description
1.0	29.02.2008	Stefan Sperling	Initial version from requirements documentation.
1.1	03.03.2008	Stefan Sperling	Screenshot reworking.
1.2	11.03.2008	Stefan Sperling	Update Resource Scanning
1.3	26.05.2008	Stefan Sperling	Reworking of the following chapters: Rack Pane (4.4), Fru Menu (was “Utilities”, 5). Added chapter about customer hardware support (chapter 0).
1.4	30.06.2008	Stefan Sperling	Updated lots of screenshots. Rewrote chapter 3.4 (“Configuring NATView”). New chapter 5 (“View menu”). New chapter 7 (“Release history”).
1.5	14./19.11.2008	Stefan Sperling	Reworked the complete manual to reflect the changes of NATView 1.21; a lot of minor changes to reflect changes of technical terms.
1.6	09.02.2009	Stefan Sperling	Updated revision history until including NATView release 1.25.
1.7	11.08.2009	Stefan Sperling	Update the complete manual to reflect NATView version 1.30.
1.8	11.02.2010	Stefan Sperling	Update the configuration keywords with the new ones from NATView 1.35. Update the feature list for releases 1.31 to 1.35.
1.9	25.03.2010	Stefan Sperling	Reworking of the complete document to include all changes of NATView 1.37.
1.10	28.06.2010	Stefan Sperling	New chapter 2.4 about license key installation.
1.11	28.10.2010 04.11.2010	Stefan Sperling Stefan Sperling	Converted to DOCX-Format. Changed all occurrences to NATview. Updated document for NATview release 1.39.

Revision	Date	Author	Description
1.12	12.11.2010	Stefan Sperling	Updated license installation chapter.
1.13	16.02.2011	Stefan Sperling	New chapter about the new backplane connectivity viewer.
1.14	01.06.2011	Stefan Sperling	This document version corresponds to NATview 2.2. New chapters about the MCH scanner, System Dump and some small adaptions caused by modified menu structure.
1.15	27.07.2011	Stefan Sperling	Added chapter 6.5.2 that describes the tool ExtractFru.
1.16	26.03.2012	Stefan Sperling	Added chapters about the SDR history viewer and the SDR auto update functionality. Various tiny updates.
1.17	11.07.2012	Stefan Sperling	New features for NATview 2.5, especially the support for the NAMC-LM.
1.18	-	Stefan Sperling	New features for NATview 2.6.
1.19	29.04.2013	Stefan Sperling	New features for NATview 2.7: * HPM update * Trail license key request * Chapter 10.1: added dimensions for additional form factors.
1.20	03.12.2013	Stefan Sperling	New features for NATview 2.9: * Backwards Compatibility Mode Checker * Updated features for NATview 2.9: * HPM-Update Changes for NATview 2.9: * various bug fixes
1.21	30.06.2014	Stefan Sperling	Reworking of almost the complete document to reflect the changes of NATview 2.13.
1.22	01.07.2014	Stefan Sperling	Added missing documentation of the extended chassis detection in NATview 2.13.
1.23	31.07.2014	Stefan Sperling	Added/modified description of the Power Configuration Manager (PCM, formerly known as Power Manager).

Revision	Date	Author	Description
1.24	12.11.2014	Stefan Sperling	Added description of the Carrier/Shelf Manager SDR viewer. Also added a chapter about the Fru File Cutter. Several screenshots have been updated to reflect the modified user interface.
1.25	14.04.2015	Stefan Sperling	Positioning of main window: Now checks it the resulting window is visible under the configuration that is currently used. If not the coordinates are corrected. Up to now it was possible that the windows were invisible because a NATview installation was being used with a two-monitor-setup before. AmcP2pLinkInfo / Backplane Connectivity Viewer: Fixed handling of descriptor channel ID; now matches what was intended by the AMC spec. Can now handle more that one AMC P2P connectivity records per FRU. HPM: Fixed handling of extended IPMI_CMD_UPLOAD_FIRMWARE_BLOCK_RSP with new offset/length; missing functionality could lead to breakups when updating. also added Skip-Functionality for HPMs UploadFirmwareBlock. Now better use of debug source HPM so user can turn on/off HPM debugs more precisely. Changed event handling for hot swap M4->M1 which could lead to a strange behavior seen on some boards on hot swap. FruEditor.java: Support for new date/time editor. Support/fix for data type GUID. Several tiny fixes, e.g. data mode defintions and descriptions. Support/workarounds for buggy backplane FRUs. Always enable the system dump when there is an active connection.
1.26	19.08.2015	Stefan Sperling	Support for Schroff's carrier coordinate system.
1.27	16.10.2015	Stefan Sperling	Lots of bug fixes, no new functionality.
1.28	04.12.2015	Stefan Sperling	Reworked&extended the complete manual, especially chapters 5.3.3, 5.3.5, 6.2, and 7.4.
2.24	10.06.2016	Stefan Sperling	Increased document version to match the software version. Added documentation for all features and bugfixes for

Revision	Date	Author	Description
			NATview releases 2.22, 2.23, and 2.24, especially the extended AMC detection.
2.25	18.08.2016	Stefan Sperling	Added chapter 13.3 in the trouble shooting section that describes the correct hot swap sensor type settings.
3.00	23.01.2017	Stefan Sperling	New license key. Slot Orientations in the Chassis INI File. Several screenshot updates.
	26.05.2017	Stefan Sperling	Reworked the new chapter Solutions: PowerConfiguration: added the tiny section from 5.3.6.1 "Starting the Power Configuration Manager". This shall it make easier for the production department to use the PCM for specific power configurations.
	14.03.2018	Stefan Sperling	Completely revisited the manual for software release 3.00.
3.01	18.01.2018	Stefan Sperling	Added section 5.3.2 “The FRU editor toolbar”.

Glossar

a.k.a.	also known as
AMC	Advanced Mezzanine Card
BIA	Board Info Area, a part of the *FRU* info record.
EAD	Extended *AMC* Detection
FRU	Field replaceable unit, usually an *AMC* board
GUI	Graphical User Interface
HPI	Hardware Platform Interface
HPM	*Hardware Platform +Management* currently defined in the *PCIMG* specifications HPM.1 to HPM.4.
IPMI	Intelligent Platform Management Interface
Java	Programming language originally developed by *Sun* Microsystems, now owned by *Oracle*.
Linux	UNIX-like, open source operating system
Mac OS X	A line of graphical operating systems developed, marketed, and sold by Apple Inc. *Mac OS X* had been the original name. It changed gradually to *OS X* and in 2016 to *Mac OS*. All of these names shall be used inside this document interchangeably.
MCH	*MicroTCA* Carrier Hub
Microsoft	a.k.a. Microsoft Corporation.
MicroTCA	Micro Telecommunications Computing Architecture
OpenHPI	Open source implementation of *HPI*
Oracle	a.k.a. Oracle Corporaction.
PCI	Peripheral Computer Interconnect
PIA	Product Info Area, a section of the *FRU* info record.
PICMG	PCI Industrial Computer Manufacturers Group
PM	Power Module (also see *PU*).
PU	Power Unit (also see *PM*)

RMCP	Remote Management Control Protocol, described in the *IPMI* specification 1.5 and 2.0 (among others).
SANI	Sensor Alarm Notification Indication
SDR	Sensor Data Record: describes the capabilities of the *IPMI* sensors in a *MicroTCA* device.
Sun	Sun Microsystems. Now owned by *Oracle*.
Windows	Operating System developed by *Microsoft*.

References for this document

1. “Intelligent Platform Management Interface Specification”,
Version 1.5,
Document Revision 1.1, 20-Feb-2002

2. “Intelligent Platform Management Interface Specification”,
Version 1.5,
Document Revision 1.0, 12-Feb-2004

3. "Hardware Platform Management IPM Controller Firmware Upgrade Specification",
Revision 1.0,
04-April-2007

1 Introduction

NATview can be used to scan, monitor and change the various resources of a MicroTCA system. The software has been developed and tested with the N.A.T. MicroTCA Carrier Hub (NMCH). It is assumed to work with MCHs from other manufacturers as well as long as they are at least IPMI 1.5 compliant.

1.1 What you should read in this Manual

Even if you hate manuals: Please take a quick look at the sections listed in the following table – it might save you a lot of time and hassle:

Chapter		Content
Number	**Name**	
1.2	What is new in this release?	Lists all new or changed features in this release (compared to the last one). Start here if you are already a NATview user.
2	Getting Started	Lists the system requirements and describes how to start the application.
13	FAQs and Troubleshooting	Provides solutions for the most common user problems.

1.2 What is new in this release?

This section shall facilitate the work with this manual for all those who are already NATview users. Check out all listed chapters and you should be done.

Chapter		Content
Number	**Name**	
1.2	Installing the NATview license key	Running NATview 3 requires a new license key. Customers that already have a key should contact N.A.T. to get their renewal.
12.1	Recent Changes (NATview 3.x)	Fundamental changes with the image management in NATview 3.x.
12.2.1.2.3	Slot Orientations in the Chassis INI File	Extensions of the chassis INI files for non-standard MicroTCA chassis.
13.4	The Chassis is ok but the AMC boards orientation is wrong	Update of the FAQ list.

1.3 Design Goals

Before starting the NATview development the following list of goals was made:

1.3.1 Platform independence

NATview will run on every system that is capable to execute the Oracle Java Runtime Environment (JRE) **1.8 or higher**.[1] It should therefore be possible to use it under Windows, Linux and Mac OS X. We have successfully tested NATview with these 32-bit operating systems (64-bit only where mentioned):

- Windows XP Professional[2]
- Windows 7 (also 64-bit)
- Windows 10[3]
- Debian Linux Sarge and Etch
- Open SuSE Linux 10 and 11
- Ubuntu Linux 6.06 – 16.04
- Mac OS X 10.5.x (Leopard), 10.6.x (Snow Leopard), and 10.7 – 10.12 (Lion, Mountain Lion, Mavericks, Yosemite, El Capitan, Sierra and, Hight Sierra – requires that Oracle Java was being installed by the user).

NATview should run as well under Windows 98 SE, Windows ME, and all other Linux derivates that run Oracle Java not already mentioned.

This documentation previously stated that NATview would not run using **GNU Java**. It appears that this is no longer true as this Java flavour has dramatically increased its power. Try at your own risk.

1.3.2 No installation required

NATview can be used without any installation. The worst thing that can happen if being executed from a read-only media is that the configuration cannot be saved. The same is true if it is executed from a root-owned directory with user privileges. Simply make `natview.ini` writeable and everything will work.

[1] The first version of NATview was written using Oracle Java 1.6. Meanwhile we upgraded the code using Oracle Java 1.8 to take advantage of the security patches that Oracle provides..

[2] Not recommended.

[3] Requires manual installation of Oracle Java 1.7 or 1.8; we recommend using Java 1.8.

1.3.3 One unregistered (“easy”) and four registered (“license”) Versions

The basic functionality always will be available for free while the more sophisticated functions are reserved for the license versions, which are NATview FE, NATview HPM, NATview PRO, and NATview PROC. Consult section 2.4.3 for more details.

1.3.4 Easy extensible

Although NATview supports a lot of chassis and AMC modules out-of-the-box there will always be hardware that is initially unsupported. This is no problem: the NATview user can easily manage new hardware by supplying a hardware graphic file plus a hardware ini-file – that’s all it takes. Details about this procedure are covered later in this documentation (see chapter 12 for details).

1.4 About this Document Version

This version of the NATview documentation has been completely reworked to cover the new features that came with NATview version 3.00. It can be used with older NATview releases as well. Features from older NATview version that have been removed or changed in a major fashion will only be described like they are implemented in the current software release.

For the first time the document release correlates with the NATview software version. This shall facilitate finding the correct NATview documentation release.

2 Getting Started

2.1 System requirements

NATview requires Oracle Java 1.8[4] – the never, the better. We chose to use this Oracle Java version to guarantee the largest possible JRE installation basis on Window, Linux and Mac OS X systems.

If in doubt enter

```
java –version
```

at a command prompt window. An appropriate copy of the Sun JRE can be obtained at **`java.sun.com`**. Carefully read the installation instructions that come with the delivery package.

2.2 Installing NATview

NATview is usually distributed as a zip archive. There is no special installation program – simply extract the content of the zip archive to an empty directory. This directory will be the **root directory** of the application.

2.3 Starting NATview

There are two ways to start NATview:

1. Start the application **from within the root directory**. (Otherwise the application cannot find the board images in the subdirectory *images*.)
   From the command line type

   ```
   cd <root-directory>
   java –jar natview.jar
   ```

2. Use the application starter for your platform: `natview.exe` for Windows and `natview.sh` for Linux systems. These starters ensure that the NATview application is running from within the correct directory environment.

   Remark for Windows user: It is currently not possible to start a Windows application from the command line using an UNC path. This behaviour is caused by cmd.exe, which is being used

[4] Also called Java 7. The version naming conventions are not very consistent but this is not our fault.

by the starter application. A workaround for this would be to use a drive letter that is being attached to a SMB share. It is also possible that the Windows starter will fail with other Windows versions than *Windows 7* as the starter is not officially supported.

2.4 Installing the NATview license key

Note: If you have purchased a NATview license key for NATview 1.xx or 2.xx you need to renew your license key. Simply send us an email with your old key and it will be updated.

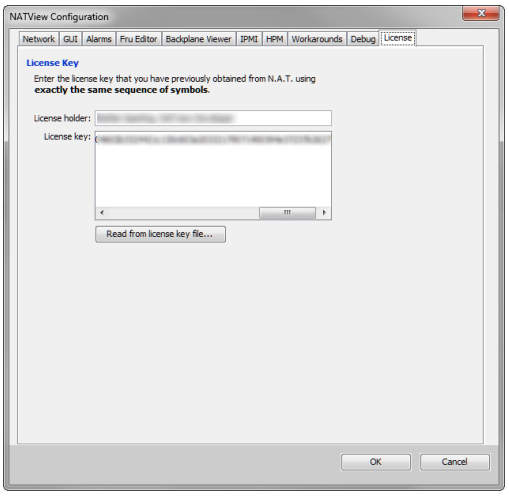
If you have purchased a NATview license key from N.A.T. you can convert your ***NATview Easy*** into a ***NATview Professional***. Starting NATview 1.40 there are two ways how to install your key, depending what you have received from N.A.T:

- If you have received an email with an attached license key file (named *natview.key*) then read section 2.4.1
- If you have received a plain-text email with an embedded license key then continue with section 2.4.2

2.4.1 Installing a license key file

If you have received your license key as an attached email file then you need to do the following:

1. Save the attached license key file (usually called *natview.key*) to your filesystem. Remember the directory path to the file.
2. Start NATview. Open the configuration dialog and select the “License” register. You should see something like this:



3. Enter the license holder name as being told by N.A.T.

4. Select the “Read from license file” button. Go to the directory where you had stored the license file. Select the license file, and then click on “Open”[5].

If everything was successful then your new license key is being displayed and being written to the NATview INI-File. Your NATview has been successfully licensed.

Note: If all else fails you can open the license file with a text editor (not Word!) and copy the key from it like it is described in the next section.

2.4.2 Installing a license key

If you have received your license key as a plain-text email you need to follow these steps to install the key:

1. Open the document that contains your license key.

   This will usually be an email containing your license key and your license name. The license name is what is displayed in the about dialog of the application. The license key is an encrypted data string that contains the following information:

   * License name
   * Version range of NATview that can be used with the license key
   * Expiry date of the license

   It looks like this:
   `032837242611146e050c0a19007408b73f1d062e2115176e7d585342406472777e4651425e6473717d584c425f6473717d59535d0e66`

2. Start NATview. Open the **configuration dialog** (Application->Configuration).
   Select the register License. A window similar to the following one should open. (If you are using NATview Easy both text fields are empty.)

3. Select the **license name** from your document and copy it onto the clipboard. Go to the license dialog, click into the text field “License holder” and type Ctrl-V to paste the data.

4. Select the **license key** from your document and copy it onto the clipboard. Go to the license dialog, click into the text field “License key” and type Ctrl-V to paste the data.

5. Close the license dialog: Click on the OK button.

This should enable your *NATview Professional*. If not close the application and reopen it.

[5] **Called “Öffnen” in german.**

2.4.3 Checking that your licensed features are enabled

Starting with NATview 2.13 the extended features of NATview can be licensed separately. Whoever needs the FRU editor only needs to pay for this feature without having to pay the unneeded HPM update feature. You only pay for those extended features that you really want!

There is no longer a single NATview Professional – but four:

1. **NATview FE** that contains the FRU editor and the Power Configuration Manager.
2. **NATview HPM** that features the HPM update feature.
3. **NATview PRO** with five personalized licenses; includes the options NATview FE and NATview HPM.
4. **NATview PROC** which is a corporate license for unlimited users; includes the options NATview FE and NATview HPM.

To check that all the features you paid for are really activated simply open the application version info box with the menu command `Help->About`. A dialog window similar to this will open (software release and some other details might be different with your release):

All activated options are marked with a green dot where all non-licensed inactive features carry a dark grey dot. When no options are licensed and there is no license holder then you are using NATview Easy.

3 Scanning a MicroTCA system

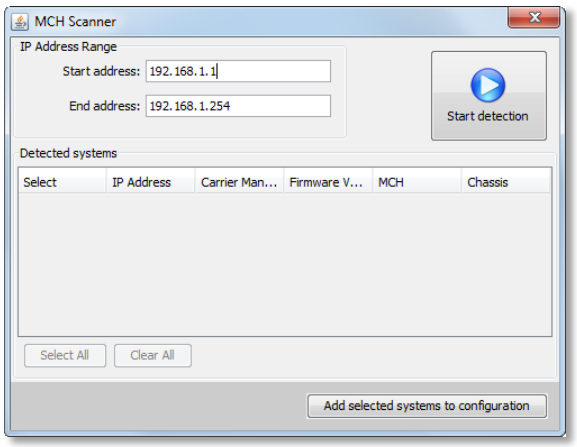
3.1 Supported NMCH firmware releases

NATview (starting with version 1.30) requires NMCH firmware release 2.5 or higher. It should work with all other IPMI-compliant MCH firmware that implements a separate shelf- and carrier-manager.

If the NMCH used is running firmware release 2.4k (or an earlier version) you need to use NATview 1.25. You may as well update your MCH firmware to the current release – please contact N.A.T. for further details.

3.2 The MCH Scanner: find all MicroTCA systems in your network

The MCH scanner is used to scan the network for all MicroTCA systems with a N.A.T. MCH. It can be started using the command **Application->MCH Scanner** of the menu. Its user interface looks like this:



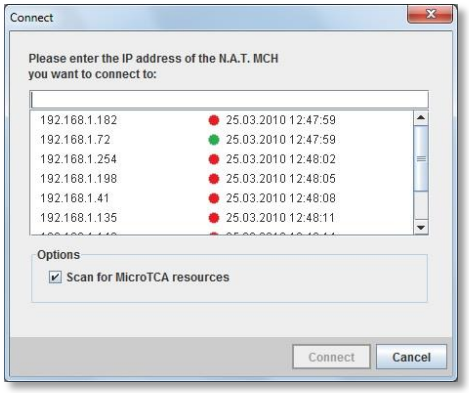
The tool scans a range of IPv4 addresses for MicroTCA systems. Use it like this:

1. Enter the start and the end IPv4 address into the text boxes.
2. Click on the “Start detection” button. The network scan starts – wait for it to finish.
3. For every MicroTCA system it finds its IP address, the carrier manager manufacturer- and product-ID, the manufacturer- and product name, its firmware version and the chassis manufacturer- and product name are listed. Check the “Select” checkbox for every system’s IP address that should be added to the configuration.

4. Import the selected IP addresses by clicking the “Add selected systems to configuration” button.
5. The MCH scanner checks if the selected addresses already exist in the configuration. Therefore there will be no duplicate addresses in the NATview configuration.

3.3 Establish a connection

To communicate with a MicroTCA system NATview must initialize an IPMI session first. This is done by selecting **Connect** from the **Application** menu. The following dialog box opens:



The Connect dialog displays all IP addresses of the systems that have previously been configured. (See section 4.4.6 for details.) Each line of the list box contains the **IP address**, the **online status** and the **timestamp of the last check**. The online status indicator has these meanings:

Yellow	System is currently under investigation.
Red	System is not responding.
Green	System is alive, NATview can establish a connection.

Select the IP address from the list or type-in a new one, and then click on **Connect.** NATview will start to scan the MicroTCA system, which will take a while.

During the scanning process the application shows a progress dialog that informs the application user about what is going on (see **Figure** 1 for examples – the ones you might see in your application might be different). Additionally every AMC module has a red flashing border as long as NATview is examining it.

There is one new feature in NATview 2.13 that shall simplify connecting to new systems a bit more comfortable: Every system's IP address NATview is able to connect to will be stored to the list of system in the configuration. If this automatism is unhandy for you it can be turned off in the network configuration.

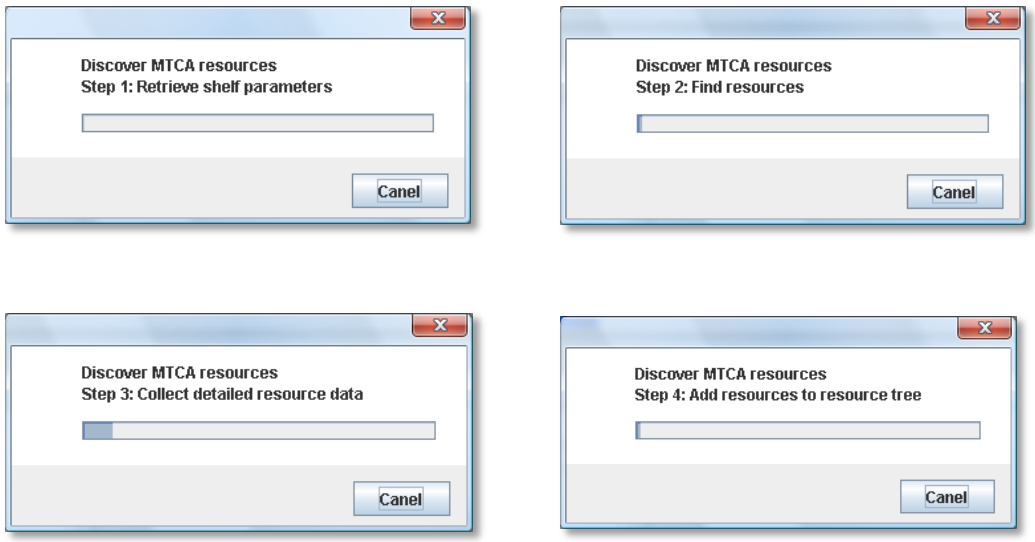


Figure 1: The Scanning Progress Dialog during the resource scanning process

After the scan has finished all detected resources are displayed in the main window.

The **connect options** are configurable, as there are:

Scan for MicroTCA resources	This option is enabled by default. Uncheck it if you only need the RMCP connection.

3.4 Checking the correct hot swap configuration

Most of the features of NATview will work right "out of the box" – this is how the software was designed and how it is intended to be used. Nevertheless there are situations where even NATview needs some assistance from the user. The hot swap configuration may be such a thing.

Why is this so?

It has something to do with different MCH firmware versions and the way that they provide the hot swap state of the FRU devices. A short review of the MCH firmware history might lead to a better understanding about what is going wrong here.

The early MCH firmware versions up and including release 2.17.13 did provide no means to read out the current M-state of FRU. The only tools at hand were the mandatory hot swap sensor of the FRU plus the SEL events for the state changes. The hot swap sensor provided the status of the hot swap leaver: if it was pulled out or pushed in. Together with the overall status of the FRU NATview was able to assume these three M-states:

M0	FRU removed
M1	FRU inserted, leaver pulled
M4	FRU fully operational, leaver pushed in

This hot swap sensor had the sensor type **0xf2**.

The whole thing changed with release 2.17.14 when the carrier provided (logical) hot swap sensors **for every** FRU device. First these sensors were grouped together under FRU 0 (the carrier FRU). Now it was the problem for the system software to assign the correct carrier hot swap sensor to its appropriate FRU device. This has been finally solved by assigning the same **Entity ID** and **Entity Instance ID** to the new hot swap sensors as has been to the FRU Device Locator Records. That way the (host) system software can attach the new hot swap sensors to the right FRU device as it does with the rest of the sensors.

The new hot swap sensors value is not the status of a hot swap handle but an **M-state bit field** that looks like this:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
undef.	**M6**	**M5**	**M4**	**M3**	**M2**	**M1**	**M0**

This new hot swap sensor has the sensor type **0xf0**.

NATview now needs to know which sensor type it shall use tro determine the M-state of the FRU devices. It is capable to read out the MCH firmware release major and minor version, but it cannot read out the subminor version. (This is actually a limitation of the IPMI specification.) It also knows that all MCH firmware releases up to version 2.16 used hot swap sensor type 0xf2. It finally knows that all MCH firmware releases starting with 2.18 provide the better new hot swap sensor type 0xf0.

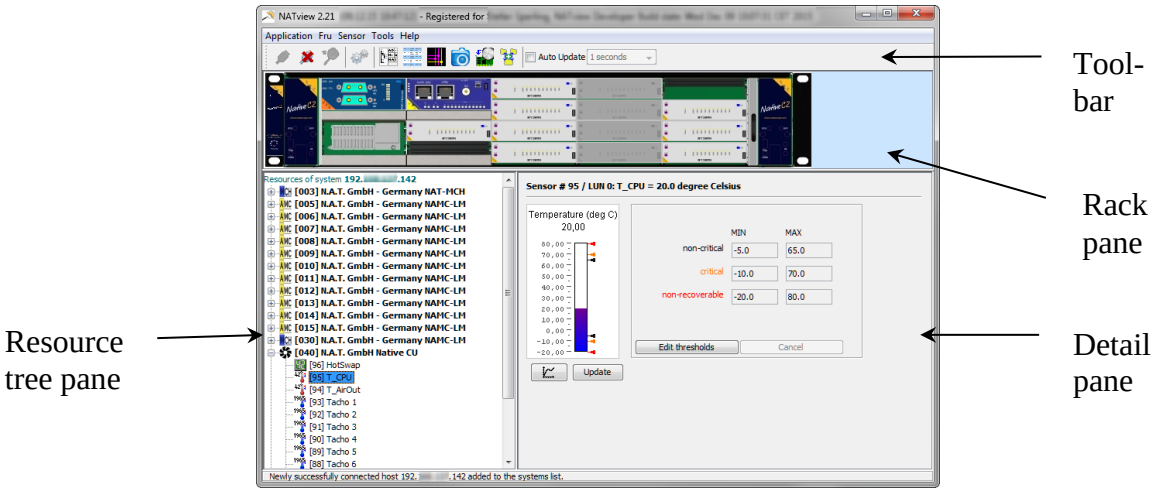
Tricky are all releases with version 2.17.x – for these releases it is up to the NATview user to configure the software correctly. This is done in the IPMI settings:

These are the correct settings for the different MCH firmware releases:

MCH firmware release	Hot Swap Sensor Type
<= 2.17.13	0xf2
>= 2.17.14	0xf0

4 Exploring the main window

After the MicroTCA system has been scanned NATview displays all detected resources in its four-parted main window:



Each of the four sections has its own functionality:

Toolbar	The toolbar contains buttons for commonly used commands, e.g. to connect to a MicroTCA system.	
Rack pane	The rack pane is used to display a graphical representation of the MicroTCA system. It shows the AMC boards that are inserted into the rack so the user can easily locate a specific board in the system. It is also used to display various alarm indications (e.g. if a temperature sensor has reached a threshold level).	Chapter 4.1
Resource tree pane	The resource tree pane provides an overview of all detected resources of the system.	Chapter 4.3
Detail pane	The detail pane is used to show the data of the selected device. If possible it also used to modify the data of the selected device (e.g.	Chapter 4.4

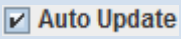
threshold sensors).

4.1 Toolbar

The toolbar contains buttons for commonly used commands. If NATview is currently offline, which means that there is no active connection to a MicroTCA system then the toolbar looks like this:

If NATview is currently communicating with a MicroTCA system then the toolbar looks like this:

The toolbar buttons have the following functions:

Button picture	Menu equivalent	Description
	**Application->Connect**	Opens the connect dialog to select the MicroTCA system to connect to. This button is only enabled as long as there is no active connection.
	**Application->Disconnect**	Disconnects an active connection to a MicroTCA system. This button is enabled only if there is an active connection.
	**Application->MCH Scanner**	Starts the MCH scanner tool to find all MicroTCA systems in your network.
	**Application->Configuration**	Opens the NATview configuration tool.
	**Sensor->Auto Update**	Globally switches sensor auto update on resp. off. Sensors are updated automatically only if their auto update option is enabled as well! (See chapter XXX for details on auto update.)
	-	Allows the selection of several auto update intervals. The interval change will get into effect immediately!

The toolbar contains also shortcuts to the NATview tools:

	Fru->Edit FRU Records	**Starts the FRU editor for the currently selected FRU.**
	Tools->Show Event Log	Opens the System Event Log viewer. This tool displays all system events that NATview read from the shelf manager event log (SEL).
	Tools->Backplane Viewer	Opens the Backplane Connectivity Viewer that visualizes the connections between the AMCs as well as between the AMCs and the MCH.
	Tools->System Dump	Easy to use tool that can collect information about a MicroTCA system. This information can be useful for the technical support.
	Tools->HPM Update	Starts the HPM update tool that can be used to update AMC modules easily and remotely.
	Tools->Fru File Cutter	This tool can be used to cut a backplane eeprom image into two discrete FRU files that can be used with the FRU editor. It also allows reassembling the file to a new eeprom image.

4.2 Rack Pane

The rack pane is used to display a graphical representation of the MicroTCA system. It shows the AMC boards that are inserted into the rack so the user can easily locate a specific board in the system. As with the real world system not all devices may be accessible in this view like e.g. power units, fan controls or logical devices.

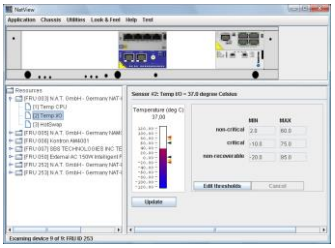


Figure 2: Landscape Rack

It is possible to display a graphic or photography of the chassis. If the chassis cannot be detected properly NATview will use a standard chassis graphic with vertically aligned AMC boards (see figure “Portrait Rack” for details).

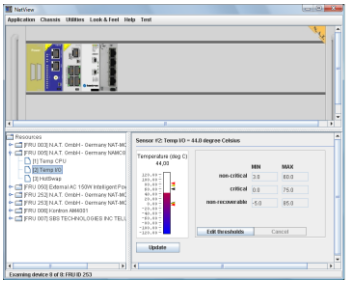


Figure 3: Portrait Rack

The board images represent the AMC modules that reside in the rack. The status of the image itself corresponds with the power state of the module:

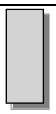
Picture...	Power State	
	M4	The AMC module is mounted to the rack and the ejector handle is closed. The board has payload power and is fully operational.
	M1	The AMC module is mounted to the rack but the ejector handle is open (pulled). The board has no payload power and is inactive. If the board has just been inserted to the rack the board has been initialized and is ready to begin the activation process.
	M0	The AMC module has been removed from the system.

Table 1: Board power-state representation in the rack pane

Hovering with the mouse-pointer over one of the board images displays a tooltip that contains the FRU ID, the manufacturer and product name, and the current power-state:

Figure 4: Tooltip of an activated board

Figure 5: Tooltip for a deactivated board

Please note that the tooltip of a deactivated board is slightly ghosted and has a different colour.

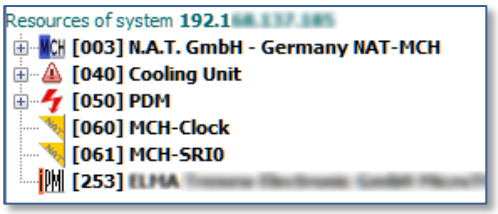
Clicking with the mouse cursor on one of the board images selects this board and the detail pane shows FRU information about the device.

4.3 Resource Tree Pane

The resource tree pane provides an overview of all detected resources of the system. It consists of FRU and SDR nodes:

The FRU nodes represent the FRU devices of the system. This may be a physical device like the AMC boards, the power units (PU), or the fan controller (FU). It includes also logical devices like the backplane EPROM (FRU 253).

The SDR nodes represent the sensors of the FRU devices. You may select one of the sensor nodes to read the current sensor values. If it is a threshold sensor you may also edit its threshold values.



The tree node icons show the following status information:

FRU Icons

- AMC Board
- Rear Transition Module
- MCH
- Cooling Unit (CU)
- Power Module (PM resp. PU)
- N.A.T. specific devices (e.g. MCH hub module)
- logical FRU device

Sensor Icons

- Threshold Sensor (general)
- Temperature Sensor
- Discrete Sensor
- Digital Sensor

Alert Icons

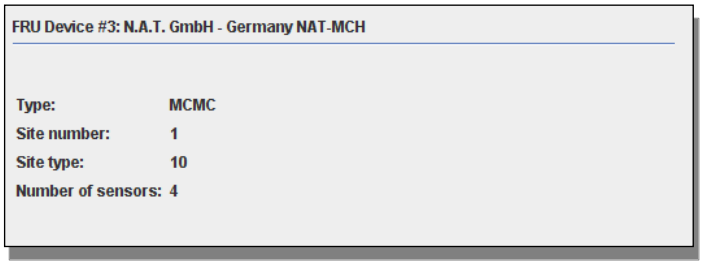
	Some communication error happened with the FRU device. Please check the NATView logfile for further information. Every faulty request-response-pair is being dumped to this file for further examination. The logfile can be configured in the NATview configuration, register "Debug".

4.4 Detail Pane

The detail pane is used to show the data of the selected device. Its content depends on the type of the data that is to be displayed.

4.4.1 FRU Devices

Clicking on a FRU node in the resource tree or on a board image in the rack pane displays information about the corresponding FRU device:



4.4.2 Sensors in General

NATview can display the values of the various sensors that a MicroTCA system can have: threshold sensors (temperature and non-temperature sensors), discrete sensors (e.g. hot swap sensors) and fan sensors.

All sensors are displayed with an **update** and a **history** button (see below):

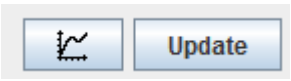


Figure 6: History and Update Button

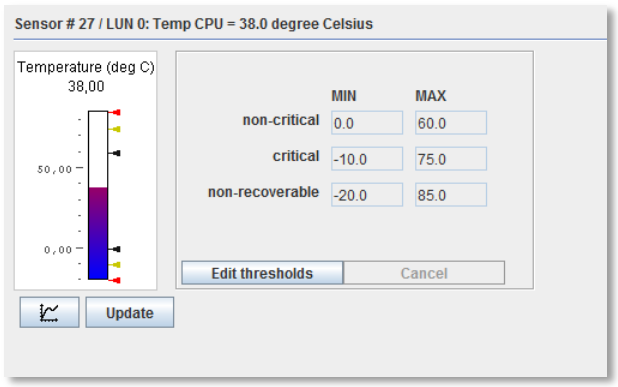
The update button requests the current value of the sensor from the shelf manger and displays it. To monitor the value course you need to press update several times or enable the auto update option (see chapter 6.2 for details).

4.4.3 Threshold Sensors

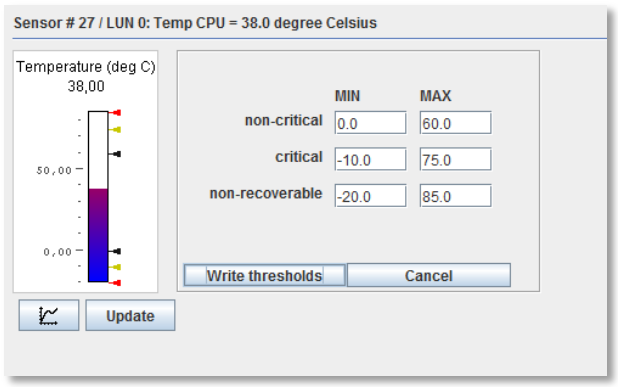
NATview distinguishes between temperature sensors and other threshold sensors – their representation differs slightly. Both sensor categories allow the modification of up to six threshold values.

4.4.3.1 Temperature Sensors

Temperature sensor data is displayed using a multi-coloured meter:



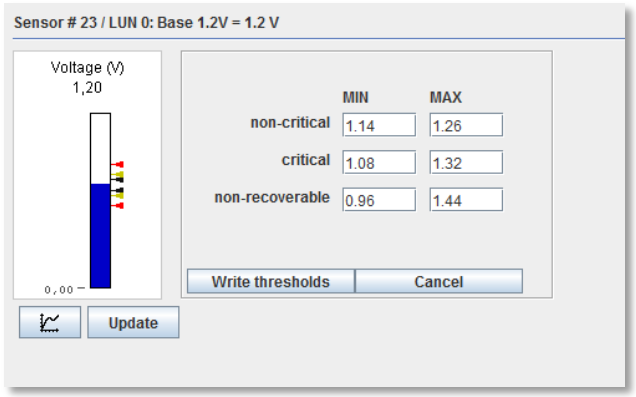
NATview displays only those threshold values that are masked readable. The currently displayed values might be refreshed using the **Update** button. **Edit thresholds** enters the edit mode, where the threshold values can be changed:



Write thresholds sends the new threshold values back to the sensor. **Cancel** leaves edit mode without changing anything.

4.4.3.2 Other Threshold Sensors

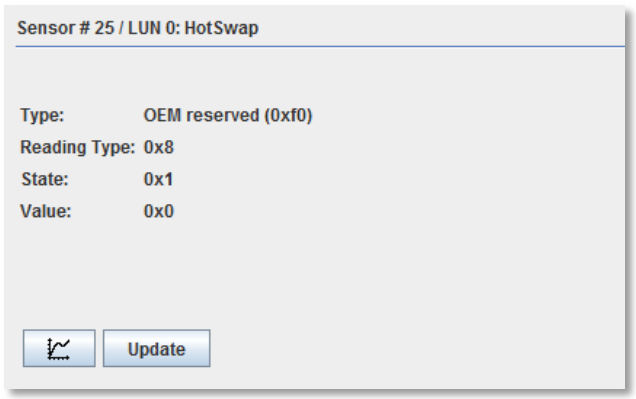
Other threshold sensors ("non-temperature sensors") are displayed in a similar fashion but with a monochrome meter:



Threshold manipulation functions in the same way as with temperature sensors (see chapter 4.4.3.1 for details).

4.4.4 Discrete Sensors

Discrete sensors are displayed using the following pane:



Clicking on Update, too, can refresh their values.

4.4.5 Fan Sensors and Fan Control (CU)

NATview knows basic FRU device support since version 1.8. Up to two cooling unit controller are supported. Clicking on the CU FRU device entry in the resource tree can control the fan level for each cooling unit. This opens the CU detail pane, which is a normal FRU detail pane with some CU specific add-ons.

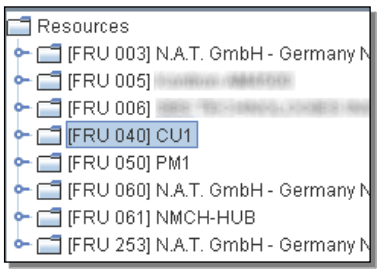
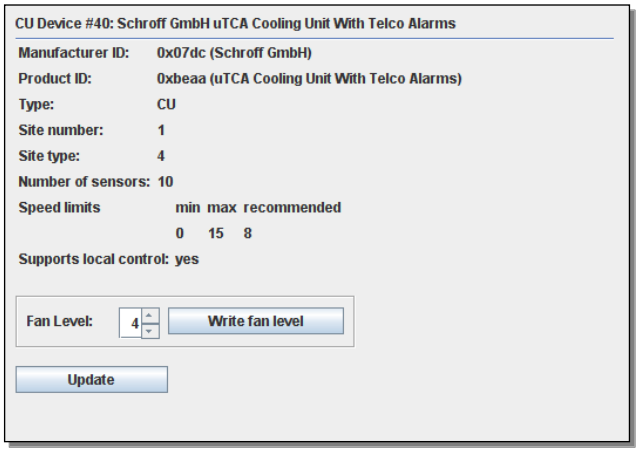


Figure 7: Selected CU

Simply write the new value to the *Fan Level* textbox then click on the *Write fan level* button: this changes the **fan level** immediately. Alternatively change the fan level by clicking on the spin controls. It is impossible to increase resp. decrease the fan level above resp. below the specified speed limits.

The requested fan level is being written to the appropriate cooling unit controller. Finally the new current fan level is being read back from the cooling unit controller which is necessary as the controller may advise a different fan level.

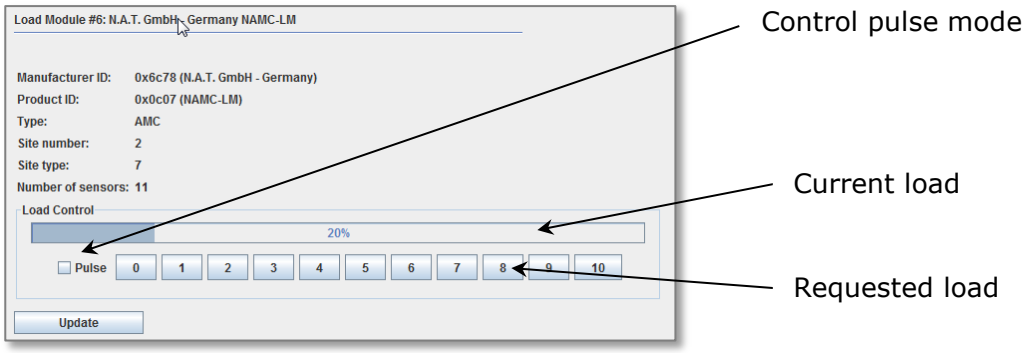


4.4.6 N.A.T. Load Module NAMC-LM

NATview 2.5 started to support the N.A.T. load module NAMC-LM. This module allows testing your system's power units in a controlled manner. For further information about the N.A.T. load module please visit our homepage at **www.nateurope.com**.

4.4.6.1 NAMC-LM Support in NATview 2.5

When NATview 2.5 finds a NAMC-LM in your system clicking on its FRU device node in the resource tree will show the following in the detail pane:



NATview will retrieve the current load whenever the load has been changed by the user or if it is updating the sensor value (on request by the user or while doing auto-updates).

The **Pulse** Option sets the load module in a special operating mode: If enabled two capacities are periodically loaded and unloaded to simulate changing power consumption of the AMC modules.

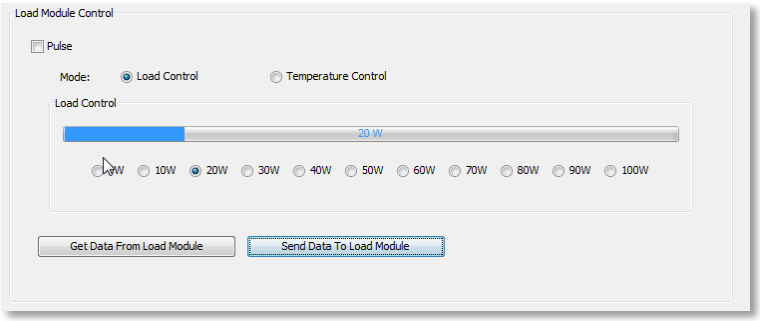
4.4.6.2 NAMC-LM Support in NATview 2.6

The support for the NAMC-LM was extended in NATview 2.6 as the zone option is supported now. To do this some changes in the load module GUI were necessary. This is what you see when clicking on the NAMC-LM entry in the resource tree:

As the NAMC-LM supports the zone mode for temperature tests NATview knows two operating modes for the load module: **Load Control** and **Temperature Control ("Zone Mode")**.

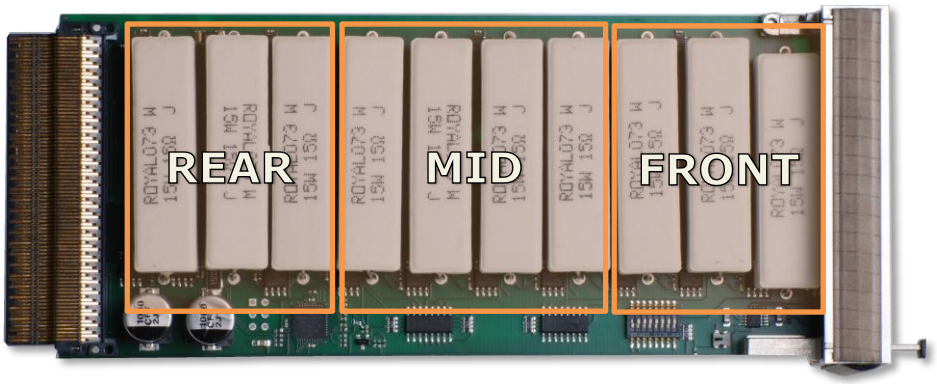
4.4.6.2.1 Load Control

This load module operating mode is used to test the stability of the power module of the system. The user can select to switch on one or all resistors of the load module – or any number in between. To do this select mode **Load Control** then choose the load to be used. Finally (re-) configure the load module by clicking on **Send Data To Load Module**. If sending was successful NATview will try to reread the configuration from the device to display it.

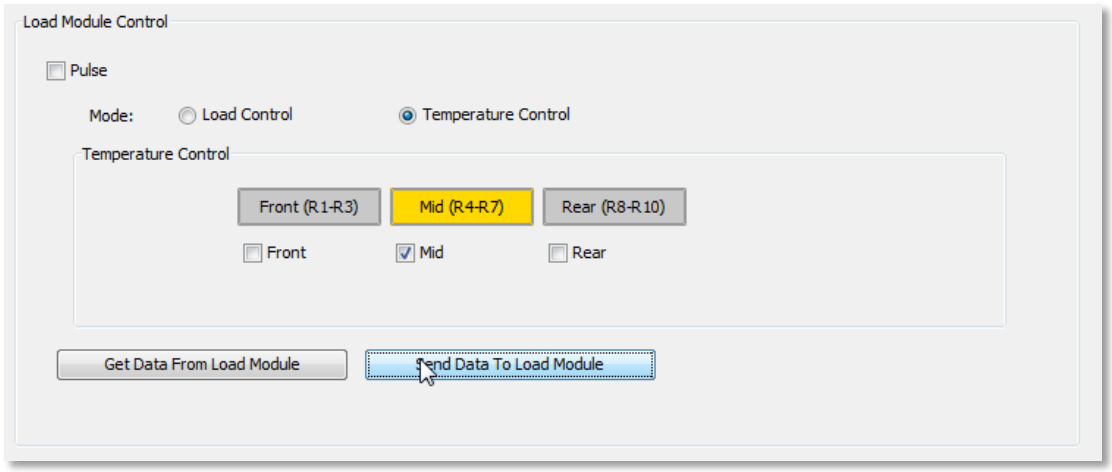


4.4.6.2.2 Temperature Control (“Zone Mode”)

Sometimes it is interesting to find out if the cooling units are really able to cool the system properly or if one AMC module is heat-sensitive. For such tests it is possible to configure the NAMC-LM into **zone** or **temperature control** mode. To do this select mode **Temperature Mode** then choose the resistor groups to be used (front, mid(dle), or rear).



Finally (re-) configure the load module by clicking on **Send Data To Load Module**. If sending was successful NATview will try to reread the configuration from the device to display it.



4.5 Configuring NATview

NATview is completely configurable from the graphical interface. All configuration data is being read from and written to an ASCII text file. This format was chosen for two reasons: Firstly this format is human readable which means that it can be exchanged between human beings. Secondly the ASCII format has been proven to be more robust as it can be repaired with a simple text editor.

To allow some kind of pre-configuration of the application a *skeleton initialization file* was invented. This file called `natview.skel` is being processed before `natview.ini` is read. This allows configuring a standard configuration that can later be modified and saved. NATview searches the skeleton- and the initialization-file in the same directory as the jar file.

The syntax of the configuration files is very simple: Both files consist of an arbitrary number of sections with an arbitrary number of key-value-definitions. A typical section may look like this:

[DefaultSection]
<key1>=<value1>
<key2>=<value2>

Comment lines start with a hash (#), a semicolon (;) or a double slash (//). The keywords do not have to start at the beginning of the line but they must not be hidden by a comment character.

Numerical (“integer”) values follow the equation sign without any quotation. Strings need to be enclosed with quotation marks.

At the time of this writing the following keys are supported[6]:

	Key[7]	Legal Values	Description
Gui			
	RACK_BACKGROUND_COLOR	<integer value>	RGB-Value of the background colour.
	SCAN_SYSTEMS	"Y", "N"	Periodically check the status of the configured systems and display it.
	LF_SELECTED	0..x-1	Index x of the selected look and feel scheme. This key is no longer supported.
	LF_SYSTEM	"Y", "N"	Activates the system look-and-feel. Otherwise the platform independent Java look-and-feel is used.
	SHOW_RESOURCE_TREE_ICONS	"Y", "N"	Displays type-specific icons in front of every entry of the resource tree.
	LF_NIMBUS	"Y", "N"	
	MAIN_FRAME_POS	"<x>,<y>,<width>,<height>	Position and dimension of the main frame window.
Ipmi			
	DISCOVER_RESOURCES	"Y", "N"	If "Y" then scan for resources after connect.
	SEND_CLOSE_SESSION_REQ	"Y", "N"	Terminates the session by sending a CloseSession request to the peer.
	SEND_POLL_MESSAGES	"Y", "N"	Periodically sends request messages to the peer to prevent the session from timing out. Not necessary if SCAN_CARRIER_MANAGER is "Y".
	SCAN_CARRIER_MANAGER	"Y", "N"	Periodically scans the carrier manager SDR to detect **M0↔M1** transitions.

[6] For documentation purpose this table still contains keys that are no longer supported.
[7] In the configuration file every key is written in a single line.

	DUMP_IPMI_MSG	"Y", "N"	Dump all incoming and outgoing messages to the debug port.
	CLEAR_SEL	"Y", "N"	Clear MCHs SEL after system scan.
	DEFAULT_CARRIER_ID	<carrier manager ID>	ID of the carrier as saved in the MicroTCA Carrier Information Record.
	USE_PRODUCT_MAPPER	"Y", "N"	Use the product mapper extension.
	IGNORE_SOME_ERRORS	"Y", "N"	Ignore some errors when writing data to FRUs. This is a workaround option! Use with caution!
	SDR_UPDATE_DELAY	<integer>	Delay in seconds for sensor auto update.
Debug			
	LOGFILE	<file name string>	File name and path of the logfile.
	SHOW_TEST_MENU	"Y", "N"	Show the test commands in the main window and in the fru editor.
	<ModuleName>	"Y", "N"	Controls which part of the application products debug messages. Currently defined are: GENERAL, IPMI_SESSION, IPMI_MESSAGE, IPMI_LIB, HPM, BPV, FRU_EDITOR, DRAG'N'DROP, PRODUCT_MAPPER, FRU, SDR, SENSORS, EVENTS, PRODUCTION_FRU, CONFIGURATION, CARRIER, HISTORY, DRAG'N'DROP (for Fru editors drag'n'drop feature)
Network			
	MTCA_SYSTEM_x	e.g. "132.147.160.254"	IP address of host number x.
	MTCA_SYSTEM_SELECTED	0..x-1	Index x of the selected system. This key is no longer supported.
	SCAN_SYSTEMS	"Y", "N"	Starts a resource scan of the MTCA system.
	AUTO_ADD_ADDR	"Y", "N"	Automatically add the IP address of a new system to the configuration.

FruEditor			
	CONFIRM_CLOSURE	“Y”, “N”	Confirm fru editor window closing.
	FRU_EDITOR_VERSION	1, 2	1 = old Fru Editor 2 = new Fru Editor This key is no longer supported as is the old FRU editor.
	LAST_READ_DIR_PATH	<valid directory path>	Path to the directory the last fru read command had taken its fru info file. This parameter is not platform independent.
	LAST_WRITE_DIR_PATH	<valid directory path>	Path to the directory the last fru write command had taken its fru info file. This parameter is not platform independent.
	PCM_POWER_BUDGET_WARNINGS	"Y", "N"	Power Configuration Manager (PCM) shows warnings if the power configuration may exceed the power budget.
Licence			
	LICENCE_KEY	<licence key string>	Licence key string as received from N.A.T.
	LICENCE_HOLDER	<licence holder name string>	Licence holder name string as registered by N.A.T.
Alarms			
	SHOW_SENSOR_ALARM_ NOTIFICATIONS	“Y”, “N”	Indicate if one or more sensors have exceeded a threshold value.
	SHOW_SENSOR_COUNTER	“Y”, “N”	Display the number of sensors that have exceeded one threshold.
	SEL_LOGFILE	<path/name of the SEL logfile>	Controls where to write the SEL event data. If no log file is defined no data will be saved.
	COLUMN_SEPARATOR_INDEX	0-6	Selects the separator character that should be used to separate the event log data columns. The index selects one character from the following: 0 = **<Space>,** 1 = **<Tab>,** 2 = **|,** 3 = **/,** 4 = **,** 5 = **-,** 6 = **_**
	INSERT_ENVIRONMENT	“Y”, “N”	Inserts the JAVA environment to the log file.

HPM			
	RETRY_INITIATE	5 – 30	Number of max. retries for the Initiate Upgrade Action.
	RETRY_UPLOAD_BLOCK	5 – 30	Number of max. retries for the Upload Firmware Block Action.
	RETRY_FINISH	5 – 30	Number of max. retries for the Finish Firmware Upload Action.
	RETRY_DELAY	0 – 2000	Delay time between every retry in ms.
	IGNORE_TIMEOUT_x	"<MAN>,<PROD>,<time>"	Ignore the upgrade timeout for one board. x is a counter, starts with 0 and is incremented afterwards for every board. <MAN> is the manufacturer ID. <PROD> the product ID. <time> is the timeout in seconds.
BPV – all parameters of the Backplane Connectivity Viewer. Please do not touch!			
DIALOG			
	selectedTab	0 – 9	Selected tab in the configuration dialog.
WORKAROUNDS			
	SWAP_ORIENTATION_BIT	"Y", "N"	The MicroTCA orientation bit indicates horizontally oriented AMC modules. If SWAP_ORIENTATION_BIT is set it indicates vertically orientated boards.
	EXCHANGE_COORDINATES	"Y", "N"	Exchange the x- and the y-part of the AMC coordinates.
	IGNORE_SOME_ERRORS	"Y", "N"	Ignores some less severe errors when writing FRU data.
	IGNORE_MISSING_SHELF_MANAGER_FRU	"Y", "N"	Does not complain if FRU #254 is missing.
FFC (Fru File Cutter)			
	COMBINEDFRUFILENAME	"Y", "N"	Path/Name of the combined FRU file.
	FRU253FILENAME	"Y", "N"	Path/Name of the carrier FRU file.
	FRU254FILENAME	"Y", "N"	Path/Name of the shelf FRU file.
Layout – all parameters for layouting the windows properly. In case some window positioning looks odd, simply delete this COMPLETE section!			

5 Fru Menu

The Fru menu of NATview contains the following commands:

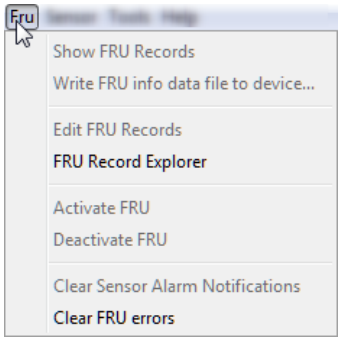


Figure 8: Fru Menu

The first three commands, **Show FRU Records**, **Write FRU info data to device** and **Edit FRU Records**, work on the currently selected FRU device.

Edit RU Records starts the FRU editor that allows extensive manipulation of the FRU data of a MTCA device or the chassis backplane.

New since version 1.21 is the command **Clear Sensor Alarm Notifications**.

Also new but since version 2.9 are the commands **Activate FRU** and **Deactivate FRU**.

Clear FRU errors resets all FRU error notification in the resource tree.

5.1 Show FRU Records

This command displays the interpreted contents of the FRU data records of the selected FRU device in a new window:

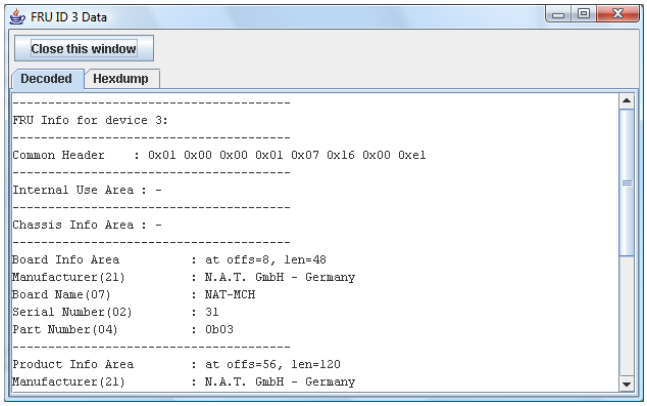


Figure 9: FRU Data Records

It is also possible to view the raw FRU data records as a hex dump by clicking on the register **Hexdump**.

The contents of these windows can be selected (Ctrl + A) and copied to the clipboard (Ctrl + C) for further processing.

5.2 Write FRU info data to device

This function allows writing the content of a FRU data file directly to the FRU device. A common usage would be to update the contents of the backplane FRU eeprom of a chassis. It is kind of a shortcut for starting the FRU editor, loading a FRU image from disk and finally writing the FRU data to the device.

5.3 Edit FRU Records

This command starts the NATview FRU record editor[8], which allows extensive data manipulation of the FRU data information of a MTCA device or of the MTCA chassis backplane FRU info.

The editor may also be started by double-clicking on a FRU node in the resource tree.

[8] Please note that there are no longer two versions of the FRU editor.

5.3.1 General FRU Editing

The menu command “Edit FRU Records” opens the FRU record editor for the currently selected FRU device:

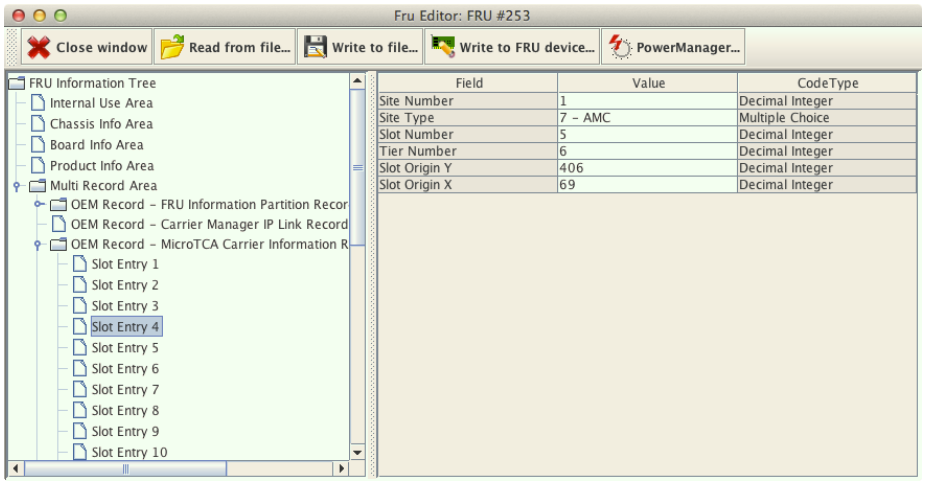


Figure 10: FRU Editor

The editor is structured like the Windows registry:

The left column contains an area tree that has a leaf for every FRU data area, one for Internal Use Area, Chassis Info Area, Board Info Area and Product Info Area. The very last leaf indeed is special as it holds the Multi Record Area. As the name implies this record area may and often does contain more than one record. Each of these records is therefore represented as a child leaf underneath the Multi Record node.

The right column of the FRU editor displays the content of the currently selected leaf of the area tree in a three-column table. Each entry of the table may be selected with the mouse.

Editable entries are displayed with white background colour, while non-editable entries show up in grey. (As the actual colour depends on the OS and the colour scheme of the desktop you could also say that all editable fields are highlighted.)

Double-click on an entry in the middle column of the data table to open an edit window:

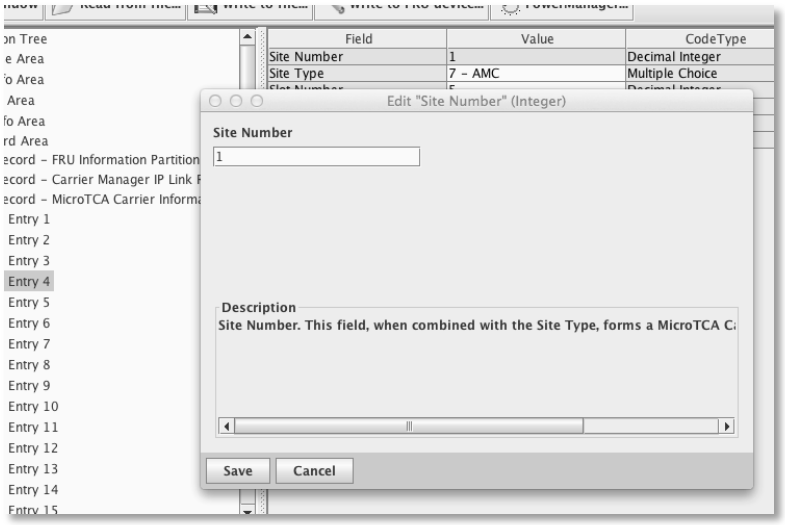


Figure 11: Edit Window for simple data types

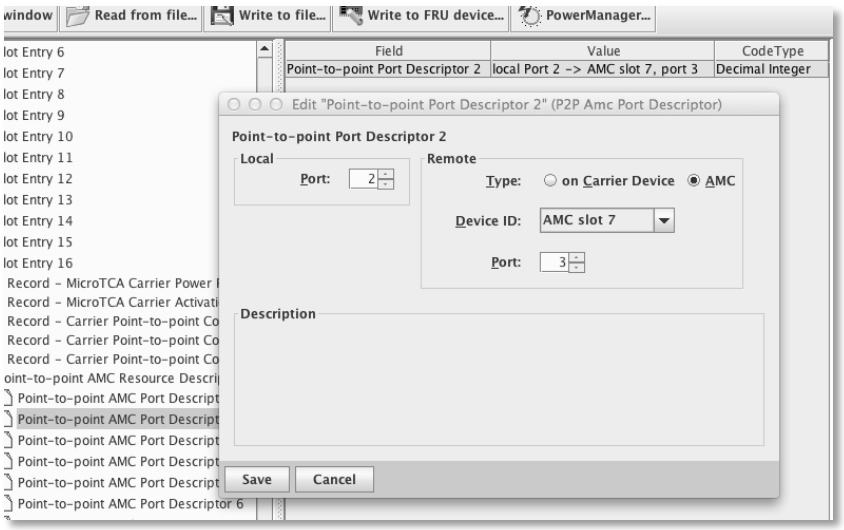
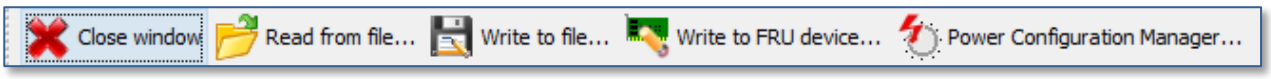


Figure 12: Edit Window for complex data types

The FRU Editor provides an input box if the entry is writable. If the entry is read-only no edit window is displayed.

5.3.2 The FRU editor toolbar

The FRU editor toolbar contains command buttons for extended FRU editor functions:



The commands are described in detail in the following sections.

5.3.2.1 Read from file

This command allows reading a FRU file from a file. The file is being read, parsed and displayed in the editor and replaces all previous data.

5.3.2.2 Write to file

This command allows writing the current FRU editor data to a file. This file may later be read to the FRU editor using the command **Read from file**.

5.3.2.3 Write to FRU device

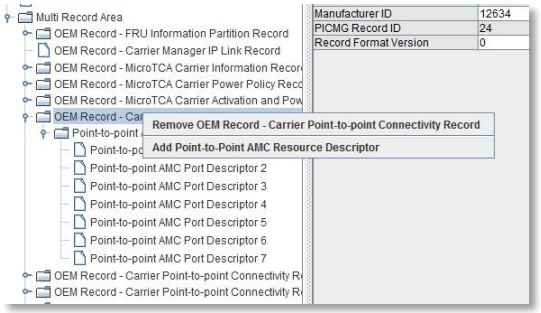
This command writes the current FRU editor data to the FRU device. FRU editor will always write back to the device where the current editor data was previously taken from. After the data has been successfully written the device should be power-cycled to ensure that the new FRU data is being processed accordingly.

5.3.2.4 Power Configuration Manager

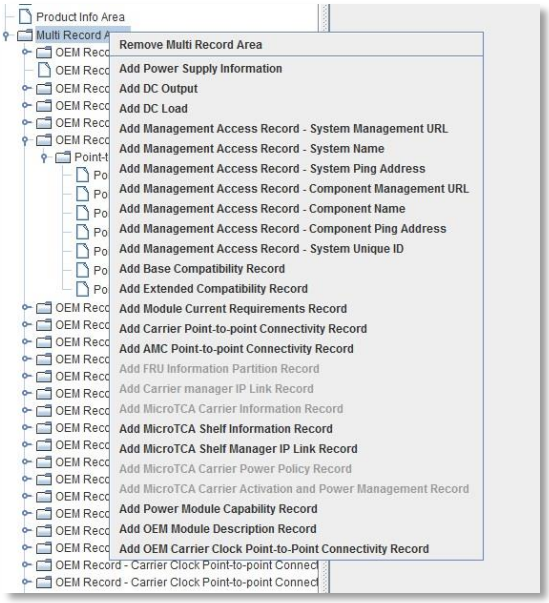
This button starts the **Power Configuration Manager** (see section 5.3.2.4 for details). This command is only available for the carrier FRU (FRU ID 253).

5.3.3 Creating and Removing Records

The FRU editor of NATview Professional can create new FRU records as well as delete existing ones. This functionality can be accessed via the popup menus of the Record Tree nodes. If the user clicks with the secondary mouse button on a tree node a popup menu like this is shown:



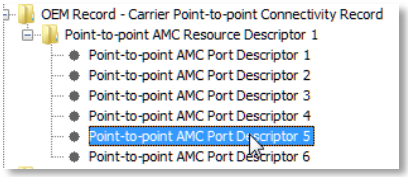
The content of this popup menu varies from node to node. Performing the same click on the “Multi Record Area” node presents a different picture:



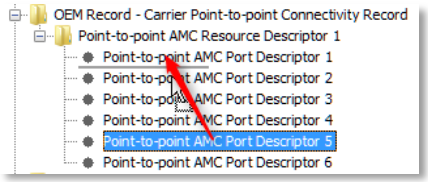
5.3.4 Moving records

Starting with NATview 2.21 the FRU editor allows changing the order of records using drag and drop.

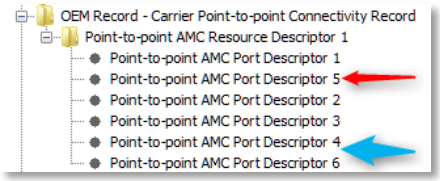
1. Simply click on the record that needs to be moved and keep the mouse button pressed.



2. Drag it to the location where it needs to be.



3. Release the mouse button. The record will appear at the new location (upper/red arrow) and be deleted at the old one (lower/blue arrow).



NOTE It is impossible to drag a record under a new parent. In the example above it would be impossible to drag **Port Descriptor 5** to **Resource Descriptor 2**!

5.3.5 Editing special FRU Record Types

The NATview FRU Editor is able to support editing some of the important record types. Currently supported record types are:

- FRU Information Partition Records
- Carrier Location Records
- Carrier Activation and Power Management Records

The editing of these record types is going to be discussed in the following sections.

5.3.5.1 FRU Information Partition Record

This FRU record type is the data directory. It is commonly used to store the layout of a backplane EEPROM. The partition record is defined like this:

Partition number	Start offset	End offset	Partition type
1	0	<offset>	Carrier manager FRU data
2	<offset>	<offset>	Shelf manager FRU data

5.3.5.2 Carrier Location Records

This FRU record type is used to describe where the AMC boards, the cooling- and the power units can be found in the MicroTCA chassis. They were originally designed to aid a service person in the field to locate the defective unit. NATview uses this data to display the AMC- and power-unit-images at their correct locations in the chassis.

5.3.5.3 Carrier Activation and Power Management Records

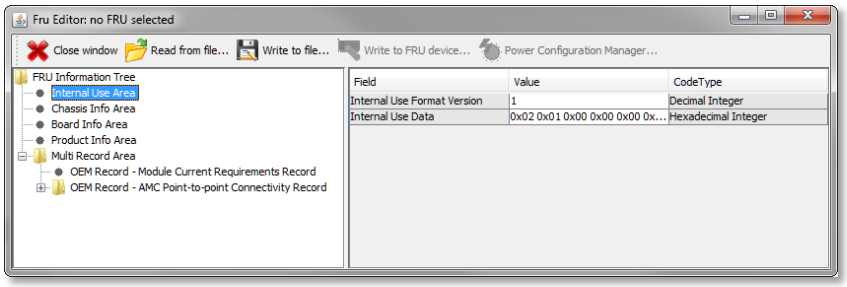
This FRU record type is used to describe how the AMC boards (and the cooling- and power-units) may be deactivated/activated and how much power each of them may require for proper operation. NATview does not interpret this data yet.

NATview 2.13 invents a new tool for editing the power-related parameters of the backplane FRU, the so-called **Power Configuration Manager**. See chapter 5.3.7 for more details.

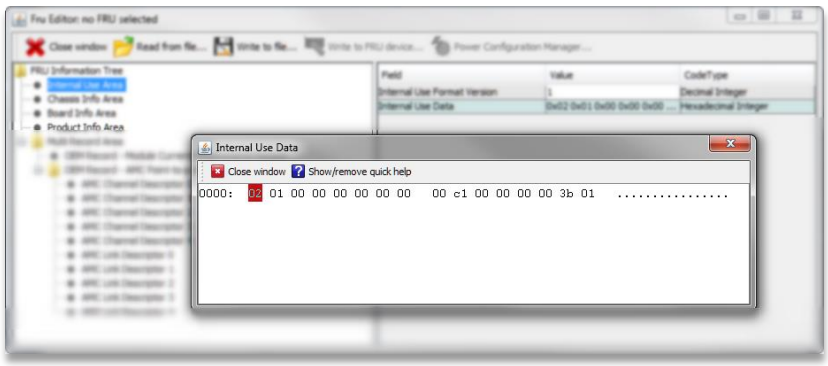
5.3.6 Internal Use Area

From NATview 2.7 the FRU editor supports the **Internal Use Area (IUA)**. The structure of this area is not defined any further by the specs and can be used freely by an application and/or the board/chassis.

Whenever an IUA exists the FRU editor will display it like this:



Double-clicking on the Internal Use Data Value field will start a hexadecimal editor that allows modifying the existing bytes and adding or removing them.



This hex editor is used to view and modify unstructured binary data. The editor buffer is of fixed length but can be extended or shortened. The hexadecimal area (to the left) accepts the characters **`0-9`** and **`a-f`**. The ASCII area (to the right) accepts all ASCII characters.

Special keys are:

Moves the cursor in the data buffer.

Extends the data buffer by one **0x00** byte when the last byte of the buffer is being selected.

Deletes the currently selected byte, then shifts the following buffer to fill the gap.

Deletes the byte to the left of the selected one, then shifts the buffer to fill the gap.

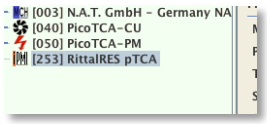
5.3.7 Power Configuration Manager (PCM)

The Power Configuration Manager (PCM) is an extension to the FRU editor that has been introduced with NATview 2.13. The PCM can be opened when editing a Carrier Manager FRU (a.k.a. FRU #253). For all other FRU devices or when offline editing a FRU file (without an active connection to a MicroTCA system) the PCM is not available.

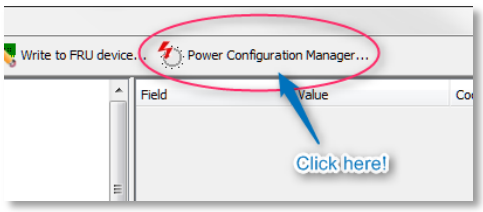
5.3.7.1 Starting the Power Configuration Manager

Start the Power Configuration Manager with these three easy steps:

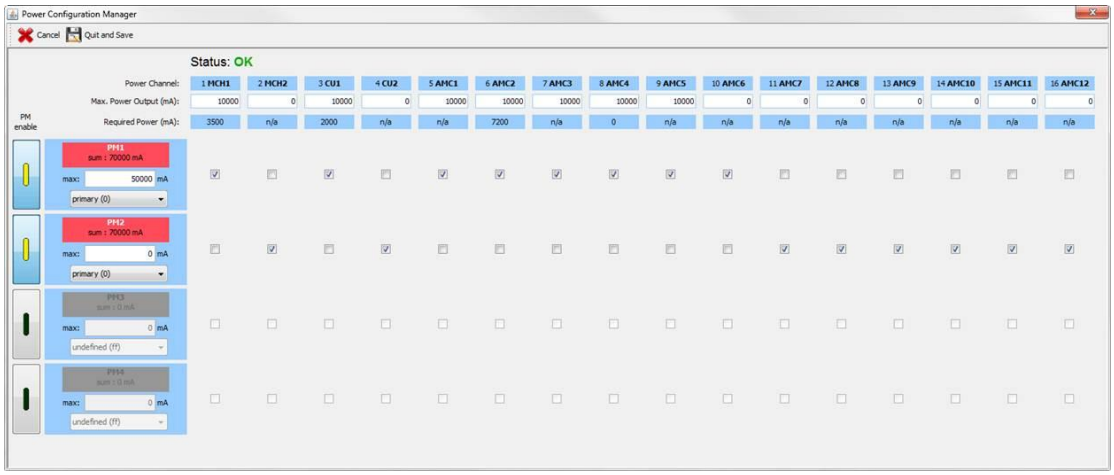
1. Open the FRU editor for FRU #253:



2. Click on the Power Configuration Manager button:



3. The Power Configuration Manager opens and you are ready to easily change your power configuration:



5.3.7.2 Principle of operation

The main purpose of the Power Configuration Manager is to assign the power channels of a system (“slots”) to the available power sources in a system. A matrix of 16 power channels and 4 power modules visualizes this assignment.

The Power Configuration Manager gets its information from different sources:

1. The Carrier FRU (FRU #253) being loaded in the FRU editor. It searches for the **Carrier Power Policy** and the **Carrier Activation and Power Management** Records.
2. It also scans all the AMC modules in the system for their **Current Requirements Record**s.

Every power module may be:

1. Enabled or disabled using the big buttons labeled "PM Enable" on the left-hand side. For every enabled power module the PCM generates a Power Policy Record. Only if a power module is enabled you can select power channels for it.
2. Every power module can be configured for **primary** or a **secondary** operational mode. (The Power Configuration Manager also supports the operational mode 0xff, which stands for undefined. It is only implemented for completeness as the specifications defines it. It is of no further use in real life and should be avoided.)

A power channel is assigned to a power module by simply ticking its correspondent check box. For every selected ("ticked") power channel the PCM will generate a Power Policy Descriptor attached to the corresponding Power Policy Record. If no power channels are selected no descriptors are generated thus leaving a standalone Power Policy Record if the power module was enabled.

It is not necessary to use the power modules in an orderly fashion, e.g. it is ok to enable PM2 and PM4 (but not PM1 and PM3). What configuration is useful depends on the target chassis and its backplane configuration.

After all changes to the matrix have been done they can be written back to the appropriate power records. It is possible to trace all changes in the power records from within the FRU editor. Therefore the Power Editor can be used as a learning tool to understand what needs to be configured where in the different power records of the Carrier FRU.

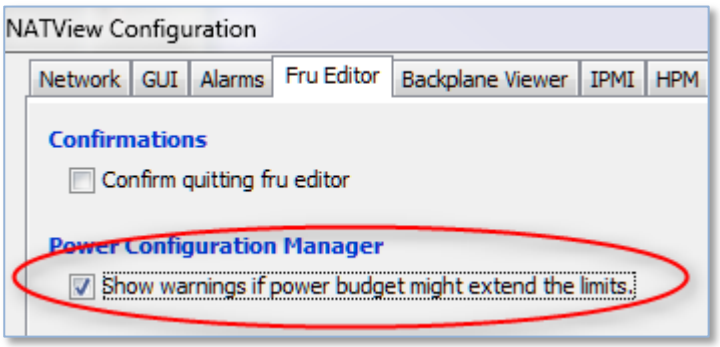
5.3.7.3 Configuration Checks

The Power Manager performs some consistency checking after every change that has been done by the user. The most important two are:

- **Primary/secondary integrity check**
  This check ensures that every power channels is – if selected – connected to one primary power module and optional to exactly one secondary power module. (If the channel is not selected for any power module at all no checks are performed.)
  If this condition is not met for one power channel the top most power channel label will be displayed in bright red and the status will change to Invalid power configuration.

- **Current requirement overflow**
  Every power module should declare the maximum amount of current it can provide. On the other side every AMC can declare the maximum amount of current it needs. Additionally the backplane manufacturer should provide information about how much current every power channel (“slot”) can handle. The Power Configuration Manager now checks that the currency sum of all selected power channels does not exceed the maximum of current a power module can provide. As the maximum power channel current of the backplane is usually much higher than what is required by the AMCs the Power Channel will detect a current requirement overflow in most cases. This is just a warning to do double checks.

Please note that this checking is disabled by default. To enable it you need to open the NATview configuration and select **Show warnings if power budget might extend the limits** (see screenshot below).



BUT the modified configuration can be saved in any case, regardless of any errors or warnings. The Power Manager is trying to facilitate the power configuration – it cannot release the system manager from understanding what (s)he is doing.

5.3.7.4 How to set up a specific power configuration

To help you in configuring your system for different power configuration setups please visit chapter 9.1 for tips in setting up some of the common power configurations.

5.3.7.5 Why is it not called Power Manager no more?

The Power Configuration Manager shall facilitate the power configuration of a chassis. It can be used to control the power distribution of power modules and AMCs that are not yet present in the system. It does NOT necessarily manage the power of your active system (chassis, MCHs, CUs, AMCs).

You could e.g. load the backplane FRU (#253) of a chassis you don't necessarily own, modify it and then save it back to the chassis.

5.4 FRU Record Explorer

The FRU Record Explorer scans all FRU devices (again) and displays the results in a separate window (Figure 13). This function is similar to the **Show FRU Records** (see chapter 5.1 for details) but it permits to select from all known FRU devices.

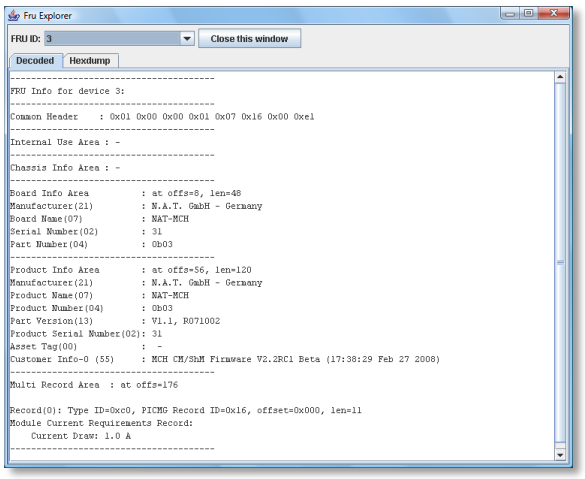


Figure 13: FRU Record Explorer

Standard is the decoded view; to view the raw FRU data click on the **Hexdump** register latch.

The contents of these windows can be selected (Ctrl + A) and copied to the clipboard (Ctrl + C) for further processing.

5.5 Activate FRU / Deactivate FRU

These commands change the state of one FRU in the same way as it would be by pulling-out resp. pushing-in the hot swap lever.

6 Sensor Menu

The sensor menu contains all commands for controlling the sensor history and the auto update feature.

6.1 Sensor History

The sensor history allows monitoring the value course of one sensor in a separate window. The history window for a selected sensor can be opened using the menu **Sensor->Open Sensor History** or by clicking on the history button ( ). This will open a new history window (see Figure 14: Sensor History).

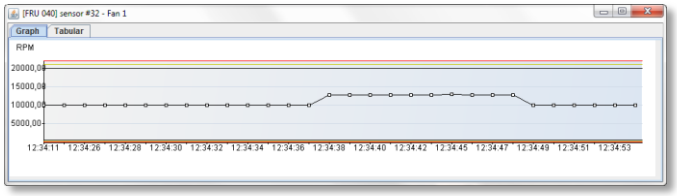


Figure 14: Sensor History

The sensor history may be displayed as a graph or as a tabular (see Figure 15: Sensor history as tabular). Values over resp. below certain thresholds are marked in a special colour.

It is also possible to open multiple history windows for one sensor – the only limit is your desktop and your memory. Every history window spawns its own child process, which retrieves its data independently. These windows will not close automatically when the connection to the MicroTCA system is disconnected – they will simply be not updated anymore as their original sensors have vanished!

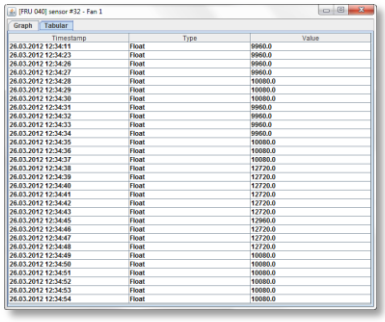


Figure 15: Sensor history as tabular

6.2 Sensor Auto Update

It is sometimes necessary to monitor one or more sensors for a certain amount of time without the need to update all sensors manually. This can be achieved by enable the **auto update option** for these sensors.

The sensor auto update option can be enabled for any sensor that has been detected by NATview. It is enabled by selecting the sensor and then right clicking on it to open the senor context menu. The sensor context menu currently contains only one command, **enable auto update** (see Figure 16: Enable Auto Update for details).

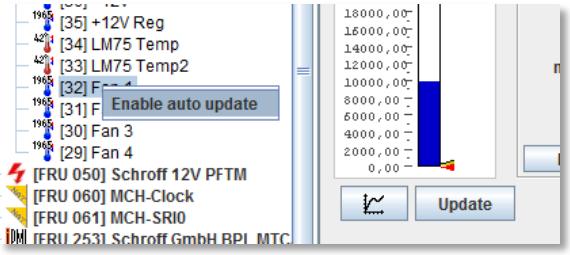


Figure 16: Enable Auto Update

A global task periodically checks all sensors of the system. If the auto update option has been enabled the current sensor value is being requested from the sensor. With the sensor response all sensor values are properly updated.

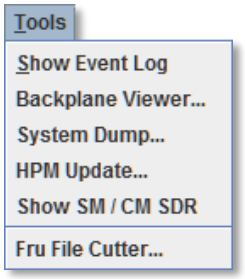
The auto update function must be switched on globally with the **Auto Update** checkbox in the toolbar:

The **sensor auto update delay time** can be adjusted using the combo box in the toolbar. Please note that this delay is really a delay in the sense that all sensors are being processed and after that the update tasks waits for the delay time to expire. After that the sensors are being processed again and so on. This means that the minimum time a sensor is being updated is the delay time plus *x*. *x* is determined by the time it takes to query all requested sensors (and this amount of time may vary).

Starting with NATview 2.21 the content of a history window is scrolled automatically so that the latest sensor value is visible. Scrolling in the direction of older values interrupts this behaviour. Scrolling back to the latest sensor value restarts auto scrolling.

7 Tools Menu

The Tools menu contains the following commands:



Note: In earlier versions of NATview the Tools menu was named *View*.

Note 2: All commands of the Tools menu are only accessible when using NATview Professional.

7.1 Show Event Log – the Event Viewer

The command **Show Event Log** is used to open the event viewer.

The event viewer can be used to monitor all incoming IPMI sensor events. The events are extracted from the MCH's System Event Log (SEL) by periodically polling for new events every five seconds.

It permits to set display filter conditions and to save the events in various file formats. When the event viewer opens the first time it may look similar to Figure 17.

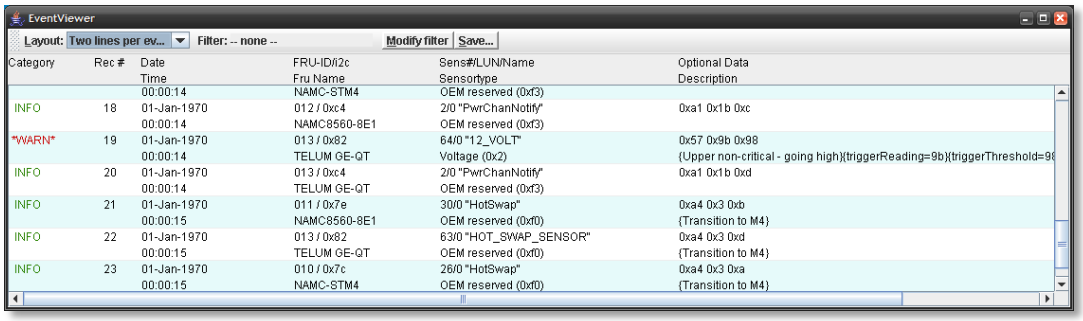


Figure 17: Two-line-mode event viewer

7.1.1 Switchable layout

The registered NATview event viewer can display the event data in two different flavors. The default setting is the two-lines-per-event mode. The user may switch to the one-line-per-event mode if a more compact data presentation is required.

7.1.1.1 Two-lines-per-event mode

Every event record is displayed using two text rows. Every row contains the data items that are described in Figure 18. The detailed meaning of every data item in one event record is described in Table 2.

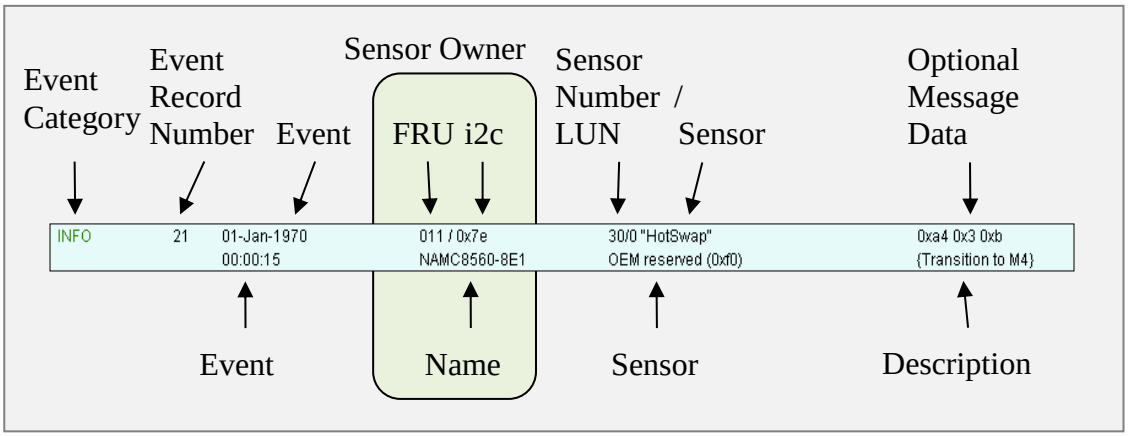


Figure 18: Two-line-mode event record

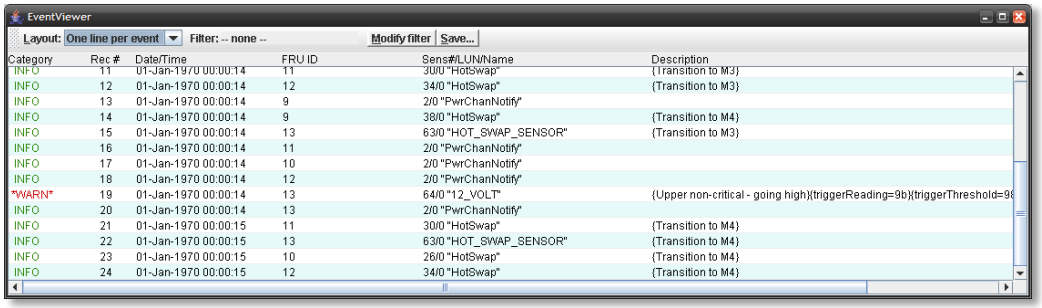
Data item	Data Type	Legal Values (Example)	Comment
Event Category	Enumeration	INFO, *WARN*, *CRITICAL*, NONRECOVER	Categorization that tries to estimate how severe the event would damage the device. Crossing a recoverable threshold is regarded to be less severe than a non-recoverable threshold.
Event Record Number	Integers starting with 1	-	Record number used by the SEL
Event Date	dd-MMM-YYYY	11-Nov-2008	Event date, starting with 01-Jan-1970 (“the epoch”)
Event Time	hh:mm:ss	14:15:32	Event time

FRU ID	Decimal Integer	6, 7, 8,…	FRU ID of the sensor owner, usually the module where the sensor is located.
i2c Address	Hexadecimal Integer	0x7c	i2c slave address of the sensor owner, usually the module where the sensor is located.
Name	String	NAMC-STM4	The product name of the sensor owner device (as being read from the device FRU data).
Sensor Number / LUN	Integer / Integer	26 / 0	MCH-mapped sensor number and sensor LUN. Please note that these values differ from the sensor-local SDR!
Sensor Type	String “(“Integer”)”	Voltage (0x2)	Sensor type, useful to properly interpret the sensor value.
Sensor Name	String	“HotSwap”	Sensor name as it was being read from the sensor SDR.
Optional Message Data	Hexadecimal integer (bytes)	0xa1 0x1b 0xc	Event Data 1 – 3[9].
Description	“{“String”}”	{Transition to M4}	Optional description of the SEL event.

Table 2: Two-line-mode event record description

[9] See IPMI specification, clause 26, “SEL records” for details.

7.1.1.2 One-line-per-event mode



Every event record is displayed using one text row. Every row contains the data items that are described in Figure 18. The detailed meaning of every data item in one event record is described in Figure 19.

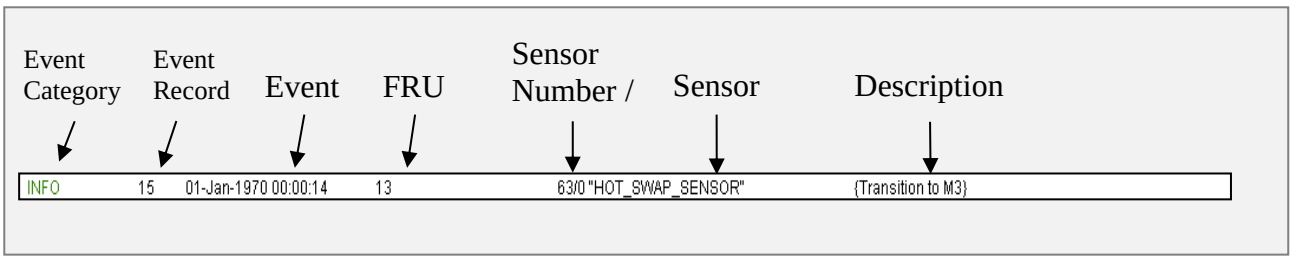


Figure 19: One-line-mode event record

Data item	Data Type	Legal Values (Example)	Comment
Event Category	Enumeration	INFO, *WARN*, *CRITICAL*, NONRECOVER	Categorization that tries to estimate how severe the event would damage the device. Crossing a recoverable threshold is regarded to be less severe than a non-recoverable threshold.
Event Record Number	Integers starting with 1	-	Record number used by the SEL
Event Date	dd-MMM-YYYY	11-Nov-2008	Event date, starting with 01-Jan-1970 (“the epoch”)

Event Time	hh:mm:ss	14:15:32	Event time
FRU ID	Decimal Integer	6, 7, 8,…	FRU ID of the sensor owner, usually the module where the sensor is located.
Sensor Number / LUN	Integer / Integer	26 / 0	MCH-mapped sensor number and sensor LUN. Please note that these values differ from the sensor-local SDR!
Sensor Name	String	“HotSwap”	Sensor name as it was being read from the sensor SDR.
Description	“{“String”}”	{Transition to M4}	Optional description of the SEL event.

Table 3: One-line-mode event record description

7.1.2 Event Filter

The event viewer of the registered version of NATview has the ability to filter its views. There is no loss of data if an event filter is active: all SEL events are recorded but only the desired ones are displayed.

7.1.2.1 Theory of Function

The event filter is activated and configured by clicking the **Modify filter...** button in the event viewer. A dialog box as in Figure 20 opens. NATview supports three different filter parameters that look into different parts of the event message: *FRU ID*, *Sensor number/LUN* and *event category*.

If there are no criteria set in one of the filter parameters this parameter is ignored. This means e.g. if there is none of the FRU IDs selected all FRU IDs will be accepted by the filter. The same applies to the other filter parameters.

All selections within one filter parameter are logical or’ed. All the filter parameters are AND’ed.

Maybe an example can clarify this subject a bit: Suggested that a user needs to see all events from two specific sensors (e.g. 20/0 and 21/0) on one AMC module (e.g. FRU ID 6) then these settings need to be done:

1. Check **FRU 6** in the Fru Filter section.
2. Check the sensor lines with sensor LUN 0 and sensor number 20 resp. sensor LUN 0 and sensor number 21.
3. Click on OK.

The event viewer will be updated to show only those event messages that fulfil the filter requirement.

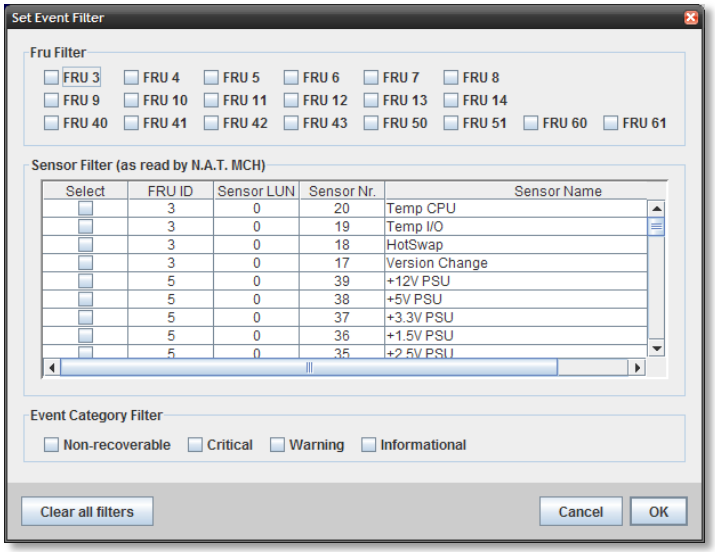


Figure 20: Event filter configuration dialog

7.1.2.2 FRU Filter

NATview currently allows filtering on both of the MCHs (FRU 3 and 4), four Cooling Units (FRU 40 to 43), two Power Units (FRU 50 and 51), the N.A.T. clock and fat pipe hubs (FRU 60 and 61) and the twelve AMC modules (FRU 5 to 14).

All event messages with at least one of the selected FRU IDs are accepted.

7.1.2.3 Sensor Filter

The sensor filter is somehow special as the SEL events display the sensor events using the MCH-mapped sensor number/LUN. These sensor numbers are different from the sensor-local numbers used by the AMC modules. Unfortunately there is no way for NATview to map these local sensor numbers into those used by the MCH.

The individual sensors are selected for filtering by checking an entry in the sensor list. All event messages with at least one of the selected sensors are accepted.

7.1.2.4 Category Filter

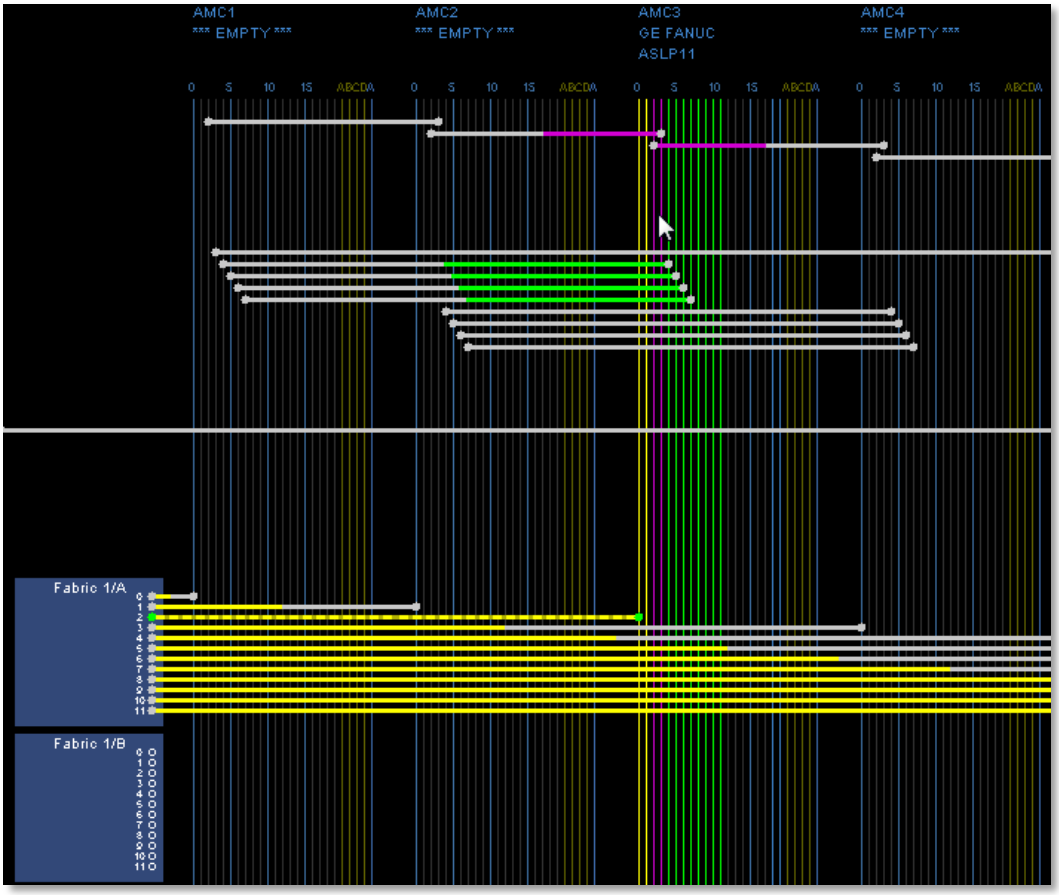
Every event message can be sorted into one of the categories: info, warning, critical, and non-recoverable.

All event messages with at least one of the selected categories are accepted.

7.1.3 Saving events to file

The NATview event viewer allows saving an ASCII- or HTML-representation of the SEL events to a file. Clicking on the **Save...** button in the event viewer windows does this. This opens a standard file saving dialog where the file name and the file format can be selected.

7.2 NATview Backplane Connectivity Viewer



The NATview backplane connectivity viewer visualizes the connections of the chassis backplane:

- Connections on the chassis backplane between the MCH(s) and the AMCs.
- Resources and protocols that are offered by the individual MCH(s) and AMC.
- The link status of the individual connections.

The following sections will discuss the different functional views of the NATview backplane viewer.

7.2.1 Operating Principle

7.2.1.1 Overview

The NATview backplane connectivity viewer (BPV) displays the connections between the MCH(s) and an AMC resp. between two AMCs. The viewer consists of the following areas:

On the leftmost resp. rightmost side all MCH data and clock fabrics are shown. There is one block for every data fabric of the MCH and one block for all MCH clocks. All fabrics for one MCH are vertically aligned from top to bottom.

The AMC ports are horizontally aligned from left to right, starting with AMC 1. For every AMC twenty data ports and five clock ports (TCLKA-TCLKD, FCLKA) are displayed.

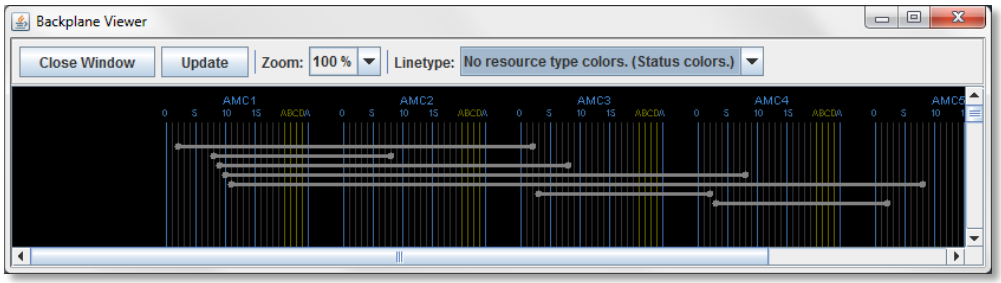
Connections are displayed using horizontal lines between an MCH fabric and an AMC port/clock or between two AMC ports/clocks. Every connection may be colored to symbolize the type of resource being supplied by the connection participants (see section 7.2.1.3 for details).
It is possible to get more information about a single connection by pointing at it with the mouse.

7.2.1.2 Connections on the chassis backplane

The individual AMCs can only communicate with each other or with the MCH(s) if the backplane has connections between its slots. Such a connection is nothing more than some type of electric wire. The precise type of data being exchanged is not subject of interest on this level.

The map of these connections can be found in the backplane FRU's AMC point to point connectivity records.

You may look at these connections by selecting connection line type "No resource type colours". (See screenshot below.)



7.2.1.3 Resources of the AMCs resp. MCH(s)

NATview can display the resources of the AMCs and the MCH(s) in the system. Examining the AMC P2P connectivity records of the AMC modules retrieves this information. The same applies to the MCH(s) of the system.

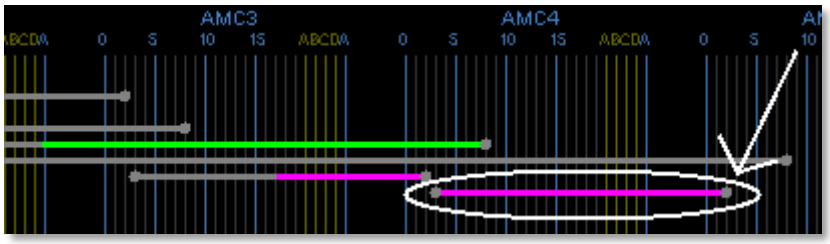


Figure 21: Example of a matching SATA connection

Every backplane connection must have two matching resource types. Only then it is possible to establish a data link between the endpoints of the backplane connection. If you have a close look at the screenshot above (Figure 21) you will notice the SATA connection coloured in pink. This shows that both participants of these connections offer a SATA resource. If only one AMC offers the resource then it would look as in the connection above: only the right side of the connection is coloured pink.

Annotation for MCH firmware releases until and including version 2.9: The MCH firmware until and including version 2.9 does not supply any information about its resources. Therefore NATview cannot display any information about the MCH resources if these MCH firmware releases are used.

7.2.1.4 Connection Status

A click on the **Update** button of will retrieve the current status of the links. An active link is displayed by moving ants (dots).

7.2.2 Connection Line Type

The NATview backplane connectivity viewer (BPV) can show the kind of resources that can be found on both sides of a single connection. This is done by colouring the backplane connection lines. The BPV supports two different ways to do this:

Resource type colours, half/half. The left half of the connection is coloured depending on the resource type of the left device. The right half of the connection is coloured depending on the resource type of the right device.

Resource type colours, two stripes. The upper half of the connection is coloured depending on the resource type of the left device. The lower half of the connection is coloured depending on the resource type of the right device.

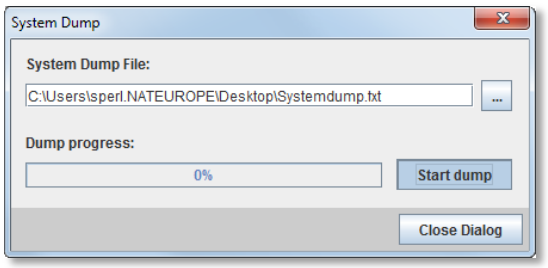
7.2.3 Zoom

It is possible to zoom in and out in a wide range. This allows getting an overview of all backplane connections as well as further examination of a specific connection.

7.3 System Dump

7.3.1 How to create a system dump file

The System Dump tool is used to collect all data about the currently connected MicroTCA system into one single file. Using System Dump is as easy as 1-2-3:



1. Select the System Dump tool from the menu **Tools->System Dump**.
2. Enter a valid file name to store the system dump to.
3. Start the system dump by clicking the **Start dump** button.

7.3.2 Extract the FRU info data from the system dump file

The system dump file contains all data that is necessary to get an overview of the customer's system. Sometimes it is necessary to dig deeper – if you need to examine the FRU info data itself. The system dump file contains hex-dumps of this. To recreate the binary files the tool **ExtractFru** can be used.

ExtractFru is a console Java application. It is started form the command line by entering the following:

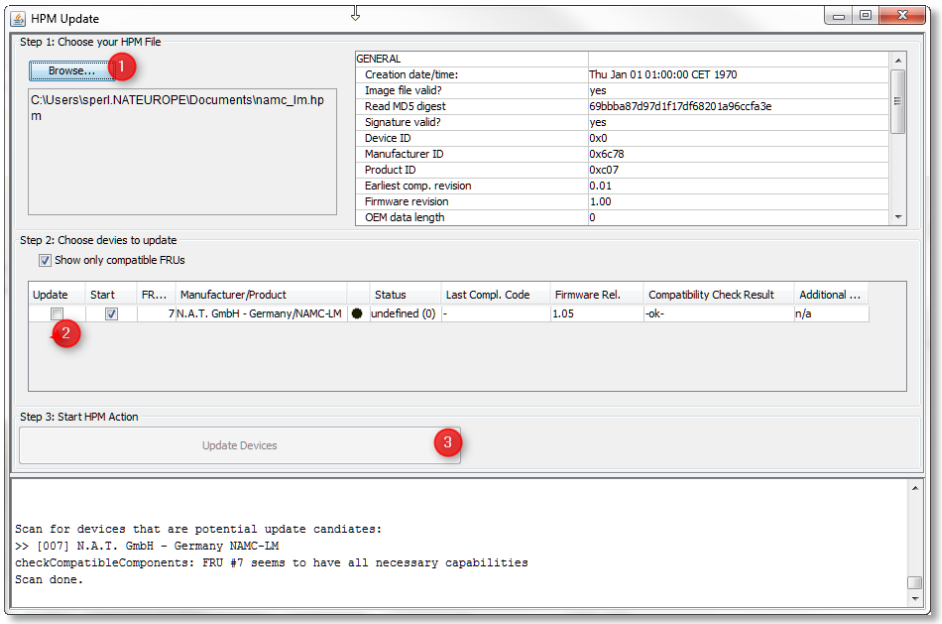
```
java –jar ExtractFru.java <SystemDumpFileName>
```

It creates a subdirectory named **`extractfru`** where it places a file for every FRU in the system. Every file is named **`fruXXX.bin`** with **`XXX`** ranging from 000 to 254. These files can be read into NATview's FRU editor.

7.4 HPM Update

Beginning with NATview version 2.7 the application supports the HPM firmware update standard. This means that all HPM compatible FRU devices can be updated using NATview.

Start the command **HPM Update** from the **Tools** menu. A window similar to the following will open:



Updating a FRU device is performed in three steps:

❶ Select your HPM file

This file is usually named **`<product_id_or_something>.hpm`**. NATview will process the file, display information about it in the right side window and list all devices it has found that can be updated with the file.

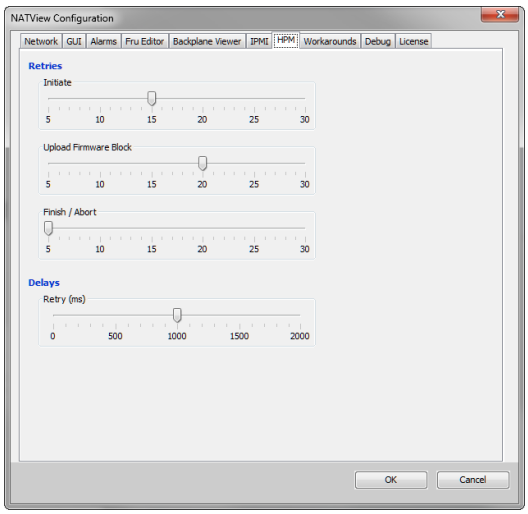
❷ Select the device(s) to be updated

By ticking the **Update** check box the devices that need to be updated are selected. If at least one device is selected that way the **Update Devices** button is enabled.

❸ Start HPM Action

Pressing the **Update Devices** button will start the update process.

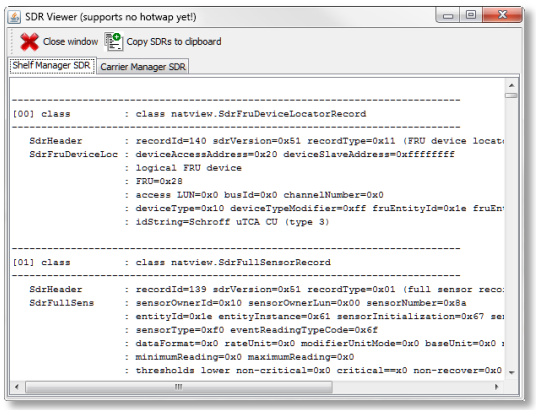
HPM defines a set of timers and retry counters whose values can be modified in the NATview configuration:



One can try to increase the number of retries or the amount of delay if the HPM update of a specific board hangs or breaks. Please inform the manufacturer if an AMC cannot be updated properly with the default configuration as this raises the suspicion that the HPM implementation of the board is faulty.

Nevertheless it might be possible to update such a faulty board with an invalid upgrade timeout by adding a new IGNORE_TIMEOUT line to the HPM section of the natview.ini. For all those cards NATview ignores the upgrade timeout value of the board and uses the manually configured timeout instead.

7.5 Show SM / CM SDR



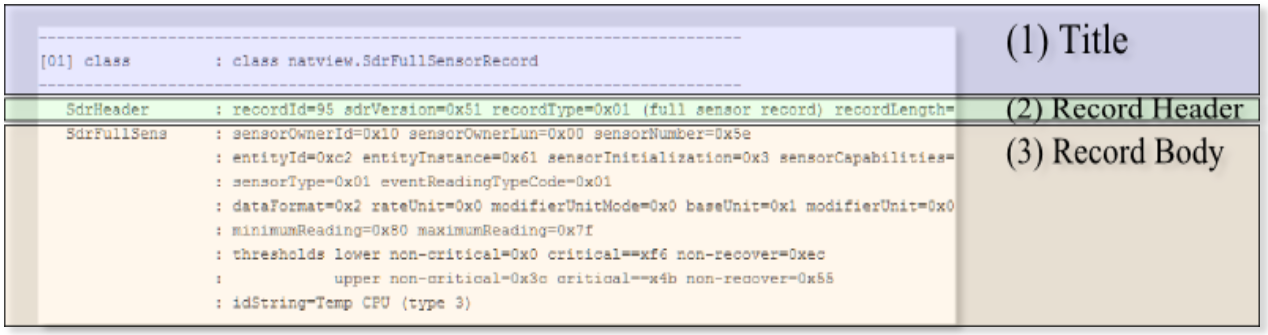
The name of this command should read "show shelf manager and carrier manager SDR" (with SDR being an abbreviation for "sensor data record") – but that was too long. The sole purpose of this tool is to display the contents of the Shelf Manager and the Carrier Manager SDRs.

These SDRs contain the information about all sensors of the system, e.g. sensor type, the range and the unit of the data. Only with this data it is possible to interpret and to process the sensor data.

The current SDR viewer is very raw – it simply dumps the information that NATview has retrieved from the MCH. There is no stylish GUI or similar.

The viewer consists of two sections, one for the Shelf Manager SDR, the other one for the Carrier Manager SDR. The SDR data is displayed as ASCII text dump which can be sent to the system clipboard for further processing.

Every SDR consists of several different records that are displayed in the following format:



(1) **Title** This is the internal class name of the record used by NATview which is more or less the same name as defined by the specifications. (As Java does not allow space characters in class names those spaces are removed.)

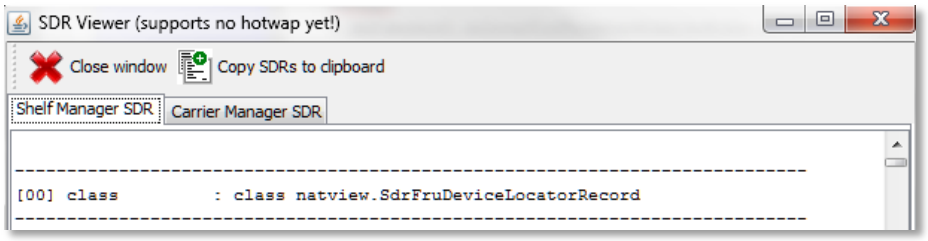
(2) **Record Header** The SDR record header that is common to all record types.

(3) **Record Body** Here goes the record data which is specific to the record type.

The detailed record structure can be looked up in the IPMI specification, see reference [1] for details. In the following the most common SDR items shall be shortly discussed.

recordId	The unique ID of the record in this repository. The record ID may and will differ for one sensor in the Shelf Manager and Carrier Manager repository!
sdrVersion	Should be at least 0x51 (for IPMI Version 1.5?).
recordType	Describes the record type and therefore what data fields can be found in the record body.
THE FOLLOWING FIELDS WILL NOT BE PRESENT IN ALL RECORDS AS THEY ARE RECORD TYPE DEPENDANT!	
deviceAccessAddress	The address of the device on its i^2c bus.
sensorType	What kind of data is displayed.
entityId	Together it's the entity Id and the entityInstance that are used together to find/attach the proper sensors. NATview finds all AMC sensors by searching up the entityId
entityInstance	and the entityInstanceId of the FRU device locator record. Whenever there are some missing sensors please check the entityIds and entityDeviceId.

The content of the two viewer windows can be copied as ASCII test to the system clipboard. Use the button **Copy SDRs to clipboard**:



7.6 Fru File Cutter

7.6.1 Purpose

A chassis' backplane FRU information is vital for a MicroTCA system in many ways:

- It defines if a certain slot is powered.
- It describes the interslot communication paths on the backplane ("wiring").
- It contains information about the location of the AMCs in the chassis.
- many many more…

Unfortunately this backplane information can be delivered in two different formats:

1. One image of the backplane eeprom (which contains the FRU information for the FRU #253 and FRU #254).
2. Two images with every image containing the FRU information for FRU #253 resp. FRU #254. You will need these two image files to load, edit, and store your backplane FRU.

This difference counts when it comes to updating the chassis eeprom:

- When the chassis backplane FRU is completely empty, NATView will usually fail scanning the system and therefore cannot open the FRU editor. In this case you need to use the MCH's diag capabilities with the eeprom image!

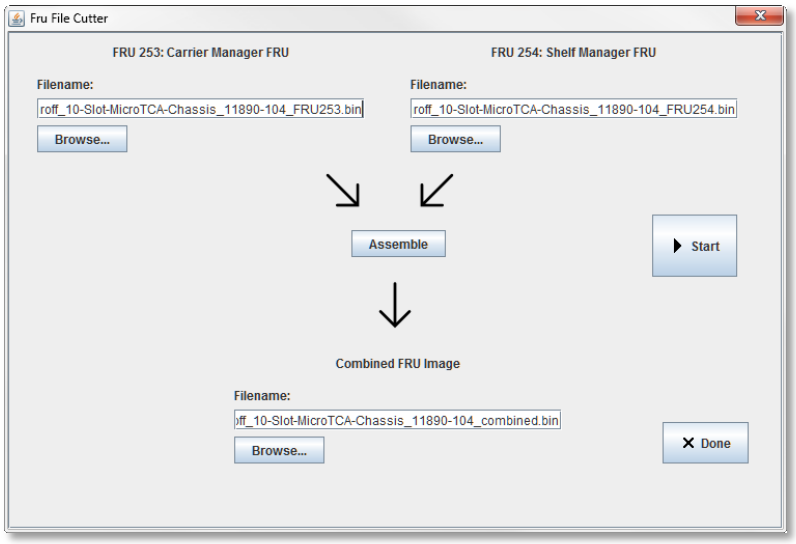
  The same applies when one of the backplane FRUs is missing (most often it is the Shelf FRU with ID 254) – when there is no FRU no FRU data can be written (a.k.a. "vicious circle").

- The chassis contains a valid eeprom image but someone needs to change certain parameters. In this case one would scan the system. After it has successfully initialized it should be possible to open the FRU editor, modify the parameters and save it back again.

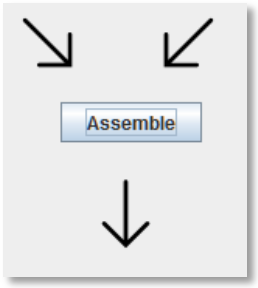
The Fru File Cutter allows to convert eeprom files to FRU files and back!

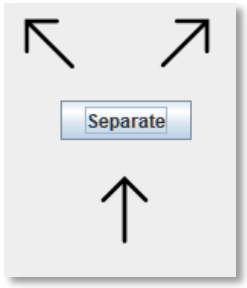
7.6.2 Usage

The usage of the Fru File Cutter is easy: Select **Tools->Fru File Cutter** from the menu. The following window appears:



You need to supply a file name for both the Carrier and the Shelf Manager FRU and for the combined FRU image. Click on the **Assemble** button to switch the operating mode:





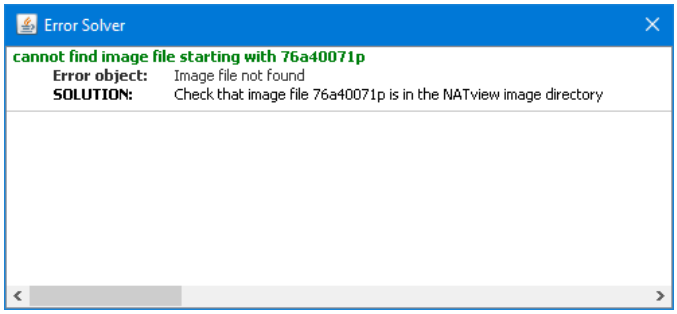
Finally click on **Start** to start the process. You can close the window by clicking on the **Done** button.

8 Help Menu

The help menu contains information about the application version and contains the command for requesting a trail license.

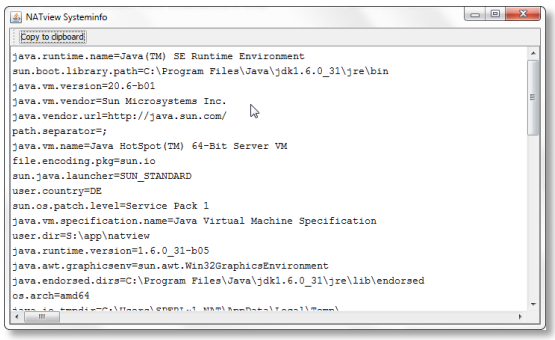
8.1 Show error recovery hints

This tiny little box collects error information and more or less useful solution hints. The Error Solver box below displays an error message that NATview cannot find an image file starting with **`76a40071p`**. It also suggests that you check that such a file can be found in NATview's image directory.

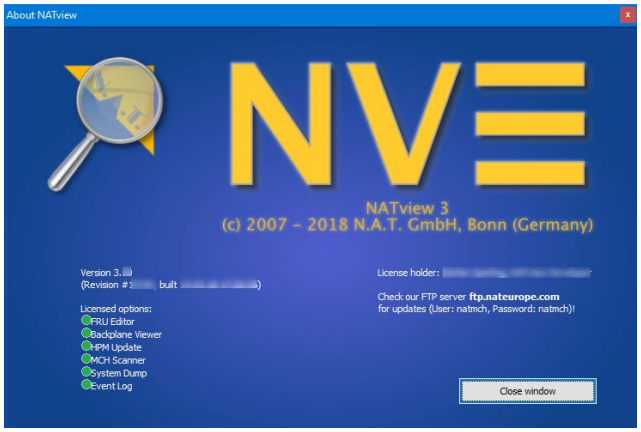


8.2 System Info

This command displays the Java environment. It may be copied to the system clipboard so it can be saved with a text editor or mailed to the NATview support. **Note: Please do never modify or delete the environment list before sending it to the NATview support!**



8.3 About



Besides the application version and the version date the user can find out:

The licensed options

This list shows which of the application features are enabled with your license key – green means that the feature is enabled, grey that it is not.

The license holder

The registered NATview version shows the name of the license holder.

8.4 Request trial license

The NATview distribution package on the N.A.T. FTP server does not contain a valid license key. Therefore only the functionality of NATview Easy can be used. If you want to test the complete NATview functionality you can either

- Buy a full NATview license; send a purchase order to sales_team@nateurope.com. Or you
- Request a trial license.

A trial license is a full NATview license key with a limited validity of 30 days. To get one select the command **Request trial license** from the **Help** menu. Enter your name and company as the license holder and store the license request file to disk. Send this request file to sales_team@nateurope.com. In return you will get one trial license key. All trial license key requests are logged to a database to prevent multiple trial license requests from the same license holder.

9 Solutions

This chapter is different from the rest of this manual. Its focus is not on specific functions of the software but on normal problems from "out in the field" and how they can be solved using NATview. You could also call this chapter "a collection of How-Tos".

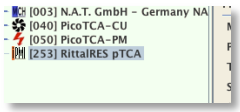
9.1 Power Configuration

This chapter will show you how you can use the NATview Power Configuration Manager (PCM) to configure some of the common power configurations of a MicroTCA system. Basic knowledge about the PCM is expected; if unsure please visit chapter 5.3.7 for details.

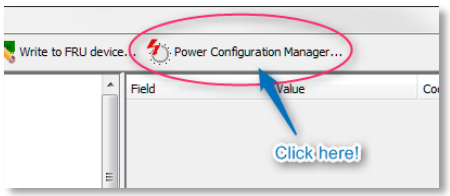
9.1.1 Starting the Power Configuration Manager

Start the Power Configuration Manager with these three easy steps:

1. Open the FRU editor for FRU #253:



2. Click on the Power Configuration Manager button:



3. The Power Configuration Manager opens and you are ready to easily change your power configuration:

9.1.2 General Aspects

Although there is a wide range of valid power configurations there are certain rules to be respected.

1. One of the basic rules is that every power channel can have one primary and one secondary power module at most. It is possible to have no power module – in this case the power channel is unused. It is also possible to have only one primary power module but no secondary – in this case there is simply no fall-back power module for the channel.
2. The other thing is that you should not assign more power consumers (e.g. AMCs) to a power module than it can handle. Unfortunately this is something that cannot be prevented completely with a clever power configuration as this depends on the power consuming devices that are used in the chassis.

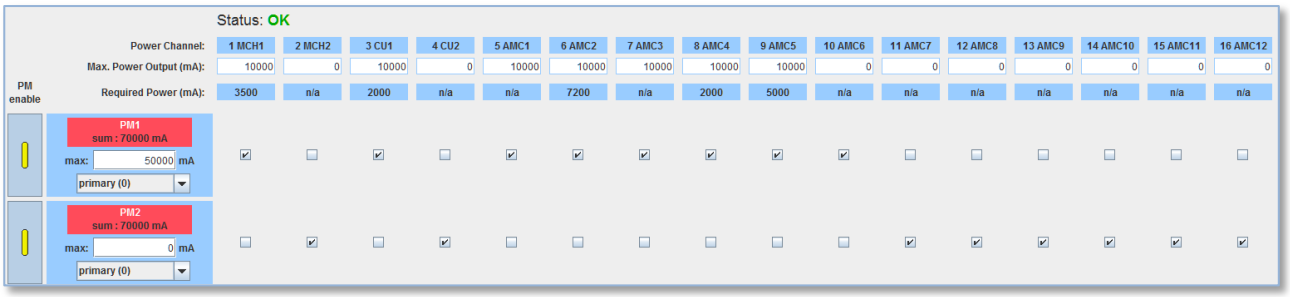
And this is more or less all you need to know when configuring your power setup.

9.1.3 Load Sharing

The typical setup of load sharing is two power modules (PM) that supply two mutually exclusive sets of power consumers. In a 12 slot chassis this might be PM1 for MCH1, CU1, and AMC1 to AMC6 while PM2 supplies MCH2, CU2, and AMC7 to AMC12. (If your chassis is smaller you need to adapt this rule accordingly.)

This configuration allows the usage of more power consuming devices in a system without exceeding power supply limits of the power modules.

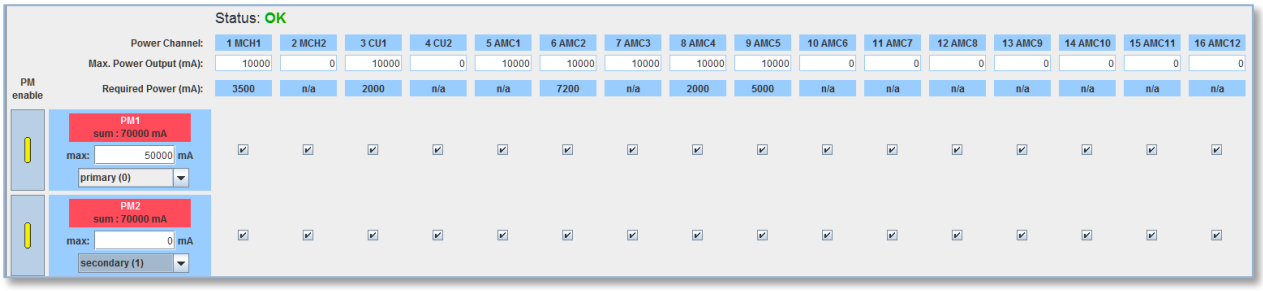
Using the NATview PCM this might look like this:



9.1.4 Full Redundancy

A system running in full redundancy mode both power modules service all power consumers. The primary power module supplies power in the first place. If this primary power module fails for some reason, the secondary power module jumps in and takes over.

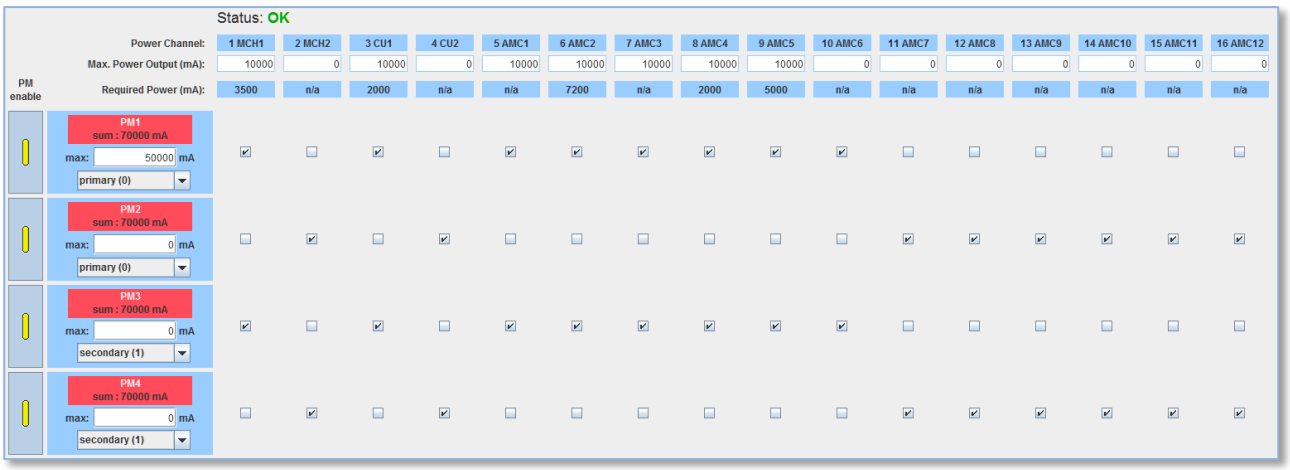
With the PCM this might be configured like this:



9.1.5 Redundant Load Sharing

Redundant load sharing is kind of a mixture of the previous modes: Every power module does not service all of the system. Every power module has a fall-back companion. You may use devices with extended power consumption without losing the power safety.

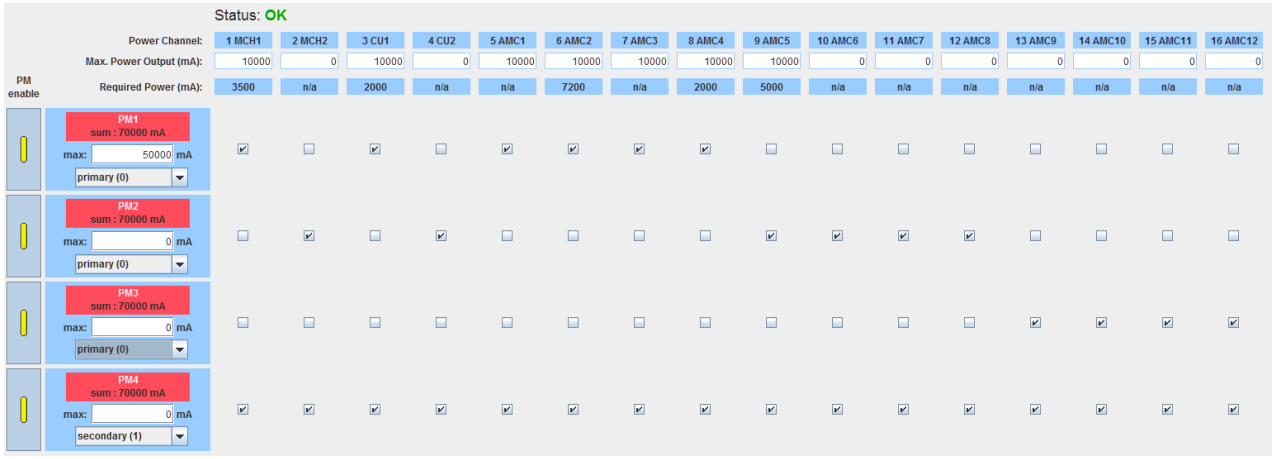
The NATview Power Configuration Manager may look like this:



9.1.6 N+1 Redundancy

N+1 Redundancy could be seen as an extension of Redundant Load Sharing: Three power modules can be used for primary power supply thus delivering more power per channel. If one of the three primary modules fails the fourth secondary one jumps in. The disadvantage of this configuration is that the whole system fails if a second power module fails.

Configure this using PCM like this:

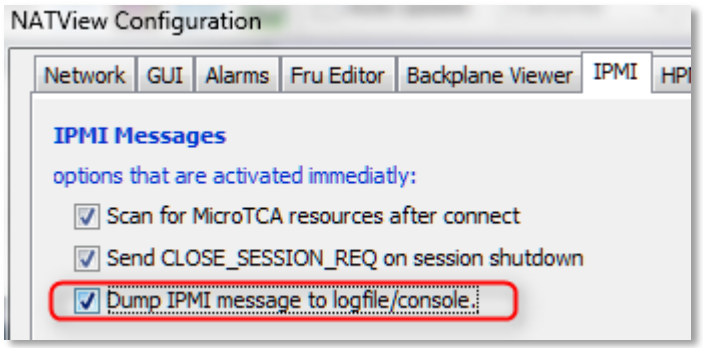

Status: OK
Power Channel:
1 MCH1
2 MCH2
3 CU1
4 CU2
5 AMC1
6 AMC2
7 AMC3
8 AMC4
9 AMC5
10 AMC6
11 AMC7
12 AMC8
13 AMC9
14 AMC10
15 AMC11
16 AMC12
Max. Power Output (mA):
10000
0
10000
0
10000
10000
10000
10000
10000
0
0
0
0
0
0
0
PM enable
Required Power (mA):
3500
n/a
2000
n/a
n/a
7200
n/a
2000
5000
n/a
n/a
n/a
n/a
n/a
n/a
n/a
PM1
sum : 70000 mA
max: 50000 mA
primary (0)
PM2
sum : 70000 mA
max: 0 mA
primary (0)
PM3
sum : 70000 mA
max: 0 mA
primary (0)
PM4
sum : 70000 mA
max: 0 mA
secondary (1)

10 Debugging with NATview

Since NATview Version 2.9 the application can be used to trace the IPMI messages that are being exchanged. In addition it is now possible to control the amount of debug messages that are printed to the console resp. debug file.

10.1 IPMI message tracing

NATview can create an ASCII dump of the exchanged IPMI messages to the console or the debug file. This feature can be enabled in the IPMI configuration:



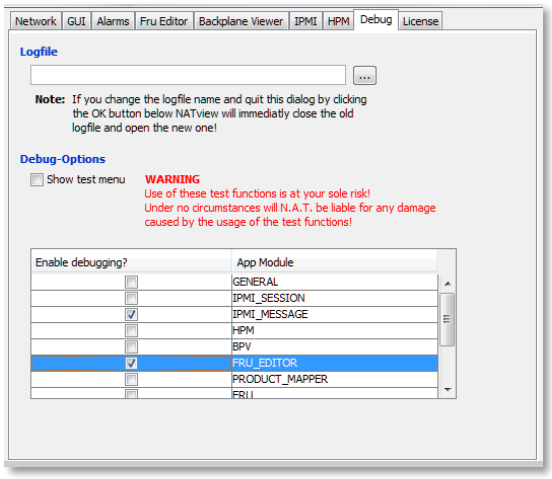
The result may look something like this:

```
<IPMI>[SMS]->[CM] netfn 0x6 [IPMI_NETFN_APP_REQ] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x20 0x18
0xc8 0x81 0xcc 0x34 0x00 0x82 0x28 0x56 0x20 0xcc 0x23 0x15 0x64 0x5a 0x00 0x2d 0x0c 0xe5 0x7f
<IPMI>[SMS]<-[CM] netfn 0x7 [IPMI_NETFN_APP_RSP] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x81 0x1c
0x63 0x20 0xd0 0x34 0x00 0x20 0x2c 0xb4 0x82 0xd0 0x22 0x00 0x16 0x64 0x12 0xdc
<IPMI>[SMS]->[CM] netfn 0x6 [IPMI_NETFN_APP_REQ] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x20 0x18
0xc8 0x81 0xd0 0x34 0x00 0x82 0x28 0x56 0x20 0xd0 0x22 0xee 0x7b
<IPMI>[SMS]<-[CM] netfn 0x7 [IPMI_NETFN_APP_RSP] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x81 0x1c
0x63 0x20 0xd4 0x34 0x00 0x20 0x2c 0xb4 0x82 0xd4 0x23 0x00 0x58 0x00 0x59 0x00 0x51 0x01 0x34
0x50 0xd8
<IPMI>[SMS]->[CM] netfn 0x6 [IPMI_NETFN_APP_REQ] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x20 0x18
0xc8 0x81 0xd4 0x34 0x00 0x82 0x28 0x56 0x20 0xd4 0x23 0x16 0x64 0x59 0x00 0x00 0x05 0x11 0x77
<IPMI>[SMS]<-[CM] netfn 0x7 [IPMI_NETFN_APP_RSP] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x81 0x1c
0x63 0x20 0xd8 0x34 0x00 0x20 0x2c 0xb4 0x82 0xd8 0x23 0x00 0x58 0x00
<IPMI>[SMS]->[CM] netfn 0x6 [IPMI_NETFN_APP_REQ] cmd 0x34 [IPMI_CMD_SEND_MESSAGE]  0x20 0x18
```

Since NATview 2.20 this message dump functionality has been extended in such a way that the contents of bridged messages are dumped and interpreted as well.

10.2 Debug messages

As with almost every application NATview prints additional diagnostic, warning or error messages to the console (or the debug file, if defined). Starting with Version 2.9 this behaviour can be configured at runtime in the **Debug** register of the configuration dialog:



Simply tick the checkbox of the application section from where messages shall be printed to the console. Currently the following sections are selectable:

GENERAL	General Information from the application.
IPMI_SESSION	IPMI session related information.
IPMI_MESSAGE	Diagnostics about transmitted and received IPMI messages.
HPM	Diagnostics from the HPM updater.
BPV	Diagnostics from the backplane connectivity viewer (BPV)
FRU_EDITOR	Diagnostics from the FRU editor.
PRODUCT_MAPPER	Diagnostics from the product mapper module.
FRU	FRU-related messages.
SENSORS	Sensor-related messages.
EVENTS	Diagnostics about the SEL and its events.
PRODUCTION_FRU	Diagnostics about the production FRU module (not yet implemented)
CONFIGURATION	Messages from the configuration database.
EVENTS	System event log (SEL) diagnostic messages (e.g. hot swap).

The selection will be active immediately the configuration dialog has been closed using the **OK** button.

11 Sensor Alarm Notification Indication

A new feature since NATview version 1.21 is the **Sensor Alarm Notification Indication (SANI)**. It is activated whenever a serious event has been reported by a sensor. The SANI is displayed in the upper left corner of an AMC device image. Look at the following image for details:

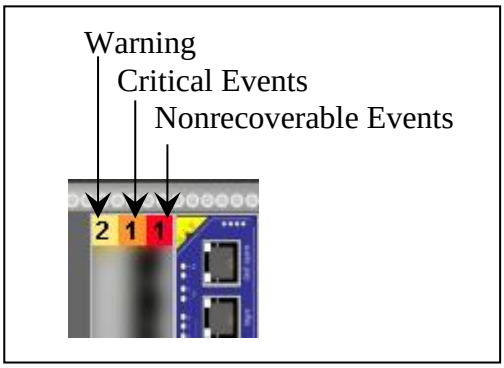


Figure 22: Sensor Alarm Notification Indication

The categorization matches primarily the threshold levels of a temperature sensor: Whenever a temperature sensor crossed a recoverable threshold towards the critical threshold this event is regarded as warning. If this happened from critical in non-recoverable direction this is regarded as a critical event. And if the non-recoverable threshold was crossed it is a non-recoverable event.

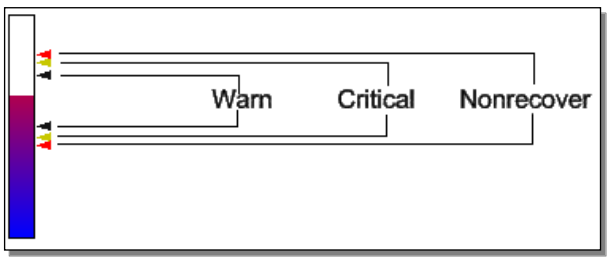


Figure 23: Event categories

The sensor alarm notification indications can be removed using the menu command **Fru→Clear Sensor Alarm Notifications**. (This command is only available if the currently selected FRU device has sensor alarm notifications. Otherwise it is rendered ghosted.)

The SANIs can be configured using the register **Alarm** of the application configuration dialog. It is possible to turn the notifications completely off or to turn off only the counter numbers.

12 Adding Customer Hardware Support

The user has the ability to customize the NATview application so it supports its hardware. He can do this by supplying customer-drawn images for new AMC boards. From version 1.14 on it is possible to supply graphics for customer chassis as well. The application searches for all images and graphic-related files in its *images* subdirectory.

12.1 Recent Changes (NATview 3.x)

NATview 3.00 does not use separate vertical and horizontal AMC graphic files anymore. Instead only the horizontal ("landscape") file is used. It is then rotated to suit the needs for the slot concerned.

To find out the slot orientation NATview uses the slot orientation definitions of the chassis INI file if it exits. (Check section 12.2.1.2.3 for details.) If there is no orientation information in the chassis INI file NATview still uses the Carrier Information Record from the carrier FRU records.

In the end all this information is mapped to a n, w, s, e orientation of the board. This information is then used to rotate the AMC board graphic to be able to show it properly on screen.

Apart from the fact that the creation of a portrait AMC file is obsolete the rest of this chapter still applies.

12.2 AMC Board and Chassis Detection

12.2.1 CCustomer AMC Boards

NATview has been designed from the start to be as extensible as possible. Therefore there had never been such things like hardcoded AMC board resp. chassis graphics. All this data is located in the **`images`** subdirectory. (BTW: The name **`images`** of this subdirectory is indeed hardcoded as we do not see much sense in renaming it.)

Until now everything NATview uses to find the appropriate graphics file for an AMC is the manufacturer ID, the product ID and the orientation of the AMC slots. All necessary details can be found in section 12.2.1.1

This simple search algorithm turned out to be insufficient for some AMC boards, e.g. when an AMC board exists in different sizes: How can NATview determine from the product ID if the user inserted the full height or the half height variant of the AMC board? The answer is: It cannot! There are several other situations where NATview will display the wrong AMC image just because it cannot determine the correct type.

The solution for all of these problems is the new extended AMC detection algorithm that uses a bunch of additional information to provide a better result. Section 12.2.1.2 discusses this new feature in detail.

12.2.1.1 The default search algorithm

For every detected AMC resource NATview tries to find an appropriate image file in its image directory. The image file must be a standard JPEG file, best with all compression disabled. The image filenames must follow the following naming scheme:

<mmmmm><pppp><d>_<descriptive text>.jpg

The filename consists of the following elements:

mmmmm	5 character manufacturer ID. (Note: Previous releases of NATview required a 4 character manufacturer ID. This format is already supported.)
pppp	4 character product ID.
D	1 character direction code: **`p`** = vertically mounted boards (“portrait”) **`l`** = horizontally mounted boards (“landscape”)
_	Underscore character; not interpreted any further.
descriptive_text	This string is not interpreted by NATview any further. Its sole purpose is to make the filenames more descriptive.
.jpg	Standard JPEG image file extension.

The manufacturer- as well as the product-ID is retrieved from the GetDeviceID response of the AMC module.

The user shall always add two versions of board images to the image repository – one for vertically mounted boards, and one for horizontally mounted ones. The image files must have the following dimensions:

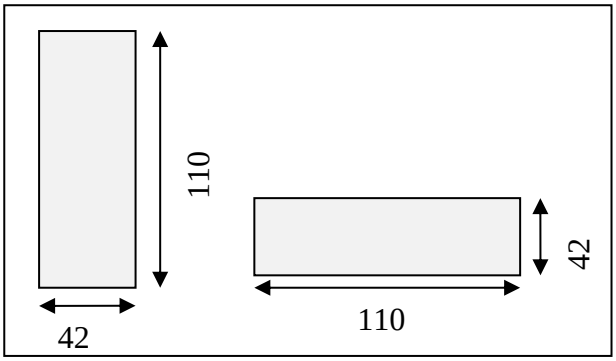


Figure 24: Board Image File Dimensions for a Full-Size, Single Width Module

Depending on the front panel dimensions AMC front panel graphics for NATview must have the following dimensions in pixel:

	Single Width Modules	**Double Width Modules**
Compact-Size (3HP)	21 x 110	21 x 220
Mid-Size (4HP)	28 x 110	28 x 220
Full-Size (6HP)	42 x 110	42 x 220

12.2.1.2 The Extended AMC Detection algorithm

The new **E**xtended **A**MC **D**etection (EAD) tries to solve several problems:

1. Inappropriately sized AMC pictures: a half-height board is displayed for a full-height card in a full-height slot.
2. Picture of the wrong production version/flavour, e.g. the picture of a low-cost module with lesser features is shown while a full-featured card has been installed.
3. A combination of standard components is assembled to a new product; unfortunately only the base component is capable of IPMI communication and is not capable of responding as the new assembled product. (One such an example is the NAT MCH-PHYS which consists of a double width PCB of a base MCH plus the physical clock hub – there is no special product ID for the NAT MCH-PHYS as it is the special combination of components that generates this MCH type.)

The EAD adds the required information to different parts of NATview which will now be discussed in detail.

12.2.1.2.1 Extended file name format

The file name format is extended to store the additional information that is needed for the EAD:

<mmmmm><pppp><d>_<slotdim>_<fruinfo>_<descriptive text>.jpg

The file name components have the following functions:

mmmmm	5 character manufacturer ID. (Note: Previous releases of NATview required a 4 character manufacturer ID. This format is still supported.)
pppp	4 character product ID.
D	1 character direction code: **p** = vertically mounted boards (“portrait”) **l** = horizontally mounted boards (“landscape”)
_	Underscore character; necessary as delimiter.
slotdim	Dimension of the AMC, e.g. SF for single-width, full-height. See section 12.2.1.2.2 for coding details.
_	Underscore character; necessary as delimiter.
fruinfo	This is either the Product Info Area **Product Part/Model Number** or some content from the Procut Info Area **AssetTag**. See section 12.2.1.2.3 for coding details
_	Underscore character; necessary as delimiter.
descriptive_text	This string is not interpreted by NATview any further. Its sole purpose is to make the filenames more descriptive.
.jpg | .png	Standard JPEG or PNG image file extension.

Not all parts need to be present. It is possible to omit parts from right to left. This means that the fruinfo part can be omitted or the slotdim and the fruinfo part. It is not legal to omit the slotdim part only!

The file name format with the optional parts could be written like this:

```
<mmmmm><pppp><d>["_"<slotdim>["_"<fruinfo>["_"<descriptive text>]]]".jpg"
```

Please check also the next two pages for a in depth description of the graphics file name components and a list of valid file names (as some parts are optional while others are mandatory) as well as an overview of the file name sources (where you can see which parts of the file name are checked against which AMC and chassis data).

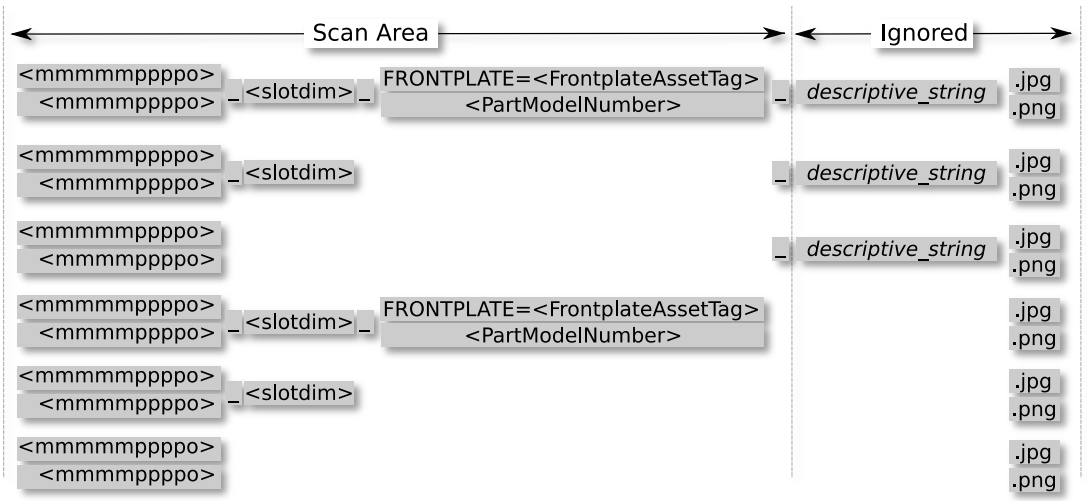


Figure 25: Valid NATview Graphics File Names

NATview Graphics File Name Components

_	A single underscore character that serves a delimiter to separate the three syntactical sections of the file name: **MPO**, **Slotdim**, and **FruInfo**.
<mmmmmppppo> <mmmmmppppo>	The **MPO** section of the filename: 4 or 5 characters for the manufacturer ID, 4 digits for the product ID, and one character for the slot orientation..
<slotdim>	The **Slotdim** section of the filename: SF - single width, full height, SM - single width, medium size, SH - single width, half height, DF - double width, full height, DM - double width, medium size, DH - double width, half size.
FRONTPLATE=<FrontplateAssetTag> <PartModelNumber>	The **FruInfo** section of the filename: this is either the FRONTPLATE tag from the Product Info Area's AssetTag or the Product Info Area Part Model Number. If the board fru info defines both then the FRONTPLATE tag has priority. If the board defines no FRONTPLATE then all files with a FRONTPLATE are skipped.
descriptive text	The descriptive string is simple text that will not be interpreted any further. Simply put any information into this string that is needed to easily identifiy the board.
.jpg .png	File name extension, either for JPEG or PNG files.

Figure 26: NATview Graphics File Name Components

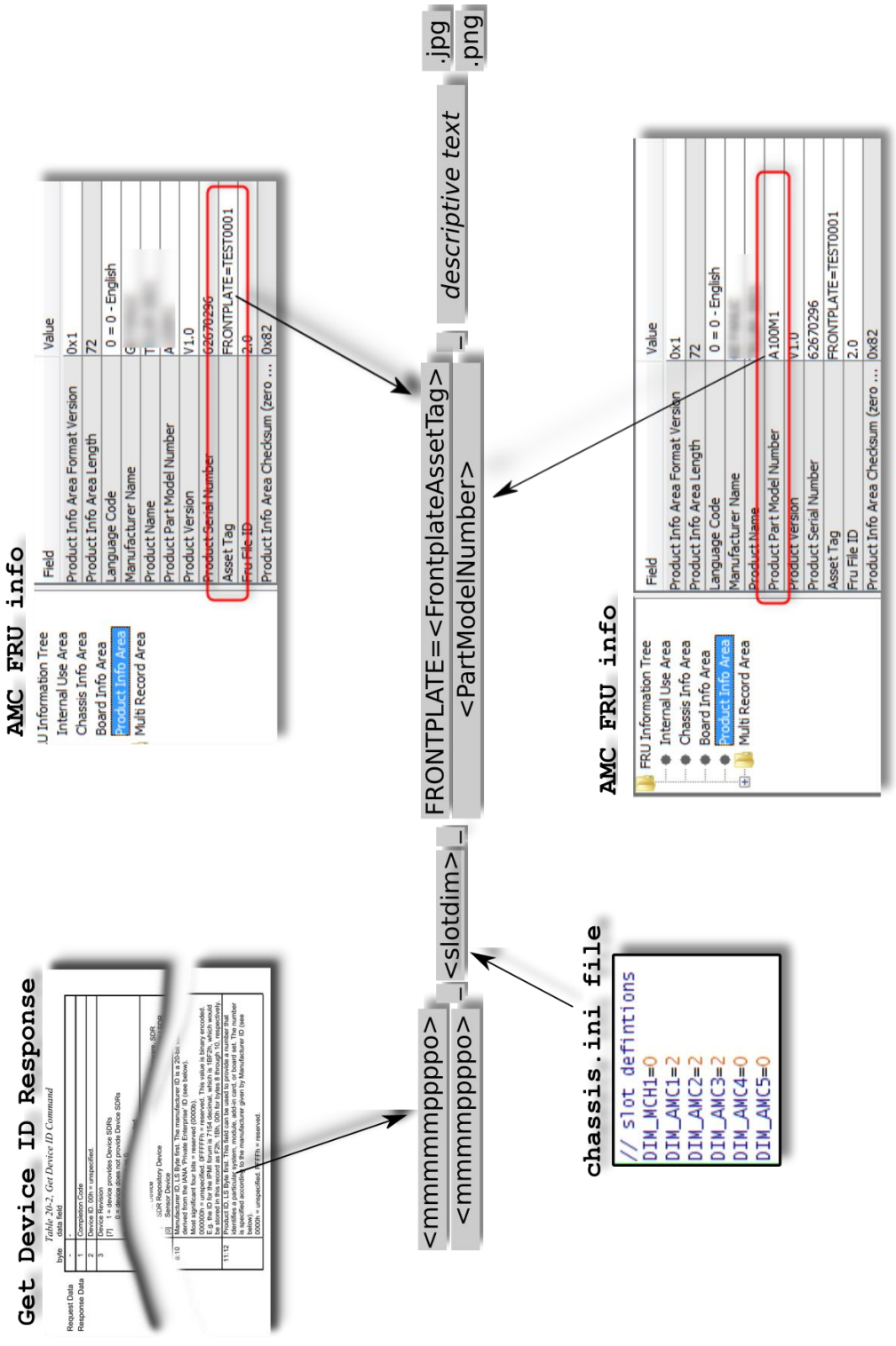


Figure 27: NATview Graphics File Name Sources

12.2.1.2.2 Slot Dimensions in the chassis INF file

MicroTCA provides no means to interrogate the dimensions of an AMC slot. There is a Carrier Information Record that contains the coordinates of the AMC boards in the chassis. Unfortunately this record does not contain information about how wide or height the slot is.

To solve this problem NATview 2.24 introduces some extensions to the chassis INF file that look like this:

```
// SchroffGmbH_rtm9_p.ini
// ----------------------
//

// AMC board image offsets
ORIGIN_X_OFF=7
ORIGIN_Y_OFF=55

SHOW_PU=Y

DIM_MCH1=0
DIM_MCH2=0
DIM_PM1=0
DIM_PM2=0
DIM_PM3=0
DIM_PM4=0
DIM_AMC1=2
DIM_AMC2=2
DIM_AMC3=2
DIM_AMC4=2
DIM_AMC5=2
DIM_AMC6=2
DIM_AMC7=2
DIM_AMC8=2
DIM_AMC9=2
DIM_AMC10=2
DIM_AMC11=2
DIM_AMC12=2
```

The slot dimension coding in the chassis ini file and the graphics file names is listed in the following table:

AMC.0 Dimension Names	MicroTCA Dimension Names	Chassis ini Coding	File name Coding
Single Width, Full Height	Single-Width, Full-Size	0	SF
Single Width, Medium Size	Single-Width, Mid-Size	1	SM
Single Width, Half Height	Single-Width, Compact-Size	2	SH
--- reserved, don't use! ---		**3**	---
Double Width, Full Height	Double-Width, Full-Size	4	DF
Double Width, Medium Size	Double-Width, Mid-Size	5	DM
Double Width, Half Height	Double-Width, Compact-Size	6	DH
--- reserved, don't use! ---		**7**	---

12.2.1.2.3 Slot Orientations in the Chassis INI File

The MicroTCA specification knows only two ways that AMC modules can be mounted to a chassis:

- **Vertical.** In this case the AMC modules are usually mounted side by side in a row. The hot swap LED and its handle are in the bottom right corner of the module.
- **Horizontal.** Here the AMC modules are mounted one above the other; sometimes multiple rows exist. The hot swap LED as well as the hot swap handle are located in the top right corner.

It is a single bit in the Carrier Information Record that is used to distinguish between these two AMC board orientations. If the chassis designer arranged the AMC modules in a different, non-standard manner NATview could not determine the correct orientation.

Therefore the chassis INI file can be extended in such a manner that information about the AMC slot orientation is being coded into the slot dimensions by using one of the additional letters: **`n`**, **`w`**, **`s`**, **`e`** – the top of the AMC shows to one of the four cardinal points, just like on a map.

n	north	The AMC module top points to the north.	
w	west	The AMC module top points to the west.	
s	south	The AMC module top points to the south.	
e	east	The AMC module top points to the east.	

A chassis INI file may look like this:

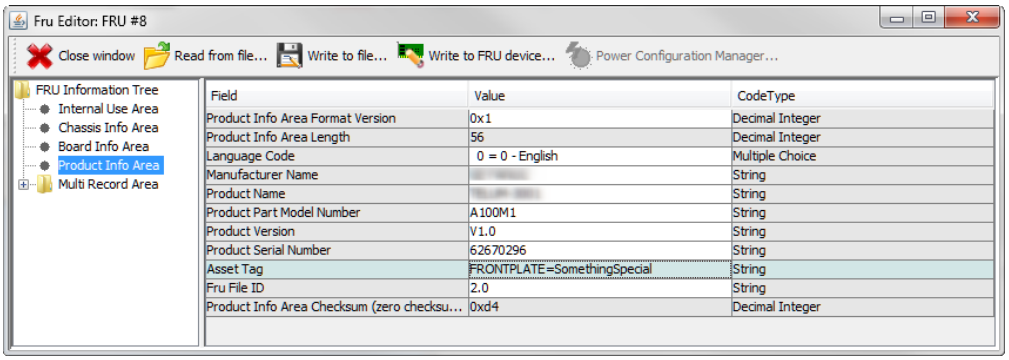
```
// SchroffGmbH_11890-179.ini
// -------------------------
//

// AMC board image offsets
ORIGIN_X_OFF=20
ORIGIN_Y_OFF=110

SHOW_PU=Y

// slot defintions
DIM_MCH1=4n
DIM_MCH2=4n
DIM_PU1=4n
DIM_PU2=4n
DIM_PU3=4n
DIM_PU4=4n
DIM_AMC1=6n
DIM_AMC2=6n
DIM_AMC3=6n
DIM_AMC4=6n
DIM_AMC5=6n
DIM_AMC6=6n
DIM_AMC7=6n
DIM_AMC8=6n
DIM_AMC9=6n
DIM_AMC10=6n
DIM_AMC11=6n
DIM_AMC12=6n
```

12.2.1.2.4 FruInfo

The fruinfo part of the filename consists either of the **Product Part/Model Number** or of some content of the **Asset Tag** field. If the AMCs fru info defines a value for the AssetTag then the Product Part/Model Number data is ignored.

The Product Part/Model Number is usually the product part number or the model number of the board.

The contents of the asset tag are not further specified by the FRU Info specification. This documentation suggests using it like this:



As this AMC defines the string **FRONTPLATE=SomethingSpecial** in its Fru Info NATview sould try to find a file name of the format

<mmmmm><pppp><d>"_"<slotdim>"_FRONTPLATE=SomethingSpecial.jpg"

The **slotdim** should match the size of the slot the AMC was installed. If there is no matching file name NATview tries all other slot dimension codes until it finds a match.

Note: If the AMCs fru info does NOT contain valid data in the asset tag then all files with **FRONTPLATE** in its name are skipped and ignored! That way you can have special marked boards and general ones without getting a file name mix-up.

12.2.1.2.5 Auto dimension negotiation (ADN)

The Extended AMC Detection introduces also a new mechanism that shall improve that the best fitting board graphic is being used. Simply spoken it works like this:

- When an AMC is installed to a full-height slot NATview will display the full-height graphic version (if it exists).
- When the same AMC is being moved to a half-height slot NATview will show the half-height graphic version (if it exists).

ADN is active whenever there is no contrary information in the AMC FRU records which means that if there is a FRONTPLATE tag it will overrule the AND mechanism. The same applies when the full-height and the half-height AMC have different product part/model numbers.

12.2.2 Customer MicroTCA Chassis

When NATview scans the MicroTCA system for its resources it starts by inquiring the backplane eeprom which is done by reading the data of FRU device 253. The chassis type, the orientation of the AMC boards and their location in the chassis are retrieved from the backplane FRU information.

Starting with NATview 2.7 this mechanism has been extended while the "old fashioned way" to detect the images is still alive. To find the correct chassis image NATview first uses the new extended algorithm. If it fails then the old fashioned algorithm will be used.

12.2.2.1 The "old fashioned" way

The following table lists the information sources of the backplane FRU and how each of them is used:

Manufacturer Name, Product Name	The information from these records is used to construct the filename of the chassis graphic and the corresponding inifile. All space characters are removed then both of them are concatenated in the following manner: <Manufacturer Name>_<Product Name>_<Chassis Orientation>.jpg - or - <Manufacturer Name>_<Product Name>_<Chassis Orientation>.ini The chassis orientation is coded in the same way as for the AMC board image filenames (see above for details). **The chassis image file size must provide enough space for placing the supported AMC board images.**
Carrier Information Record	This record contains information about which resources are supported by the chassis and where they are located. The positioning of the AMC board images on the chassis graphic can be tuned with the ORIGIN_X_OFF and the ORIGIN_Y_OFF keys in the chassis inifile.

The chassis inifile supports the following keys:

ORIGIN_X_OFF	Integer	Positive or negative offset in pixel that is added to the calculated X coordinate of an AMC image. It is used to compensate the chassis dependent positioning deviation.
ORIGIN_Y_OFF	Integer	Positive or negative offset in pixel that is added to the calculated Y coordinate of an AMC image. It is used to compensate the chassis dependent positioning deviation.
SHOW_CU	Y, N	If Y then a detected cooling unit will be displayed in the rack pane if an appropriate image file is found.
SHOW_PU	Y, N	If Y then a detected power unit will be displayed in the rack pane if an appropriate image file is found.

12.2.2.2 The (new) extended chassis detection

The manufacturer and product name of the board- resp. product info area are not always helpful in detecting the correct image file. Therefore the chassis detection mechanism has been extended.

The following combinations of data fields from the Product Info Area (PIA) resp. Board Info Area (BIA) are tried:

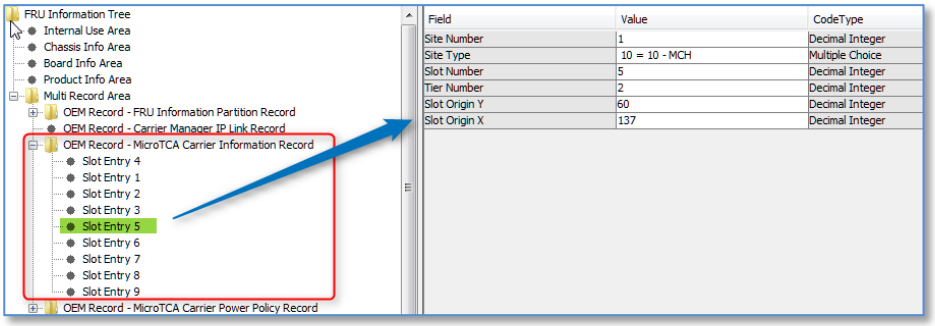
1	PIA Manufacturer Name	"_"	PIA Part Model Number	"_"	PIA Product Version

2	PIA Manufacturer Name	"_"	PIA Part Model Number	-	-
3	PIA Manufacturer Name	"_"	PIA Product Name	"_"	PIA Product Version
4	PIA Manufacturer Name	"_"	PIA Product Name	"_"	orientation character
5	BIA Manufacturer Name	"_"	BIA Part Number	"_"	BIA Serial Number
6	BIA Manufacturer Name	"_"	BIA Part Number	-	-
7	BIA Manufacturer Name	"_"	BIA Product Name	"_"	BIA Product Name
8	BIA Manufacturer Name	"_"	BIA Product Name	"_"	orientation character
9	"default_default_	"_"	orientation char	-	-

Please note that the chassis orientation is in general not relevant for this algorithm anymore, except for the cases where an orientation character is used.

12.3 AMC Coordinate Handling

To be able to position the AMC board images properly onto the chassis image NATview needs some information: the *MircoTCA Carrier Information Record(s)*. Every chassis carrier FRU must have at least one of these records – if the chassis is big and has a lot of slots more than one of these records could be required.[10]



[10] As every FRU record's record length field is 8 bits wide one record (with its record header) cannot be longer than 255 bytes.

The type and position of every slot is described by a *Slot Entry* which looks like this:

Site Number	This is the number of the device of this site type.
Site Type	Describes what type of device the slot contains: Cooling Unit (4), Fan (5), Alarm (6), AMC (7), PMC (8), RTM (9), MCH (10), Power Module (11).
Slot Number	Slot Number and Tier Number are kind of a slot index. In systems with vertically oriented AMC modules the slots increase from left to right while the tiers increase from bottom to top. If the AMC modules are horizontally oriented this is exchanged. As the AMC module alignment is rarely like on a chessboard these values are of limited use.
Tier Number	
Slot Origin Y	see diagram
Slot Origin X	

Depending on the coordinate scheme that is being used the processing of the two parameter *Slot Origin Y* and *Slot Origin X* can differ for chassis with horizontally mounted AMCs. Currently there exist two processing schemes which will be discussed in more detail in the following sections:

(1) The default coordinate scheme

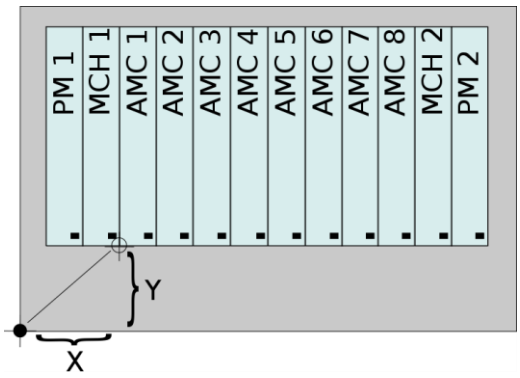
(2) The Schroff Horizontal Mode

Chassis with vertically mounted AMCs always use the default coordinate scheme.

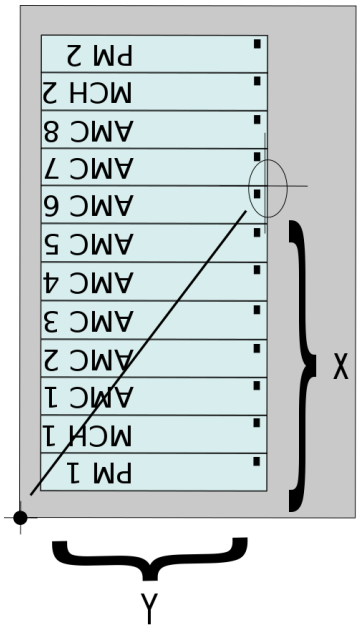
12.3.1 Default coordinate scheme

Slot Origin Y and *Slot Origin X* are measured in Millimeters, starting from the *lower left* corner of the chassis to the *corner with the hot swap handle* of the AMC.

For chassis with vertically oriented AMC modules this looks like this:



For chassis with horizontally oriented AMC modules the situation is a bit more complex. The most common interpretation of the specification would look like this:

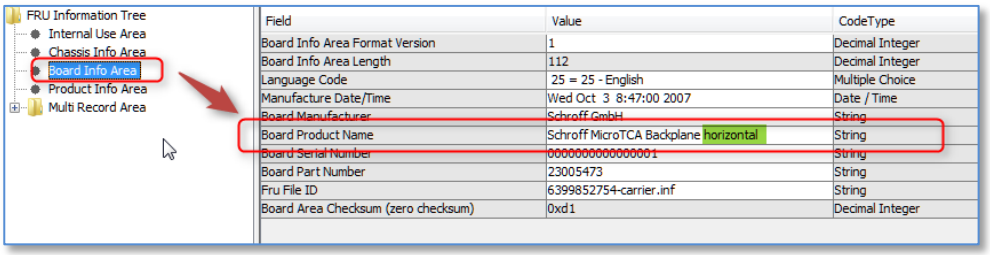


Chassis that follow these coordinate schemes should work more or less out of the box. It may only be necessary to adjust the offsets in the ini-file of the chassis.

12.3.2 Schroff Horizontal Mode

The German chassis manufacturer Schroff likes to code the coordinates differently. In accordance with N.A.T. both companies agreed that NATview shall support this mode.

To enable the *Schroff Horizontal Mode* the field **Board Product Name** of the Board Info Area must contain the string **`horizontal`**.

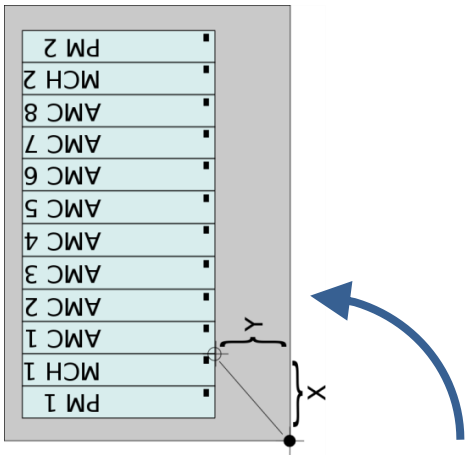


Schroff distinguishes between the Shelf and the Carrier:

- The Carrier is the inner part of the chassis that carries the AMC modules. It also contains the chassis origin.

- The Shelf is the outer part that holds the Carrier.

If you follow this logic then a chassis with horizontal oriented AMC modules is simply a Carrier that has been rotated 90 degrees counterclockwise while the Shelf is fixed. As the Carrier contains the chassis origin for the coordinates, it is rotated, too – leading to this result:



12.4 The Product Mapper - support for special devices

Note: The mechanism described in this chapter should be obsolete thanks to the extended AMC detection (see section 12.2.1.2 for details).

Starting with version 2.7 NATview is able to handle special devices like the NMCH-PHYS.

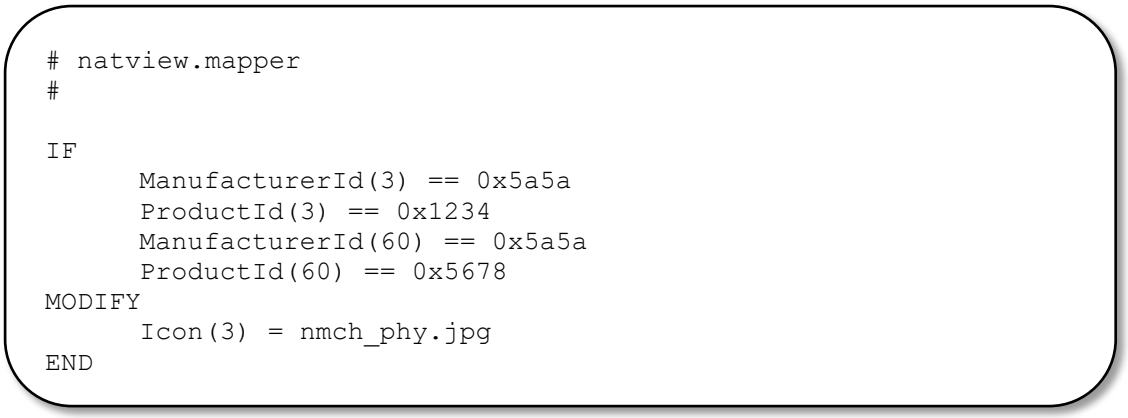
What does this mean?

Devices like the NMCH-PHYS have no special product ID but can be characterized by their components. A NMCH-PHYS is defined to consist of the following components:

- NMCH Base plus
- NAT-CLK-PHYS plus
- NAT-HUB-PCIe-48x

This special MCH is delivered as a double-width, full-height AMC which should be displayed differently in NATview. To achieve this the icon of the MCH should be exchanged whenever this special combination of components has been detected.

This job is performed by the **Product Mapper**. This module of NATview is configurable so new special devices can easily be added. The Product Mapper finds all necessary information in a text file named **`natview.mapper`** which looks like this:



```
# natview.mapper
#

IF
      ManufacturerId(3) == 0x5a5a
      ProductId(3) == 0x1234
      ManufacturerId(60) == 0x5a5a
      ProductId(60) == 0x5678
MODIFY
      Icon(3) = nmch_phy.jpg
END
```

The syntax of **`natview.mapper`** is quite simple. It consists of one or more entity blocks, like the one displayed above. Every entity block consists of an IF and a MODIFY section. The entity block is terminated by the keyword END. The commands in the MODIFY section are only executed if **all** conditions in the IF sections are true.

Blank lines as well as lines that start with a **#** are ignored.

The IF section can contain one or more of the following system calls:

`ManufacturerId(fruId) == <id_value>`	Checks if the manufacturer ID of FRU fruId equals id_value.
`ProductId(fruId) == <id_value>`	Checks if the product ID of FRU fruId equals id_value.

The MODIFY section can contain one or more of the following system calls:

`Icon(fruId) == <icon_filename>`	Changes the icon for FRU fruId to the content of icon_filename. The icon is being searched in the default images directory!

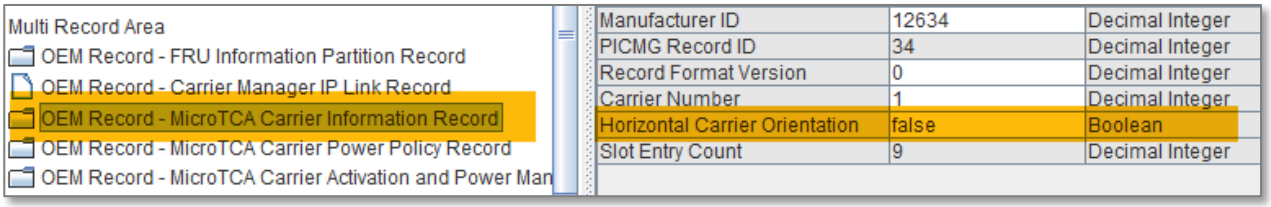
13 FAQs and Troubleshooting

13.1 My chassis is not being displayed. All I see is a default chassis.

Whenever NATview scans a system it needs to find out the name of the chassis graphic file. All necessary information must be taken from the backplane FRU (FRU 253). As there are unfortunately no manufacturer- and product-ID NATview needs to use the manufacturer and product name to construct the file name. The algorithm is like this:

1. Scan the Board Info Area. If it exists and the Board Manufacturer and Board Product Name are not empty use them as manufacturer and product name.
2. If nothing has been found yet scan the Product Info Area and use the Manufacturer and the Product Name.

Another important information is the orientation of the AMC boards in the chassis. This information is coded in the MicroTCA Carrier Information Record (see image below).



It is very important that the board orientation is coded correctly as it is a part of the chassis graphic file name! If your chassis is not being displayed although the chassis graphic file exists check the board orientation!

The complete chassis graphic file name is assembled like this:

```
<Manufacturer Name>_<Product Name>_<Chassis Orientation>.jpg
```

The correspondent INI-File is assembled likewise:

```
<Manufacturer Name>_<Product Name>_<Chassis Orientation>.ini
```

(Carefully read chapter 12.2.2 as well, please.)

13.2 I cannot connect to my chassis (although I see it in the MCH scanner)

NATview communicates with the Shelf Manager (usually your MCH) using IPMI messages that are packaged into RMCP frames. These RMCP frames are basically UDP frames being sent on port 623.

If you cannot connect to your system one possible cause for this might be a firewall, either on the PC running NATview or between the MicroTCA system and the PC. If your firewall has some kind of application control then either NATview or the Java Virtual Machine must be able to send requests through the firewall.

13.3 NATview does not show the proper hot swap state of my boards

One of the reasons why NATview does not display the correct hot swap state (a.k.a *M-state*) of the boards in a system might be that it uses the wrong hot swap sensor type.

Please check chapter 3.4 "Checking the correct hot swap configuration" for details.

13.4 The Chassis is ok but the AMC boards orientation is wrong

Check if the chassis INI file is set up accordingly. See section 12.2.1.2.3 "Slot Orientations in the Chassis INI File" on page 92 for details.

14 Release History

This chapter lists the new and changed features of the appropriate releases starting with version 1.17. Previous releases are not covered in this document.

14.1 Version 1.17

Fixed carrier info record handling of the fru editor.

Added default offsets being used in conjunction with the default chassis;
should prevent module image misplacement.

New connect dialog that initially hides the scan options.

Fru editor now saved the last read- and write-directory.

Chassis backgound color is now configurable.

Extended log file support.

Added support for floating point sensors.

In case that the module graphic file cannot be found the module manufacturer- and product-name is being written onto a default AMC module image.

Changed horizontal and vertical alignment of the AMC module images; this should allow proper handling of double width and half-height modules.

Checks carrier manager's firmware version to chooses proper resource scanning algorithm.

Displays the FRU firmware version (as being retrieved from the GetDeviceId response).

14.2 Version 1.18 (Internal Release)

Added check for session ID; now drops incoming message for another session.

Saves the last event messages to the appropriate fru object so it can be displayed when the mouse hovers over the fru device image (only in registered version).

Retrieve the initial AMC module state in case that the MCH does not send activation event messages.

14.3 Version 1.19 (Internal Release)

Does not enable sensors as they should be enabled by default. This fixes an error which disabled all sensors.

Prepare Sdr class with new fields mappedNumber and mappedLun to store the MCH-mapped. Unfortunatly IPMI does not define a clean way to map from local sensor number / LUN to MCH sensor number / LUN.

Fixed offset for Maximum Current Output as there had been an illegal bias of 5 (method PowerModuleCapRecord() of the FruEditor).

Rewrote session shutdown to avoid invalid messages during shutdown.

Subtract full chassis height in any case when calculating ypos in landscape mode; this fixes a misplacement with non-full-height AMC boards.

Trigger an insertion event as long as the fru is invalid (=not used); this should fix the now missing `M0->M1` transition event.

14.4 Version 1.20 (Internal Release)

New event viewer (registered version only).

Print application name, version and timestamp at the beginning of the logfile as well as the ini-file.

Rewrote range calculation of Meter control (which is used to display the temperature and discrete values).

Configuration database now supports key removal if the key value is null.

Rewrote resource tree handling code to make it Swing thread safe.

14.5 Version 1.21

Added sensor alarm notification indicator.

Extended license key support.

Fixed decoding of firmware version in several places; now the minor version number should be correct.

New application info dialog with new application logo.

Event viewer can now save the events to an ASCII- or HTML-file (only registered version).

License type of the application now determines the type of the event viewer: only the registered version can use the full blown event viewer (with filtering and saving).

Changed CU control to spinner control as this is more mouse-compatible.

Changed fill color of rectangle from red (200,0,0) to blue (0,0,200).

14.6 Version 1.22

Switchable resource tree icons.

Various modifications to reflect the new NATview writing.

Event filter: New filter clearance functionality support;

Meter control: quick hack to prevent overflows.

Support for electronica's NAT show controller.

Workaround to speed-up shutdown.

14.7 Version 1.23

New renderer for ResourceTree.

FruEditor.java: FruInfoData: added new field parent for storing a reference to a superordinate entry; finished support for P2P connectivity record;

FruEditor.java: Completed reformatting and translation of the code.

14.8 Version 1.24 (Internal release)

Internal check-in release which has never been published.

Code and comment clean up.

Converted (existing) class Meter to a Java Bean.

14.9 Version 1.25

Changed behavior of Fru device blinking which led to an infinite blinking of the Fru device image if the device had no sensors or did not response to the SDR requests.

Check completion code of the embedded IPMI message; drop it if completion code is not 0.

Removed FRU ID 252 from being scanned.

14.10 Version 1.30

NATview version 1.30 implements support for a separate shelf- and carrier manager. Although not tested it should work with all IPMI-compliant MCH devices. The changes in detail are:

Support for separate shelf- and carrier manager. Initial resource recovery is now done by scanning the shelf manager repository. Resource changes are detected by periodical scans of the carrier manager repository (once every 5 seconds).

Adapted the event viewer and the event filter to work with the new resource management.

Reworked the session connection and resource scanning mechanism.

Reworked the session disconnection (includes several minor bug fixes).

14.11 Version 1.31

- Fixed some bugs in the SDR handling.

14.12 Version 1.32

- New configuration keys SEND_POLL_MESSAGES and SCAN_CARRIER_MANAGER.

- Fixed bug in SDR end flag detection (check for -1 must be a check for 0xffff instead).

14.13 Version 1.33

- setBoardLabel now retrieves the FRU ID on its own.
- Fix in hotswap handling which lead to missing sensors.

14.14 Version 1.34

- Fix in hotswap handling: when reaching M4 it is checked if all resources have already been collected; if not this is done with this transition.

14.15 Version 1.35

Added the event logfile feature which allows saving of all SEL events to a textfile for later processing. The logfile path/name can be specified in the configuration as well as the data column separator character.

14.16 Version 1.36 (Internal release)

- Added support for Backplane Connectivity Viewer.
- Integrated new FRU editor and added support for multiple FRU editors.

14.17 Version 1.37

- Finished integration of Backplane Connectivity Viewer.
- Finished first release of the new FRU editor.
- Changed RMCP timeout from 5000 to 1000 milliseconds to speed up the system online status scan (of the Connection dialog).
- Added Shelf Manager backplane FRU scan.

14.18 Version 1.38 (Internal release)

This version contained various bug fixes and was released for testing purpose only.

14.19 Version 1.39

This version contains the following fixes and changes:

- The application name was changed to **NATview**. Adapted documentation and application accordingly.
- Fixed graphic garbage of the temperature / voltage meter in the detail pane. (Drawing of the meter scale happened several times with different scale factors making the meter scale completely unreadable.)
- Fixes, modifications and extensions of the Backplane Viewer to support more chassis types and more backplane connectivity features.
- The FRU Editor now supports enumerations with its new multiple choice data type: Valid data values can now be chose from a list. Data is displayed in a more descriptive manner.

14.20 Version 1.40 (Internal release)

This version contained various bug fixes. It also implemented some features that are about to be used in later versions by the license key management. It has never been officially released.

14.21 Version 1.41 (Internal release)

This internal releases only purpose was to adjust the version number of some files that had been moved into separate packages. In Java a package is always represented by a subdirectory. The currently used revision control system does not allow to rename or move files without deleteing them in their old and adding them from their new location. But newly added files always get release number 1.1. Therefore a separate check-out-check-in cycle was necessary to update their release numbers accordingly.

14.22 Version 2.0

This NATview version introduces – among other fixes – the completely new written and newly designed Backplane Connectivity Viewer (BPV).

14.23 Version 2.1

Bug-fix release that has never has been released officially.

14.24 Version 2.2

Officially released on June, 1st 2011, this version includes the following new features:

1. Backplane Viewer (bug-fixes)
2. MCH Scanner
3. System Dump.

14.25 Version 2.3

Reorganized the tools Event Viewer, Backplane Viewer and System Dumb into a new Tools menu. This menu is only enabled for NATview Professional.

14.26 Version 2.4

Invented the new sensor history and auto update option.

Implemented several bug fixes, e.g. in the FRU editor and the AMC filenames (supports now 5-character manufacturer IDs).

14.27 Version 2.5

New in the FRU editor: The Power Channel ID is now not only displayed as an integer value but also shows its meaning; also added Rear Transition Modules in the Carrier Information Record.

Extended hot swap event handling for NAMC-DC780, especially in the Backplane Viewer.

General code cleanup for SDR-related classes.

Added Support for NAMC-LM.

14.28 Version 2.6

Extended support for NAMC-LM: now it is possible to configure the new zone mode which can be used to do temperature control test.

Also fixed some bugs related to the license feature checking as well as some bugs in the FRU editor.

14.29 Version 2.7

- Supports HPM firmware update so all supported FRUs can be updated remotely.
- Trial license key request command to create a request file for a trial license key.
- Product Mapper for support of special devices (e.g NMCH-PHYS).

- MTCA.4 Support.
- Colored threshold labels.
- Fru editor support for Internal Use Area with the new hexeditor.
- AMC13 support.
- Various bug fixes.

14.30 Version 2.8

Internal release.

14.31 Version 2.9

New features:

- Reworking of HPM GUI. Better error handling. Support for multiple card updates.
- Info dialog and main window now show the build date of the jar-File (which more or less corresponds with the project date).
- Support for shelf manager option "send SEND_MSG confirmation to SMS".
- Support for carrier ID's > 1.
- IPMI frame dump.
- Dynamically switchable debugs.
- Optional clearing of MCH SEL after the system has been scanned to avoid processing of old events.

Bugfixes:

- Default carrier: moved PUs to the outside and the MCHs to the inner side to get a symmetrical view.
- Version number handling: fixed handling of empty appendices.
- Sensor update: automatic update of the detail pane if a sensor is currently being displayed.
- HPM Update: check if the currently removed device is one that is about to be debugged.
- Meter scaling

14.32 Version 2.10

- Configuration dialog, register license: License holder field now only accepts letters A to Z and digits 0 to 9 plus the blank character. This shall avoid problems that occurred when transferring license key between different platforms. In one case the license key for a user did not work because of the differences in upper character coding of Windows and Linux.
- New IPMI option IGNORE_SOME_ERRORS that is used to ignore some less severe errors when writing FRU info data to devices.
- Removed support for the so-called old FRU editor.

14.33 Version 2.11

- Finished reworking of the meter scale (temperature and voltage).
- Added support for application function modules.
- Added workaround for the first tested address in the Connection dialog by polling it twice whenever this system is detected as being offline.

14.34 Version 2.12

Internal release.

14.35 Version 2.13

- Release of the FRU editor extension **Power Manager**.
- Fix of a CRC error bug when writing FRU data directly to a device after parts of the Product Info, Board Info and/or Chassis Info Area had been changed. When this error occurred the MCH rejected all backplane info data and used a default chassis instead.
- Various small bug fixes.
- Several tiny extensions, like automatically adding a new system's IP address to the system configuration list of MTCA systems.

14.36 Version 2.14

Internal release.

14.37 Version 2.15

Reworked the Power Configuration Manager (formerly known as Power Manager). Also fixed a bug in the chassis graphics file search algorithm. This version was never released.

14.38 Version 2.16

Fixed a tiny bug in the configuration settings of the Power Configuration Manager that enabled the configuration checking. This bug caused the checking to be permanently disabled.

14.39 Version 2.17

This version contains lots of bug fixes in session management and event processing.

It introduces the Carrier- and Shelf Manager SDR Viewer and the Fru File Cutter.

14.40 Version 2.18

- Positioning of main window: Now checks it the resulting window is visible under the configuration that is currently used. If not the coordinates are corrected. Up to now it was possible that the windows were invisible because a NATview installation was being used with a two-monitor-setup before.
- AmcP2pLinkInfo / Backplane Connectivity Viewer: Fixed handling of descriptor channel ID; now matches what was intended by the AMC spec. Can now handle mor that one AMC P2P connectivity records per FRU.
- HPM: Fixed handling of extended IPMI_CMD_UPLOAD_FIRMWARE_BLOCK_RSP with new offset/length; missing functionality could lead to breakups when updating. also added Skip-Functionality for HPMs UploadFirmwareBlock.
- Now better use of debug source HPM so user can turn on/off HPM debugs more precisly.
- Changed event handling for hotswap M4->M1 which could lead to a strange behaviour seen on some boards on hot swap.
- FruEditor.java: Support for new date/time editor. Support/fix for data type GUID. Several tiny fixes, e.g. data mode definitions and descriptions.
- Support/workarounds for buggy backplane FRUs.
- Always enable the system dump when there is an active connection.

14.41 Version 2.19

- Started to add drag'n'drop support in the Fru editor. Please do not use as it is not completely implemented. (You will lose no data but all drag'n'drop operations will be ignored when the data is being written to storage or device.)
- Support for new Schroff's coordinate system for horizontal carriers.
- HPM Update: Added progress bar and timeout for preparation stage; also better handling for extreme long initialization phases.
- Added support for another 4 power modules (makes 8 in total).
- Several bug fixes.

14.42 Version 2.20

- Fixed bug with bigger chassis and RTM-modules that led to a crash of the system scanning thread.
- Fru Editor: Fixed in the read processing of the Board Info Area's manufacturing date causing invalid date/time values.
- Fru Editor: Now saves the read FRU file and write FRU file directory path separately.

14.43 Version 2.21

- Implements a hex editor for the Fru Editor. It can be used to edit the *Internal Use Area* as well as unsupported record types. The hex editor can load a byte buffer, operate on it (modify, extend, shorten) and store it back for further processing. The mode of operation is intentionally different from what is known from a standard text editor. The buffer must be explicitly extended or shortened if the current length is insufficient. New bytes will not be appended automatically to its buffer.
  KNOWN LIMITATIONS: Currently the hex editor cannot insert bytes. It also does not support Copy&Paste as well as Drag&Drop.
- This modification disables the default JTree behavior that opens ("expands") a tree branch when a tree node has been double-clicked. This interfers with the double-click action for opening the FRU editor. (It works but the tree branch is opened as well which is somehow confusing.)
  Opening a tree branch now works exclusively by clicking on the cross or arrow of the tree node.
- NatViewConfigurationDlg.java: Minor changes in Debug configuration table.

- FRU editor: Reworking of drag & drop support.
  Fru Editor now allows the creation of more records, e.g. Carrier Information Record. To allow the user to consolidate these records - old and new - together, drag and drop support has been extended. It is now possible to move all records around in its own level. It is not possible to change the hierarchical level o f a record.
- Added new debug flag DRAG'N'DROP for Fru Editor Drag'n'drop debugging.

14.44 Version 2.22

- Reworked Threshold editor: apart from visual changes the editor now reports threshold write errors. It is now checked if the thresholds are writable – only then the edit button is enabled (as editing without writing the changes back makes no sense).
- FullSensorRecord and CompactSensorRecord now support the concept of the modifier unit mode.
- The ShelfManager/CarrierManager Repository Viewer now has primitive search capabilities.
- Bug fixes in request message de-queuing.
- License owner supports now all valid ASCII characters.
- More generous handling of missing Get SEL Info Response messages: no immediate disconnection if up to two responses are missing.

14.45 Version 2.23

- HPM timeout overrule to enable updating HPM targets with an invalid upgrade timeout (too short).
- Several bug fixes.

14.46 Version 2.24

- Extended AMC detection.
- Minor bug fixes.
- Fixed bug in sensor history viewer that made it stop working after 100 events.
- Made sensor history viewer configurable in such a way that the absolute number of values in the history can now be limited.

- Fixed bug in AMC M-State detection.

14.47 Version 2.25

- Support for the new hot swap sensor type 0xf0.
- Various bug fixes.
- Online help system.

14.48 Version 2.26

- The hot swap sensor configuration that had been invented in release 2.25 has been taken out again as this kind of configuration requires an in-depth knowledge of IPMI, AMCs, and MicroTCA that cannot be expected from the end user.
  The hot swap handler code in release 2.26 should be able to handle all MCH firmware releases.
  It is accepted that MCH firmware release 2.17.13 and 2.17.14 might cause some hot swap problems.
  All such related configuration stuff has been removed.

- Removed application version and release date from title bar.

- Changed behavior of the MCH scanner Start-Button: it is now simply disabled; action cancelling is not possible anymore. This has been done as there had been crashes on the Linux platform after the Stop-Button had been clicked. The application simply stopped responding.

14.49 Version 2.27

- Tag version of Version 2.27.

14.50 Version 3.00

- Complete reworking of the image management: The AMC images can now not only be horizontally or vertically aligned, but also upside down. As NATview rotates the images as needed only a single AMC graphic file is needed for one AMC board (instead of the former two).

- Fixed a hot swap error that made NATview display a hot swap inserted AMC module with pulled hot swap handle as being in state M4.

- Extended help viewer (more topics, some fixes)
- Rewrote FRU editor and power configurator interface and behavior to avoid impression that the power configurator directly writes to device/file.
- Checks for updates of the application.
- NatViewAppInfo: changed version from 3.00 RC to 3.00.
- Toolbox: Fixed decoding the sensor name from ASCII to Java-Unicode.
  (Fixed an error at customer where the degree sign was not correctly displayed.)
- ResourceTree: handleMouseClicked(): added check for Fru editor license in case of double-click.
- Lots of bug fixes.

15 Known Limitations

This is a list of all limitations that are known to N.A.T. at the time of this document's release. It shall be understood that this list can never be complete.

- First of all: This version of NATview does not support the NMCH compatibility mode anymore!
- Although this is no bug of NATview it shall be mentioned: Using PNG graphic files with transparent areas may cause graphic garbage in the chassis pane when Aero is being used (Windows 7 / Vista). Do not use PNGs with transparent areas or switch off Aero. (This might be fixed with Java 7 in 2011.)
- Currently the hex editor cannot insert bytes. It also does not support Copy&Paste as well as Drag&Drop.