DECT Messenger Installation and Commissioning — Book 1
Avaya Communication Server 1000

7.5
NN43120-301, 03.02
March 2012

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

“Documentation” means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya generally makes available to users of its products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on its Hardware and Software (“Product(s)”). Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language, as well as information regarding support for this Product while under warranty is available to Avaya customers and other parties through the Avaya Support Web site: http://support.avaya.com. Please note that if you acquired the Product(s) from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/ ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE (“AVAYA”).

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, or Hardware provided by Avaya. All content on this site, the documentation and the Product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Third-party components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements (“Third Party Components”), which may contain terms that expand or limit rights to use certain portions of the Product (“Third Party Terms”). Information regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://support.avaya.com/Copyright.

Preventing Toll Fraud

“Toll fraud” is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud Intervention

If you suspect that you are being victimized by Toll Fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support Web site: http://support.avaya.com. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

The trademarks, logos and service marks (“Marks”) displayed in this site, the Documentation and Product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation and Product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners, and “Linux” is a registered trademark of Linus Torvalds.

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support Web site: http://support.avaya.com.

Contact Avaya Support

Avaya provides a telephone number for you to use to report problems or to ask questions about your Product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: http://support.avaya.com.

Contents

Chapter 1: New in this release

The following sections detail what's new in this document for Avaya Communication Server 1000 Release 7.5:

Features

There have been no updates to the feature descriptions in this document.

Revision history

March 2012	Standard 03.02. This document is up-issued to support Avaya Communication Server 1000 Release 7.5, and contains changes relating to updates to the Messenger software.
November 2010	Standard 03.01. This document is up-issued to support Avaya Communication Server 1000 Release 7.5.
June 2010	Standard 02.01. This document is up-issued to support Avaya Communication Server 1000 Release 7.0.
November 2009	Standard 01.07. This document is up-issued to support Communication Server 1000 Release 5.5, and contains clarifications relating to add-on modules.
October 2008	Standard 01.06. This document is up-issued to support Communication Server 1000 Release 5.5, and contains changes relating to updates to the Messenger software.
September 2008	Standard 01.02. This document is up-issued to support Communication Server 1000 Release 5.5, and contains changes relating to updates to the Messenger software.
May 2008	Standard 01.01. This document is issued to support Communication Server 1000 Release 5.5. Some of the information contained in this new document was previously contained in DECT Fundamentals, NN43120-114.

Chapter 2: Introduction

DECT Messenger is a client-server software platform for processing and converting messages between various protocols. It consists of several modules, some of which can be deployed independently on other PCs in the same network.

The DECT Messenger modules are grouped as follows:

- Core—core components of the software, including security and maintenance tools.
- Input/Output modules—used for sending or receiving messages to or from supported devices.
- Add-Ons—optional expansion modules adapted for specific customer needs.
- Web Administrator—a web application that enables web-based access to a limited set of functions.

A configuration tool called eCONFIG is also available. See Getting Started with DECT Messenger for details on configuring a fresh installation.

There are two installation types available for DECT Messenger:

- Server mode – includes the Kernel module, the database and the web access module; all other modules are also available for selection.
- Client mode – includes a limited set of modules that can be deployed on a client PC.

These modules connect to a DECT Messenger Server installed in the same network. Using the client mode installation is optional; for small sites the modules can be deployed on the same PC using the server mode installation.

Chapter 3: System requirements

The following sections describe the system requirements for DECT Messenger.

Supported operating systems

The following operating systems are supported by DECT Messenger (both client and server install):

- Windows XP Service Pack 3 (x86)
- Windows Server 2003 Service Pack 2 (x86)
- Windows Server 2003 R2 Service Pack 2 (x86)
- Windows Server 2008 (x86/x64)
- Windows Server 2008 R2

Note:
When using a Microsoft SQL Server database engine for the DECT Messenger database, a server operating system is required.

Server PC requirements

DECT Messenger is supported on all servers that fulfill the Microsoft Hardware Compatibility List (HCL) and meet the following minimum requirements:

- 1.4 GHz or higher CPU (2 GHz or higher is recommended)
- 256 MB of RAM (1 GB or more is recommended)
- 1 GB of free hard disk space (10 GB or more is recommended)
- DVD drive
- 1 free USB port for Security Dongle (2 if you use the Monitoring Diversion feature in conjunction with a 2000 IPS PBX system)

The recommended computer name for the server PC is Messenger.

Client PC Requirements

DECT Messenger client modules are supported on any PC capable of running the supported operating systems. A DVD drive is required in order to install the product.

Optional hardware

Depending on the way DECT Messenger is being used, there are several hardware devices that may be needed in order to use certain features:

- National Instruments equipment for digital input/output (contacts) and analogue input for software module eIO. This is only required if you use the eIO software module.
- V.24 multi-port card
- Analogue Modem for dialing to GSM provider to send SMS messages. This is only required to send SMS messages to a GSM (cell phone) provider using a dial-in option.

Database server requirements

DECT Messenger optionally uses Microsoft SQL Server to store data (the choice is presented when installing in Server mode). In most cases, Microsoft SQL Server 2008 Express Edition is sufficient; a copy is distributed with the DECT Messenger DVD.

If necessary, you can use a non-Express edition of SQL Server (e.g. for maintenance tools and to increase the database size limits).

Installation of non-Express editions is not covered by this guide and is subject to additional licensing costs to Microsoft. DECT Messenger supports SQL Server 2005 and 2008 editions.

Note:

During the installation, the system administrator (sa) account is used. Make sure you know the password for the sa account and that the SQL Server is configured to run in mixed mode (SQL Server and Windows Authentication).

Network information

Table 1: Network information

Port	Description
80	Port for HTTP traffic used by Web Administrator application
1433	Default SQL Server port (when using SQL Server as database storage engine)
3101	Default eKERNEL server port for eDMSAPI
3102	Default eKERNEL server port for eCAP – Eldad
3103	Default eKERNEL server port for eCAP – Televic
3104	Default eKERNEL server port for eCAP – Generic
3105	Default eKERNEL server port for eASYNC module
3106	Default eKERNEL server port for eVBVoice*
3107	Default eKERNEL server port for eCSTA
3108	Default eKERNEL server port for eIO
3109	Default eKERNEL server port for eWEB
3110	Default eKERNEL server port for eSMTP Server
3111	Default eKERNEL server port for eSMTP
3112	Default eKERNEL server port for eAPI
3113	Default eKERNEL server port for eESPA (Master)
3114	Default eKERNEL server port for eESPA (Slave)
3115	Default eKERNEL server port for eLOCATION
3116	Default eKERNEL server port for eSNMP
3117	Default eKERNEL server port for eSMS
3118	Default eKERNEL server port for eNET
9000	Default eKERNEL server port for eCONFIG

Chapter 4: Installation steps

The Avaya DECT Messenger installation steps assume that all operating system and hardware requirements are met. The following flow chart summarizes the installation and setup procedures.

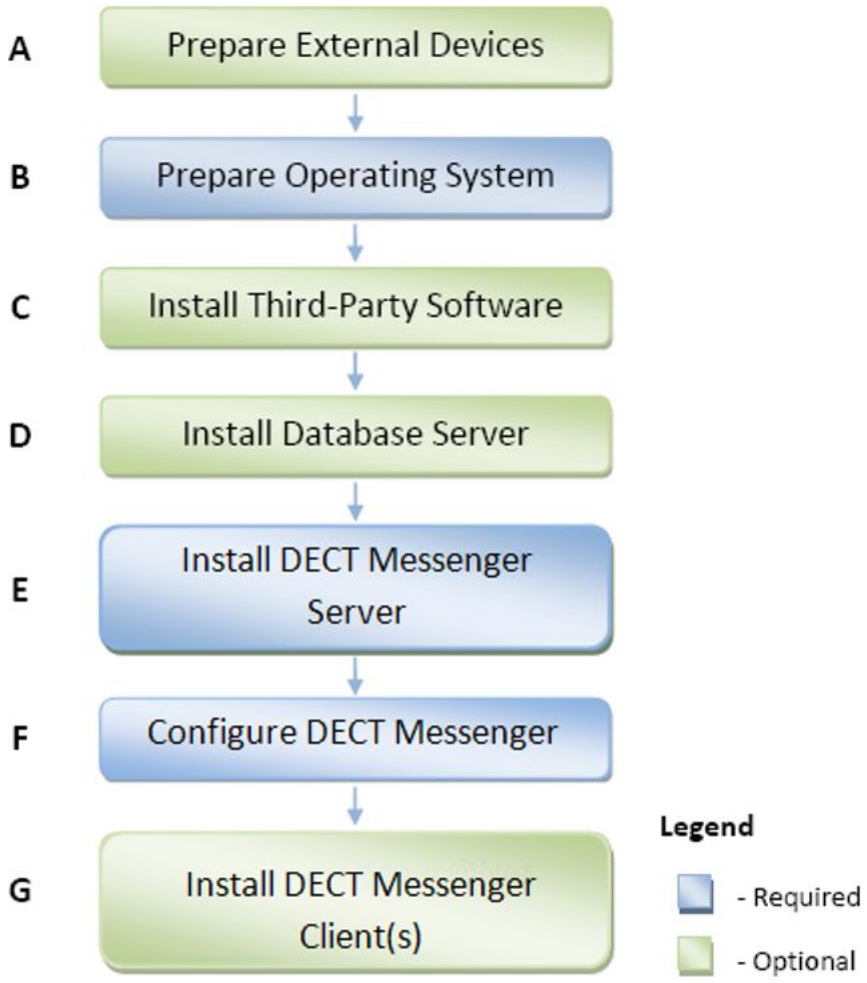


Figure 1: DECT Messenger installation steps

For more information about each step, refer to the following topics:

- Preparing external devices on page 17
- Preparing the Operating System on page 25
- Installing third-party software on page 31

- Installing SQL Server on page 32
- Installing DECT Manager on page 37
- Configuring DECT Messenger on page 46
- Install a DECT Messenger Client on page 41

Chapter 5: Preparing external devices

This chapter provides information on preparing external devices.

Integration with IP-DECT Manager

You can connect DECT Messenger to an IP-DECT system. The communication between DECT Messenger and the IP-DECT system is only for LRMS to or from the DECT handsets. DECT Messenger uses the CTI port on the IP-DECT system to send and receive LRMS messages. This port does not support CSTA.

The connection consists of a TCP/IP connection between the DECT Messenger and the IP-DECT system. However, the IP-DECT system is normally placed in a separate VLAN that is used only for VoIP. Therefore, in general, the connection between DECT Messenger and the IP-DECT system is established using a VLAN router.

Before installing DECT Messenger, you must verify that the VLAN router allows traffic from the DECT Messenger to the IP-DECT system. The IP-DECT system can be connected to CS 1000.

The following subsections give a description of the connections.

Integration with Call Server 1000

The following figure shows an IP-DECT system configured with CS 1000 and DECT Messenger. As shown here, DECT Messenger can be connected directly to the VLAN which is used for VoIP, or it can be connected to that VLAN via a VLAN router. Connection via a VLAN Router requires that the VLAN Router allows traffic between the DECT Messenger PC and the DAP Controller PC (IP-DECT system). DECT Messenger uses a TCP/IP port on the DAP Controller.

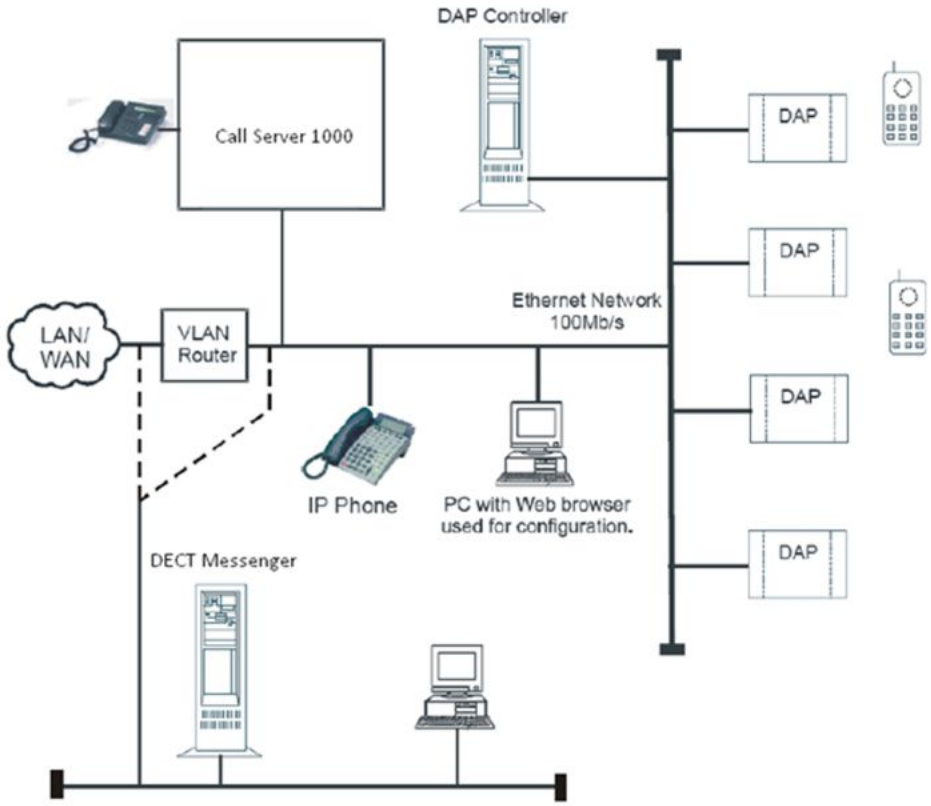


Figure 2: Integration with Call Server 1000

IP-DECT Manager TCP/IP port for DECT Messenger

On the IP-DECT Manager system, there is a TCP/IP port open for LRMS (E2) messages sent to and from handsets. The port number is not fixed but determined in a configuration file (dapcfg.txt) located on the DAP Controller PC. The path to the dapcfg.txt file depends on the version of the IP-DECT Manager system, as shown in this example:

```
C:\Documents and Settings\All Users\Application Data\Avaya\DAP
Controller\<version/system name>\dapcfg.txt
```

Open the file with an ASCII editor of your choice (such as Notepad).

Important:

Do not make any changes to the file.

The following is an example of the dapcfg file contents:

```
; dapcfg.txt for system My System
; Created by DapConf.exe on 09/07/2010 16:20:56
;
; Please do not modify this file!!
;
[DAP-IMAGEFILE] ; Start of DAP image file section
4910b510.dwl
[DS] ; Start of DS address section
192.168.17.74 28000-28017
[DAP] ; Start of DAP address section
239.192.49.49 3000-22635 1 ;
```

```
[DAPPRF] ; Start of DAPPRF address section
192.168.17.74
[CDA] ; Start of CDA address section
192.168.17.74 30160
[GK] ; Start gatekeeper address section
192.168.17.200 5060
[XDS] ; Start SIP section
b2b_ua=yes
[CONFIG] ; Start of static config section
CONFIGFILE=replace
IPCONFIG=replace
```

The TCP/IP port for Messaging is located in the [DS] section. The IP address of the DAP Controller PC is followed by a port number range.

The port for LRMS messaging is the first port number plus 1. In the above example, the LRMS port on the DAP Controller is port 28001. This is the port number that must be entered in the DECT Messenger eDMSAPI module along with the DAP Controller IP address.

The port number for Location Detection is the first port number plus 8. In the above example, the Location Detection port on the DAP Controller is port 28008. This is the port number that must be entered in the DECT Messenger eLOCATION along with the DAP Controller IP address.

National Instruments Analog/Digital Hardware

To use Digital Input/Output and Analogue Input functionality in message processing, DECT Messenger requires National Instruments FieldPoint (FP) hardware modules.

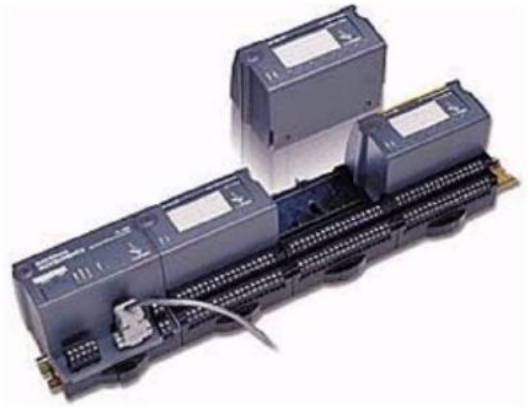


Figure 3: National Instruments FieldPoint hardware modules

DECT Messenger supports the following types of IO modules:

- Control modules
- I/O modules

The following tables provide an overview of these modules.

Table 2: Control modules overview

Module type	Description	Additional Info
FP-1601	Control module with IP interface	Interface module between the I/O modules and DECT Messenger. Controls up to 9 I/O modules directly.
FP-1000	Control module with V.24 interface	Interface module between the I/O modules and DECT Messenger. Controls up to 9 I/O modules directly. Up to 24 FP-1001 modules can be connected using RS485 bus to expand the system with extra I/O modules.
FP-1001	Expansion control module	Must be connected to the RS485 interface on the FP-1000. One FP-1001 can control up to 9 I/O modules. The maximum number of FP-1001 modules one RS485 bus is 24
PS-2	Power Supply	24 Volts DC

Table 3: I/O modules overview

Module type	Description	Additional Info
FP-AI-100	Analogue input	8 Analogue inputs each can be set to one of the following ranges: 30V, 15V, 5V, 1V, 0-30V, 0-15V, 0-5V, 0-1V, 20mA, 0- 20mA, 4-20mA.
FP-DI-300	Digital input	8 discrete input channels. These inputs are sinking inputs for 24VDC
FP-DI-301	Digital input	16 discrete input channels. These inputs are sinking inputs for 24VDC
FP-DI-330	Digital input	8 discrete input channels. Universal inputs work with any voltage from 5V TTL up to 250VDC/VAC. Compatible with sourcing, sinking, or power sensing applications.
FP-DO-400	Digital output	8 discrete output channels. Max. 2A per output, max 8A per module. Maximum voltage 30VDC
FP-DO-401	Digital output	16 discrete output channels. Max. 2A per output, max 8A per module. Maximum voltage 30VDC

Note:

For each I/O module, one Terminal Base (TB-1) is required.

The following figure shows an example of how a rail of National Instruments I/O modules can be connected to DECT Messenger. On a rail, there can be various types of I/O modules. The maximum number of modules for each rail is 8.

Important:

Although you can build configurations with many input or output modules, there is a limitation on the number of input or output modules than an eIO DECT Messenger module instance can handle. One eIO module can handle up to 8 input/output modules. So when you connect a 9th module, even though it is recognized by National Instruments Measurement & Automation Explorer, it cannot be handled by the eIO module.

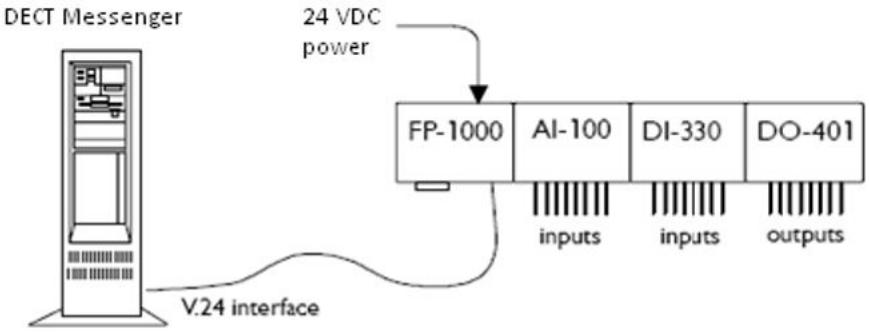


Figure 4: Example of an I/O module connected to DECT Messenger

Note:

The maximum number of contacts per eIO Module in the DECT Messenger is 128.

If necessary, more than one rail can be connected to DECT Messenger. The following figure shows an example configuration of three rails with National Instruments modules connected to a DECT Messenger. The three rails with modules are connected together via the RS-485 bus.

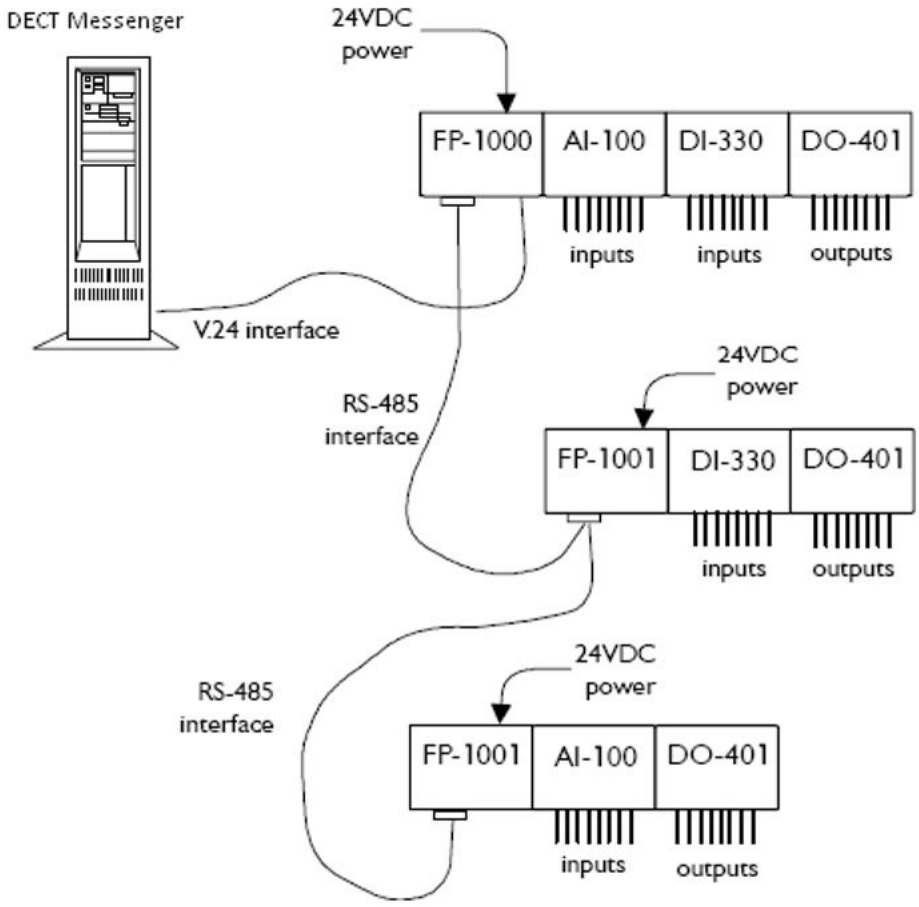


Figure 5: Example of connecting multiple FP rails to DECT Messenger

The connection between the DECT Messenger PC and the first rail is achieved by means of V.24. This means that the maximum cable length is determined by the V.24 characteristics and the cable type.

If you have more than one rail (only available on a project basis), the connection between the rails and therefore the connection between the FP- 1000 and FP-1001 modules is achieved by means of an RS-485 connection. This is a 4–wire bus connection that allows a maximum distance of approximately 1000 meters.

Instead of using an FP1000 module as Controlling Module on a rail, you can use the FP-1601 module. The FP-1601 module has an Ethernet interface to DECT Messenger instead of the V.24 interface. However, this module is not supported in the standard DECT Messenger product range; it is only supported on a project basis.

Hardware installation

The detailed installation procedures for the FP modules are in the installation documents located on the DECT Messenger DVD:

```
D:\Documentation\\Background –eIO – xxx.pdf
```

where **D:** is the drive letter of your DVD drive.

For the latest version of the documents, also refer to National Instruments website at http://www.ni.com.

Software installation

To connect to the FP I/O modules, DECT Messenger requires third-party software components from National Instruments. The software consists of two main parts:

- Measurement & Automation Explorer software for setting up the configuration of the FieldPoint modules
- OPC (Object Linking and Embedding for Process Control) Server

The DECT Messenger eIO module connects through the OPC Server to control the I/O modules, as shown in the following figure:

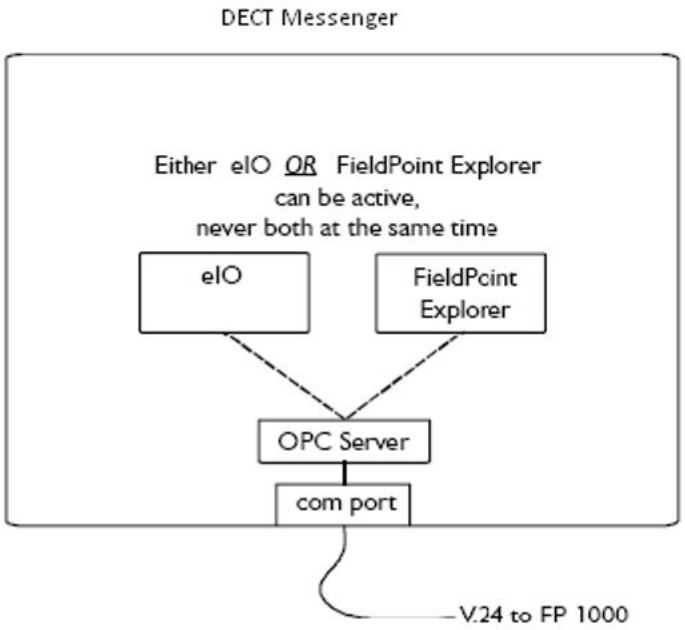


Figure 6: DECT Messenger interaction with FP I/O modules

Important:

The OPC Server software can be controlled by only one application. This means that you can have either the eIO Module or the Measurement & Automation Explorer active, but not both. Remember to close one of the applications before starting the other.

Both components are part of the National Instruments FieldPoint software package that is located on the DECT Messenger DVD. Also required is a DataSocket package that DECT Messenger uses to interact with the OPC Server.

For details on how to install these packages, refer to Installing National Instruments Field Point and DataSocket on page 34.

Chapter 6: Preparing the Operating System

As a general rule, ensure that the operating system of the target PC is up-to-date, with the latest service packs and patches installed prior to installing DECT Messenger product.

Note:

Avaya recommends that you do not configure the Automatic Updates feature in Windows to **Automatic**. This will avoid system reboots while DECT Messenger is operating.

The following modules require additional features that are not enabled by default during a standard installation of Windows:

- Web Administrator—requires Internet Information Server (IIS). On IIS 7.x (Windows 2008/Windows 2008 R2), CGI support and IIS 6 Metabase compatibility options must also be enabled.
- SMTP Server—requires IIS and SMTP Service

Installing IIS

The steps necessary to install Internet Information Server depend on the operating system version.

Important:

Before installing IIS, verify that you are logged in using a computer administrator account. During installation, you may be prompted for the Windows operating system installation media (CDs/DVD). Ensure they are available before proceeding.

Use the following procedures to install IIS:

Installing IIS on Windows XP/Windows Server 2003/Windows Server 2003 R2

To install IIS or to verify that all necessary components are installed, perform the following steps:

1. Click **Start > Settings > Control Panel**
2. Select **Add or Remove Programs**
3. Select the**Add/Remove Windows Components** tab. The Windows Components Wizard window appears.
4. Ensure that **Internet Information Services (IIS)** option is checked, as shown in the following figure.

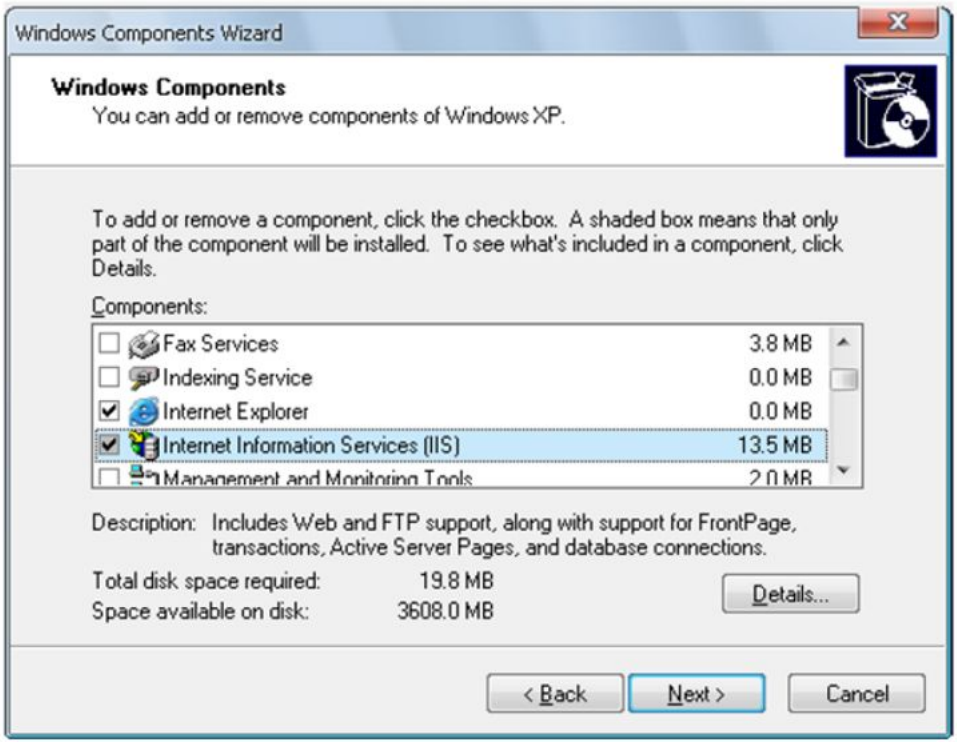


Figure 7: Windows Components Wizard window

5. Select **Details**. The Internet Information Services (IIS) details window appears, listing the installed IIS subcomponents.
6. Verify that **Common Files** and **Internet Information Services Snap-In** options are checked.
7. (Optional) If you intend to use SMTP Server module in DECT Messenger, ensure that **SMTP Service** option is also checked, as shown in the following figure.

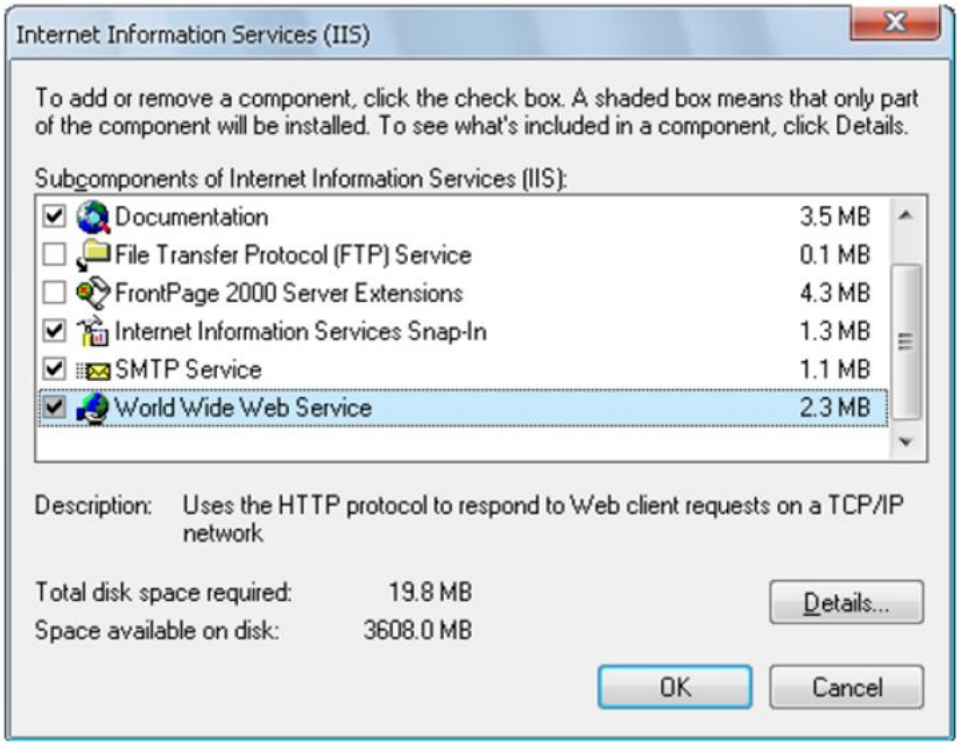


Figure 8: Internet Information Services (IIS) window

8. Select **World Wide Web Service** and click **Details**; in the World Wide Web Service details window that appears, ensure that **World Wide Web Service** is checked, then click **OK** to close it.
9. Click **OK** to close the Internet Information Services (IIS) details window.
10. Click **Next** to begin the installation. You may be prompted to insert the Windows installation CD/DVD during this step.
11. Click **Finish** to complete the installation. You may be prompted to restart the computer.

Installing IIS on Windows Server 2008/Windows Server 2008 R2

Windows Server 2008/R2 platforms introduce the concept of server roles and server features. On these operating systems, IIS is built into the Web Server role, while SMTP Server functionality is a server feature that runs on top of the Web Server role.

Use this procedure to add the Web Server (IIS) role on a Windows Server 2008/R2 server.

1. Go to the Server Manager window. (This window automatically opens when the computer is started. If it does not open, right-click the **Computer** icon on the desktop and select **Manage**.)
2. In the left pane, select **Roles**. The right pane is updated with the server roles currently installed, as shown in the following figure.

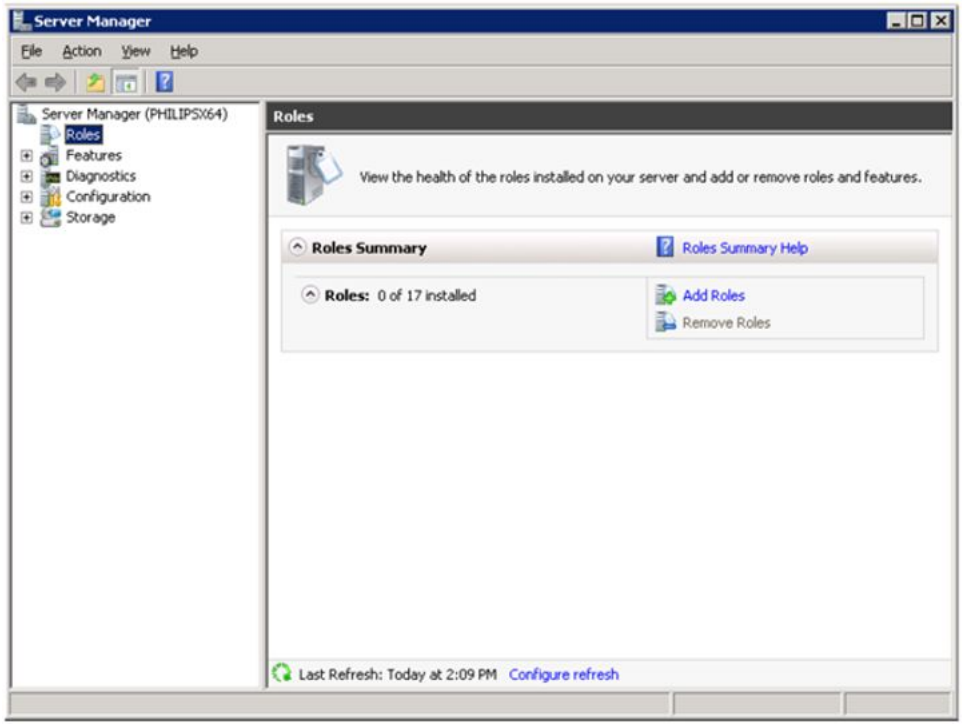


Figure 9: Server Manager window

3. Click **Add Role**. The Add Roles Wizard opens. Click **Next** to skip the opening screen.
4. Check the **Web Server (IIS)** box in the list of available roles and click **Next**.
5. Read the introductory information displayed, and then click **Next** again to reach the Select Role Services wizard page.
6. In the list of role services, verify that the following options are checked:
    a. **Web Server > Application Development > CGI**
    b. **Management Tools > IIS 6 Management Compatibility > IIS 6 Metabase Compatibility**
7. Click **Next**. The Confirm Installation Selections page displays, where you may review the components to be installed.
8. Click **Install** to being the installation.
9. When installation is completed, click **Close** to close the wizard.

Adding the SMTP server feature

To add the SMTP server feature, do the following:

1. Go to the Server Manager window (automatically opens when the computer is started; if not open, right-click on the **My Computer** icon on the desktop and select **Manage**.)
2. In the left pane, select **Features**. The right pane is updated with the server features currently installed.

3. Click **Add Features**. The Add Features Wizard window opens, as shown in the following figure.

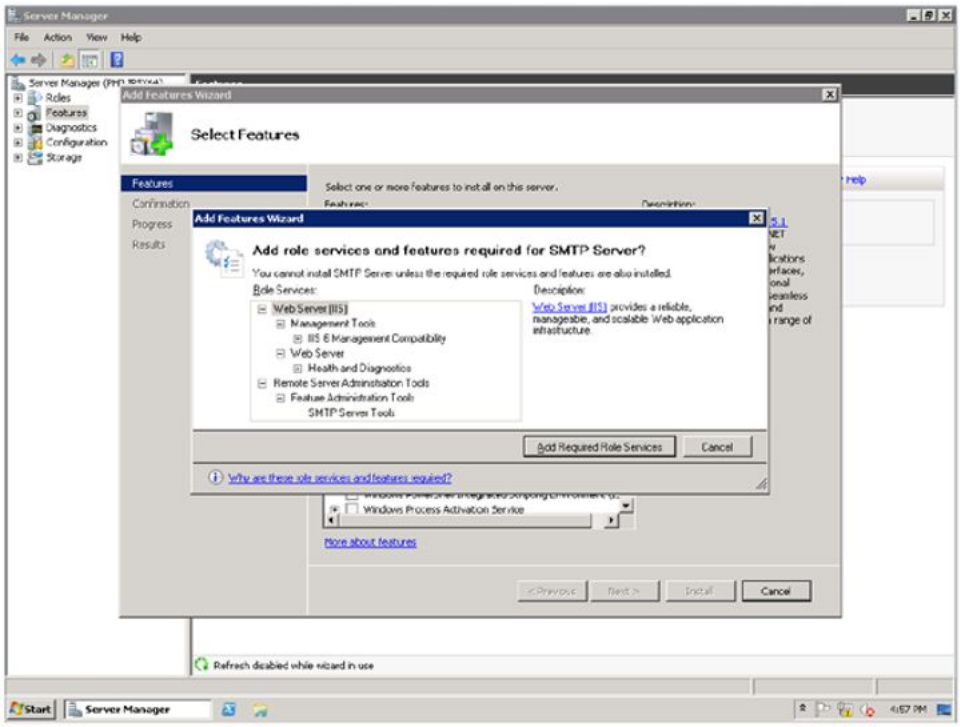


Figure 10: Add Features Wizard window

4. Check **SMTP Service** from the list of available features. A confirmation screen displays, prompting to add other features and services dependent on SMTP Service.
5. Click **Add Required Role Services**.
6. Click **Next** to proceed to the Web Server (IIS) role information screen.
7. Click **Next**. The Select Role Services wizard page appears, with the necessary services already selected.
8. Click **Next**. The Confirm Installation Selections page displays, where you may review the components to be installed.
9. Click **Install** to being the installation.
10. When installation is completed, click **Close** to close the wizard.

Chapter 7: Installing third-party software

This chapter describes how to install the optional third-party software that is required to run certain DECT Messenger modules. The software is located on the DECT Messenger DVD.

Installing Adobe Reader 9

Adobe Reader 9 is required to read the documentation on the DVD. On the server, it is also used by the eGRID configuration module to display information about the configurable database fields.

Note:
Only the English version of Adobe Reader is available on the DVD. You can download a copy in other languages from the Adobe website: http://get.adobe.com/reader.

To install Adobe Reader 9, complete the following steps.

1. Insert the DECT Messenger product DVD.

   The DVD Main Menu window appears. (If the Main Menu does not appear, double click **D:\Autorun.exe**, where **D** is the drive letter of your DVD drive).

2. Select **Adobe Reader 9** from the DVD Main Menu.

   The installer prepares the files and then the Destination Folder window appears, as shown the following figure.

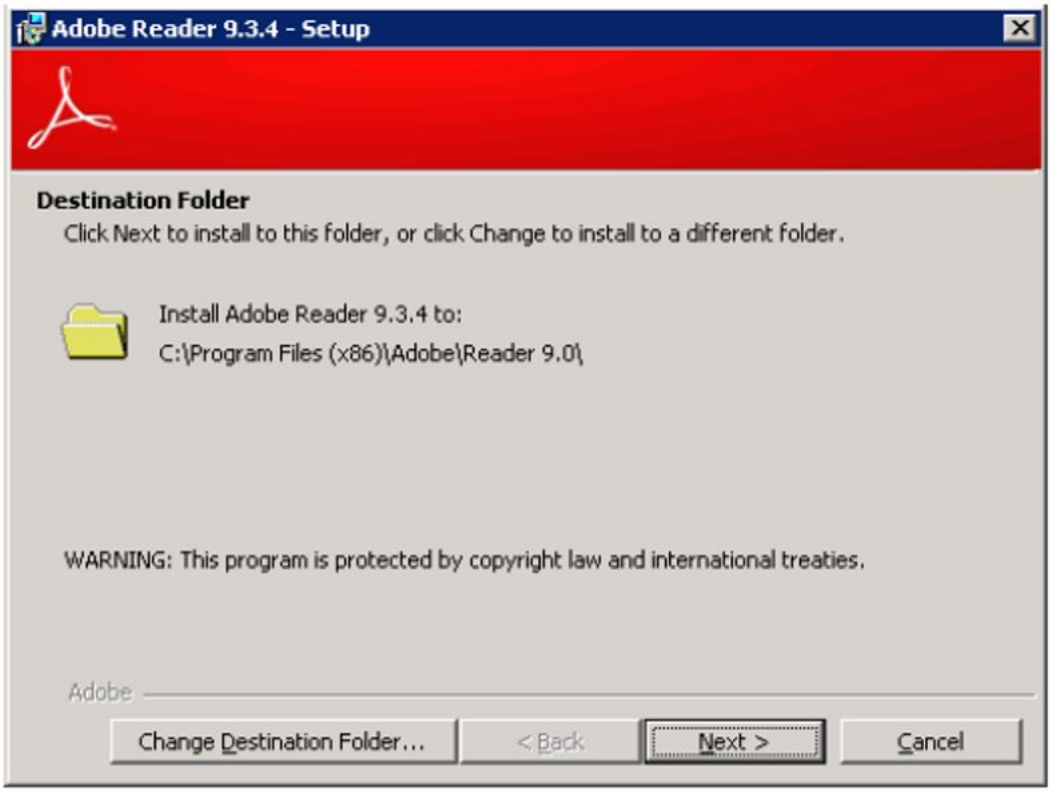


Figure 11: Adobe Reader 9 Destination Folder window

3. Verify the installation location and click **Next**.

   If you want to change the destination folder, click **Change Destination Folder** and specify a new location.

4. Click **Install** to start the installation.
5. Click **Finish** after installation completes.

Installing SQL Server

DECT Messenger optionally uses SQL Server for its MessengerData database. This database contains runtime message information and increases the performance in high-load environments. If performance is not an issue, you do not need to install SQL Server as DECT Messenger includes a Microsoft Access database that can be used with no additional configuration.

If an existing Microsoft SQL Server is available for use, DECT Messenger can deploy its database to it. In this case, you only need to provide the credentials (SQL system administrator account and password) necessary to connect to it.

Note:

SQL Server 2000/MSDE2000 are not supported. When using an SQL Server installed on a PC different from the DECT Messenger PC, the uptime of DECT Messenger depends on the uptime of the database server PC. The software continuously uses the database for storing temporary data, such as active alarms. Therefore, stopping the database server PC causes DECT Messenger to stop functioning.

Installation of the SQL Server Express Edition software requires that the following software be installed on the system:

- Microsoft .NET Framework 3.5 with SP1 (selectable from the DVD Main Menu)
- Microsoft Windows Installer 4.5 (selectable from the DVD Main Menu)

If these items are not already installed, you can install them during the SQL Server 2008 Express Edition installation process as they are included on the DECT Messenger DVD.

Installing SQL Server 2008 Express Edition

Note:

For convenience, a free copy of SQL Server 2008 Express Edition is included on the DVD.

1. Insert the DECT Messenger product DVD.

   The DVD Main Menu window appears. (If the Main Menu does not appear, double click **D:\Autorun.exe**, where **D** is the drive letter of your DVD drive).

2. If not already installed, select and install the following from the DVD Main Menu:

   - Microsoft .NET Framework 3.5 with SP1
   - Microsoft Windows Installer 4.5

3. From the DVD Main Menu, select **SQL Server 2008 Express Edition + SP1**.
4. Select **Default SQL 2008 Instance** or **Named SQL 2008 Instance**.
5. Enter the password for the sa administrator account (you will also need this during the main DECT Messenger install).

Although the SQL Server Express Edition meets all requirements for a live production DECT Messenger system, it has the following limitations:

- The maximum database size is 4 GB.
- SQL Server 2008 Express Edition uses only one processor from the server computer.
- To schedule backups, you must install additional software as SQL Server 2008 Express Edition does not include an SQL agent for this purpose.

Installing National Instruments Field Point and DataSocket

This section describes the installation of modules related to the eIO component of DECT Messenger. You can skip this section if you do not plan use the eIO module.

1. Insert the DECT Messenger product DVD.

   The DVD Main Menu window appears. (If the menu does not appear, double click **D:\Autorun.exe**, where **D** is the drive letter of your DVD drive).

2. From the DVD Main Menu, select **National Instruments Field Point 6.0.6 f1** .

   The automated installation begins. Wait until it completes.

3. Select **National Instruments DataSocket 4.7.1**.

   The automated installation begins. Wait until it completes.

4. Restart the computer.

5. Verify that the software was correctly installed by double-clicking the **Measurement & Automation** shortcut on the desktop.

   The Measurement & Automation Explorer screen opens, as shown in the following figure.

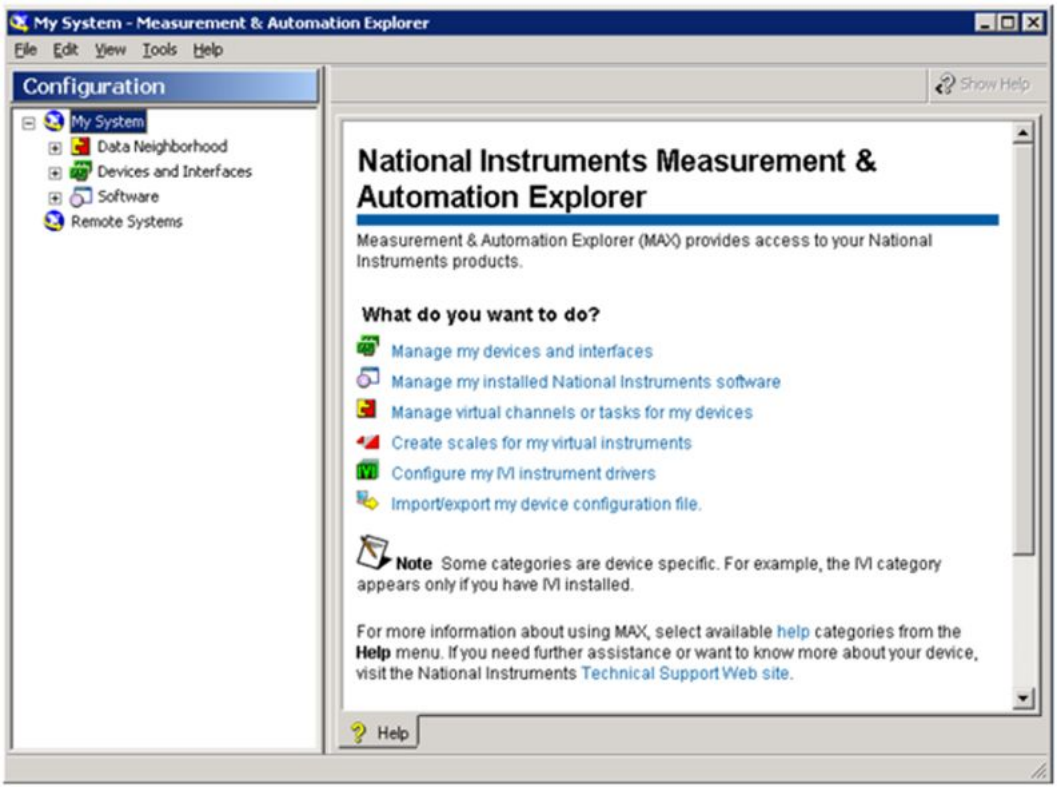


Figure 12: National Instruments Measurement & Automation Explorer screen

For details on how to configure the Field Point modules with Measurement & Automation Explorer software, refer to the National Instruments documentation.

Chapter 8: Install DECT Manager

This chapter describes how to install the DECT Messenger software. If you need to deploy one or more DECT Messenger software modules on a different PC, you can find the steps for installing additional DECT Messenger copies in client mode in the Installing DECT Messenger Client subsection.

Before you start

Before installing DECT Messenger software, ensure that the following preconditions are met:

- The computer and operating system are prepared, as described in Preparing external devices on page 17 and Preparing the Operating System on page 25.
- Any third-party software for modules that you intend to use (such as National Instruments software) is installed, as described in Installing third-party software on page 31

Installing DECT Manager

1. Insert the DECT Messenger product DVD.

   The DVD Main Menu window appears. (If the Main Menu does not appear, double click **D:\Autorun.exe**, where **D** is the drive letter of your DVD drive.)

2. Select **Server** from the DVD Main Menu.

   The Requirements Setup Wizard checks that the server meets the configuration requirements, as shown in the following figure.

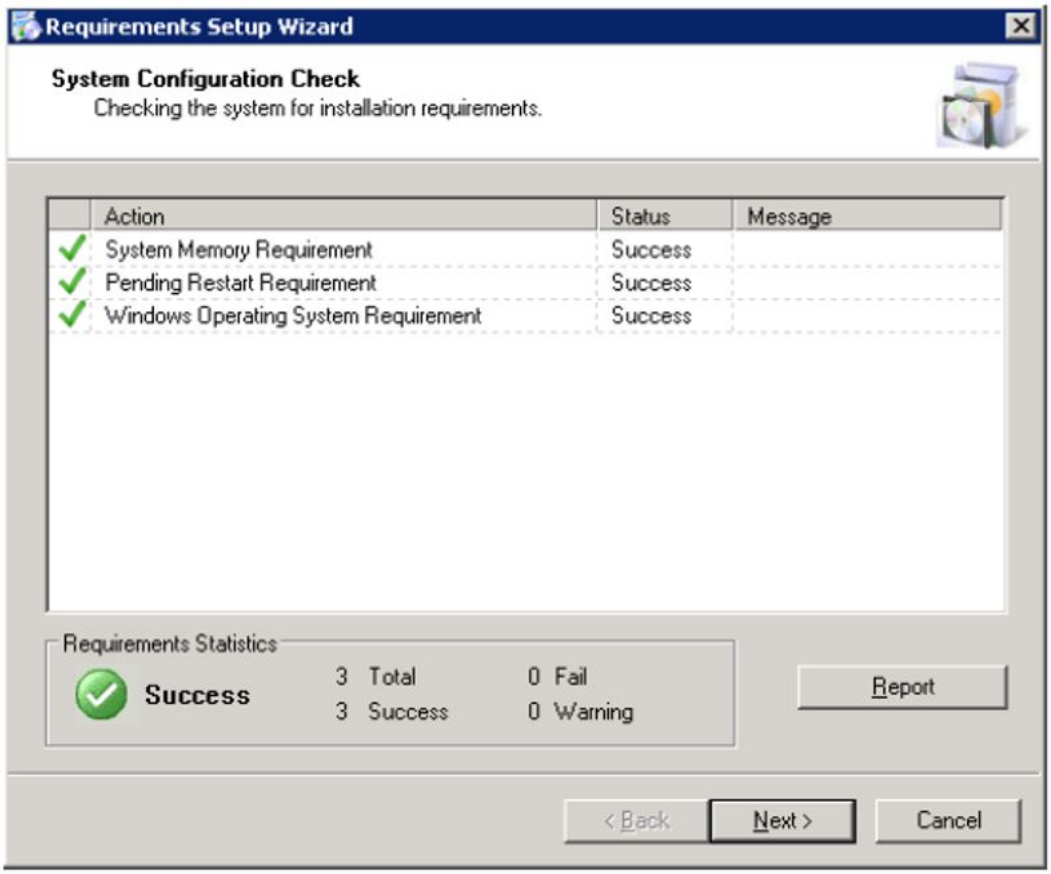


Figure 13: System Configuration Check window

Note:

If the Pending Restart Requirement check fails (for example, due to a conflict with other installed software), click **Cancel** to stop the installation and reboot the PC, and then restart the installation procedure.

3. Click **Next** to continue.

   The Install Prerequisites window appears, as shown in the following figure:

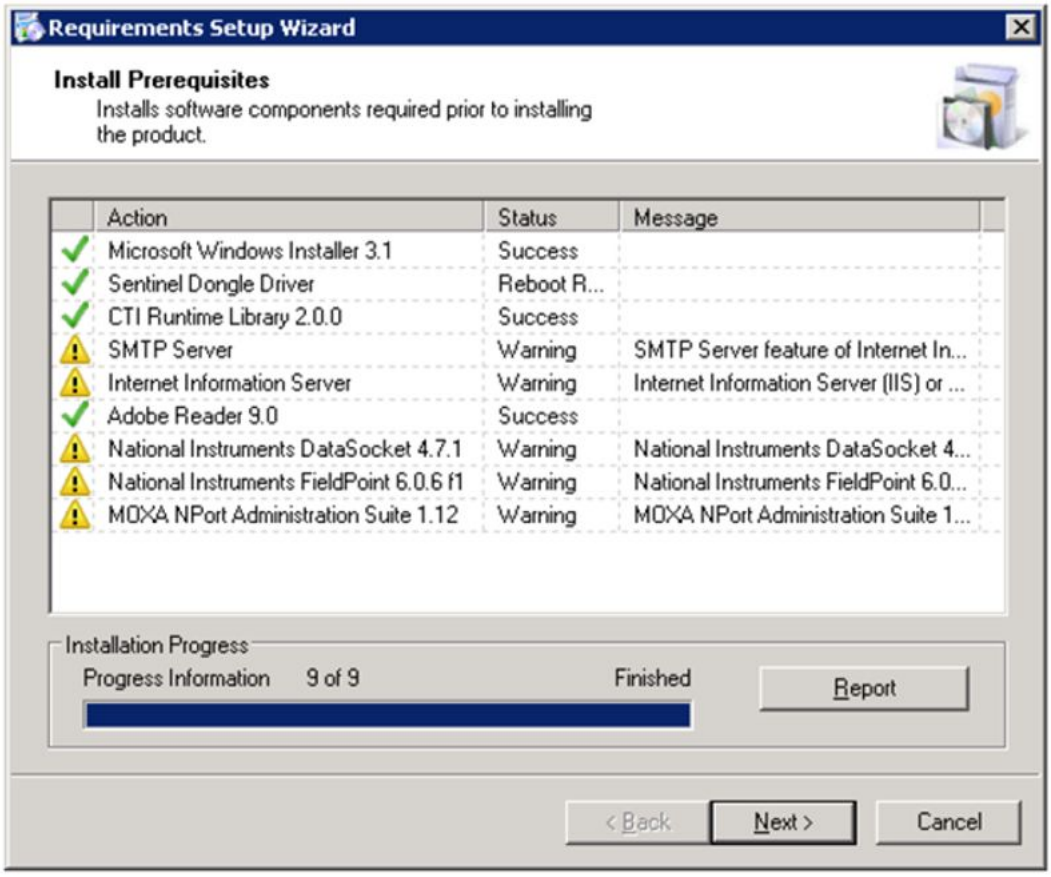


Figure 14: Install Prerequisites window

4. The Setup Wizard checks if all necessary third-party software components are present and attempts to automatically install any missing mandatory ones. The mandatory components are as follows:
   - Sentinel Dongle Driver package
   - CTI Runtime Library 2.0
5. Verify the warning messages about missing optional components. If you intend to use DECT Messenger software modules that are mentioned as “not available” in the warning messages, cancel the installation. Then, install the required components as indicated in the Preparing the Operating System and Installing Third-Party Software chapters.
6. Click **Next**.
7. When prompted to restart the system, click **Restart Now**. Wait until the PC is restarted and the installation continues.
8. When the Welcome to the InstallShield Wizard for Messenger@Net window appears, click **Next**.

   The Setup Type window appears.

9. Select **Complete** to install all available components in the default installation path and click **Next** to continue.

   The Database Installation window appears, as shown in the following figure.

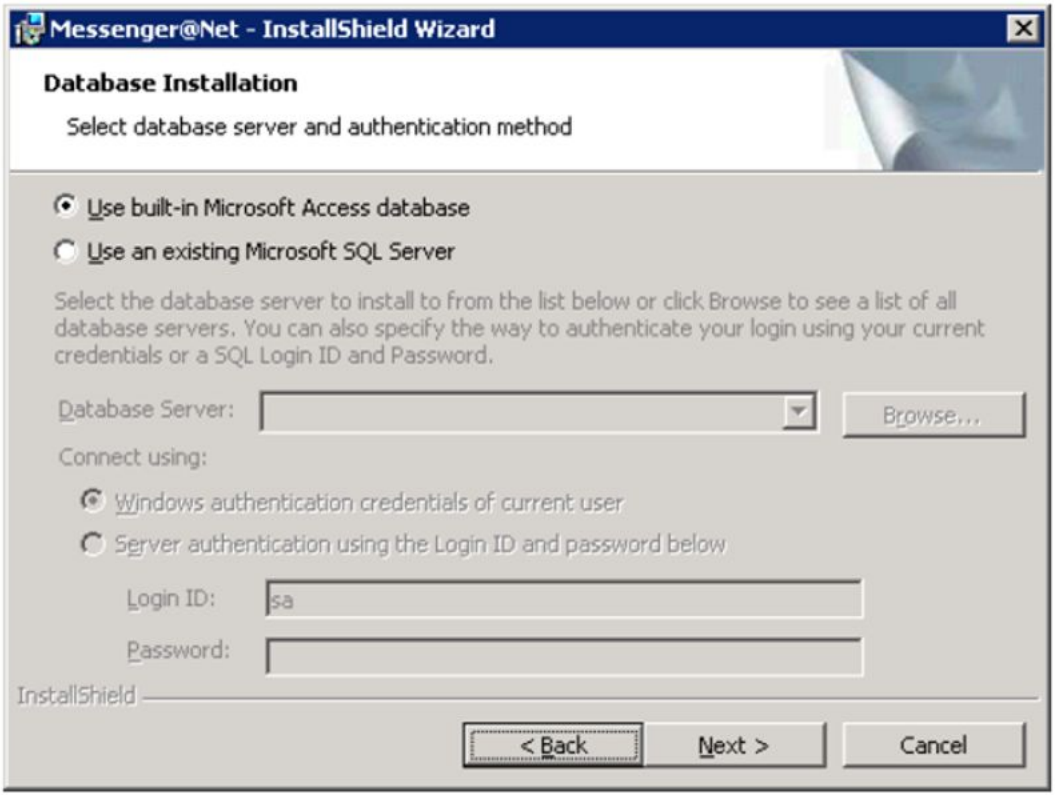


Figure 15: Database Installation selection window

Note:

On Windows Server 2008, you might get a **Files in Use** warning window. If this happens, select **Automatically close and attempt to restart applications** and click **OK**.

Note:

The default installation path is `C:\Program Files\Avaya\Avaya DECT Messenger` on 32-bit platforms and `C:\Program Files (x86)\Avaya \Avaya DECT Messenger` on 64-bit platforms.

10. Choose whether to use a Microsoft Access-only database or to use an existing Microsoft SQL Server. If using Microsoft SQL Server, also enter the location of the database server and the credentials required to connect to it.
11. Click **Next** to validate the database connection and continue.

    The Ready to Install window appears.
12. Click **Install** to begin the installation process.

13. When the installation completes, the **InstallShield Wizard Completed** screen appears. Click **Finish** to finalize the installation process.
14. When prompted to restart the computer, click **Yes** .

Install a DECT Messenger Client

Depending on customer needs, there might be situations when it is necessary to spread certain DECT Messenger components on several computers in the same network. The Client mode installation is actually part of the same installation package and, with a few exceptions, follows the same steps described for the Server mode setup.

Before you continue, ensure that the following preconditions are met:

- Ensure that the computer and operating system is prepared, as described in the preparation chapters.
- Ensure that any third-party software necessary for modules that you intend to use on this client PC (for example, National Instruments software) is installed.

Installing a DECT Messenger Client

1. Insert the DECT Messenger product DVD.

   The DVD Main Menu window appears. (If the Menu does not appear, double click **D:\Autorun.exe**, where **D** is the drive letter of your DVD drive.)
2. Select **Client** from the DVD Main Menu.

   The Requirements Setup Wizard checks that the server meets the configuration requirements.

   If the Pending Restart Requirements check fails (for example, due to conflicts with other installed software), click **Cancel** to stop the installation and reboot the PC, and then restart the installation procedure.
3. Click **Next** to continue.

   The Install Prerequisites window appears.
4. Verify the warning messages about missing optional components. If you intend to use Messenger@Net software modules that are mentioned as “not available” in the warning messages, cancel the installation. Then, install the required components as indicated in the Preparing Operating System and Installing Third-Party Software chapters.
5. Click **Next**.

   The Welcome to the InstallShield Wizard for Messenger@Net window appears.
6. Click **Next**.

   The Setup Type window appears.

7. Select **Custom** and click **Next** to continue.

   The Custom Setup window displays, allowing you to specify the components you want to install and run on the client PC, as shown in the following figure.

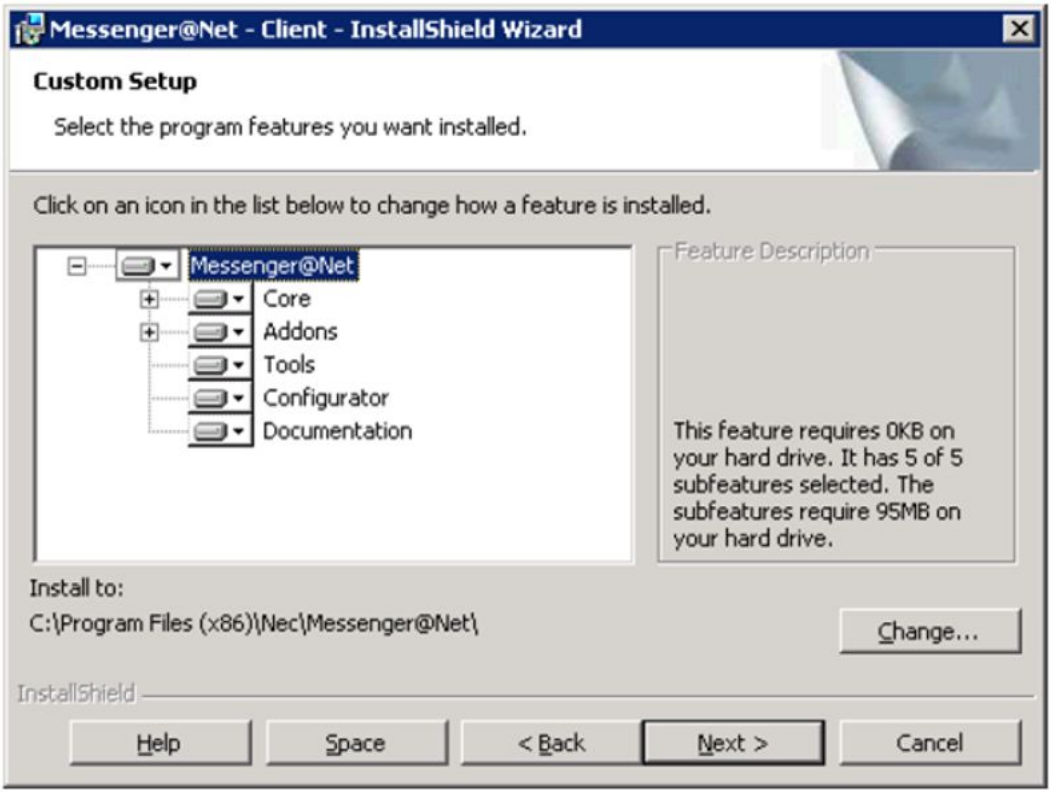


Figure 16: Custom Setup window

8. Select the features to install and click **Next**.

   The Ready to Install window appears.

9. Click **Install** to begin the installation process.
10. When the InstallShield Wizard Completed screen appears, click **Finish**.

When running a DECT Messenger Client installation, the following components are not available:

- eKERNEL and associated core components
- Web Administration console
- eSMTP Server module

Uninstalling DECT Messenger

DECT Messenger can be uninstalled using one of the following procedures. The procedures are applicable for both Server and Client installs.

Note:

During uninstall, the DECT Messenger databases and module configuration files (*.ini) are automatically removed. Make backups if you intend to restore them at a later point in time.

Uninstalling from Control Panel (Windows XP/2003)

1. Go to Start > Settings > Control Panel and open ‘Add or Remove Programs’.
2. Select DECT Messenger from the list of programs and click ‘Remove’. Wait until the product is completely removed.

Uninstalling from Control Panel (Windows 2008)

1. Go to Start > Control Panel and select ‘Uninstall a program’ (below ‘Programs’)
2. In the Uninstall or change a program window, select DECT Messenger and click ‘Uninstall’. Wait until the product is completely removed.

Uninstalling from the DVD

1. Insert the DECT Messenger product DVD. The DVD Main Menu window appears. If not, double click D:\Autorun.exe (where D is the drive letter of your DVD drive).
2. Select ‘Server’ from the DVD Main Menu (or ‘Client’ if this is a client installation).
3. The Requirements Setup Wizard checks that the server meets the configuration requirements.
4. Click ‘Next’ to continue to the Install Prerequisites page.
5. Click ‘Next’.
6. The Welcome to the InstallShield Wizard for Messenger@Net window appears. Click ‘Next’.
7. The Program Maintenance window appears.
8. Select ‘Remove’ and click ‘Next’.
9. On the following window, click ‘Remove’ to begin uninstalling.
10. When done, click ‘Finish’ to close the wizard.

Chapter 9: Getting started with DECT Messenger

This chapter describes the actions required for getting a DECT Messenger Server up and running. For further details on how to configure and use the Input/Output modules, refer to *DECT Messenger Fundamentals, NN43120-120*.

Loading licenses

Note:

Before using DECT Messenger, you need a license key and a dongle attached to one of the USB ports of your PC.

1. On the DECT Messenger Server PC, go to **Start Menu > Programs > CTI Developer Kit > Configurators > License Manager**.

   The License Manager tool starts up, showing a warning message that no license key is present, as shown in the figure below.

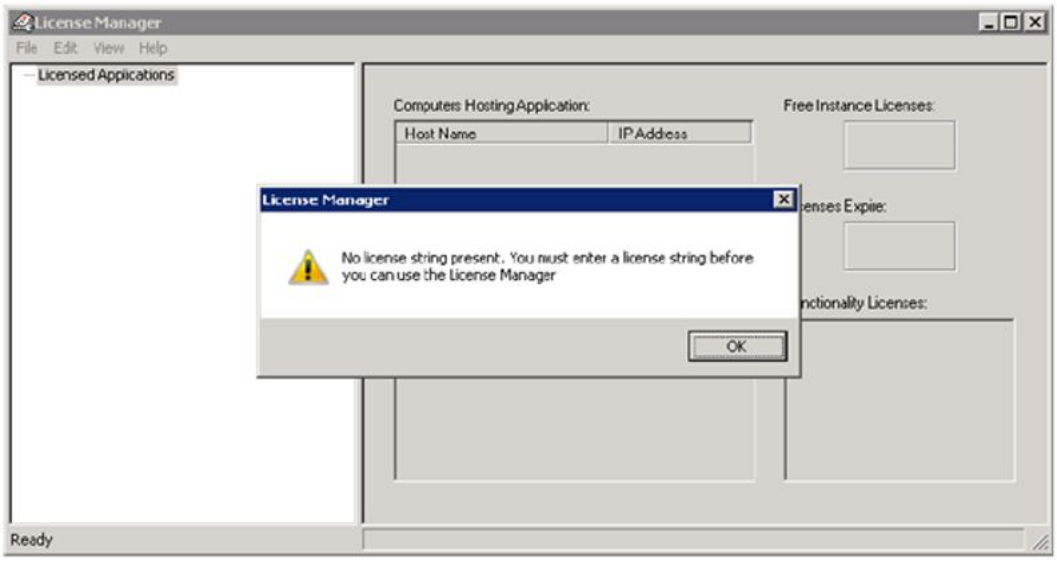


Figure 17: License Manager no license warning

2. Click **OK** to discard the warning.

3. In the License Manager menu, go to **File > Load a New License String** and browse for the license key file received from Avaya. Click **Open** to load it.
4. Verify the loaded licenses are correct. The following figure shows the loaded licenses in License Manager.

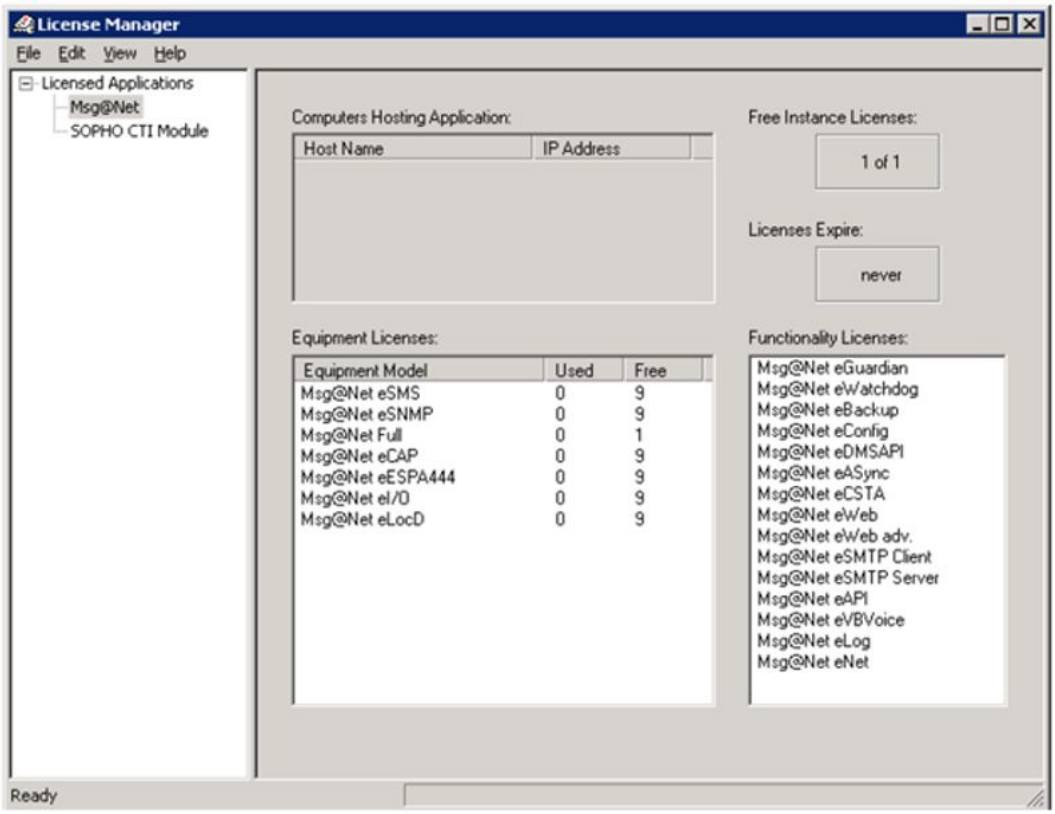


Figure 18: License Manager loaded licenses

Configuring DECT Messenger

Most DECT Messenger configuration settings are stored in a Microsoft Access database (*.mdb file). The database is located on the server PC, together with eKERNEL and other server-specific core components, in the following location:

`[INSTALLDIR]\Database\MessengerConfig.mdb`

The recommended way to configure DECT Messenger is by using the eCONFIG (Configurator) module. For expert users, it is also possible to modify directly the configuration database by using the eGRID module [link to subchapter below].

For convenience, DECT Messenger comes with a set of standard configuration databases [link to subchapter below], also known as “template” databases. These databases already come with preconfigured input/output modules, alarms and alarm groups. Avaya recommends using

– whenever possible – a standard configuration as a starting point when setting up a customer site.

Standard configuration database

Standard configuration databases are installed with the main product during a Server-mode installation. They are located in the following directory:

`[INSTALLDIR]\Default Templates\Database`

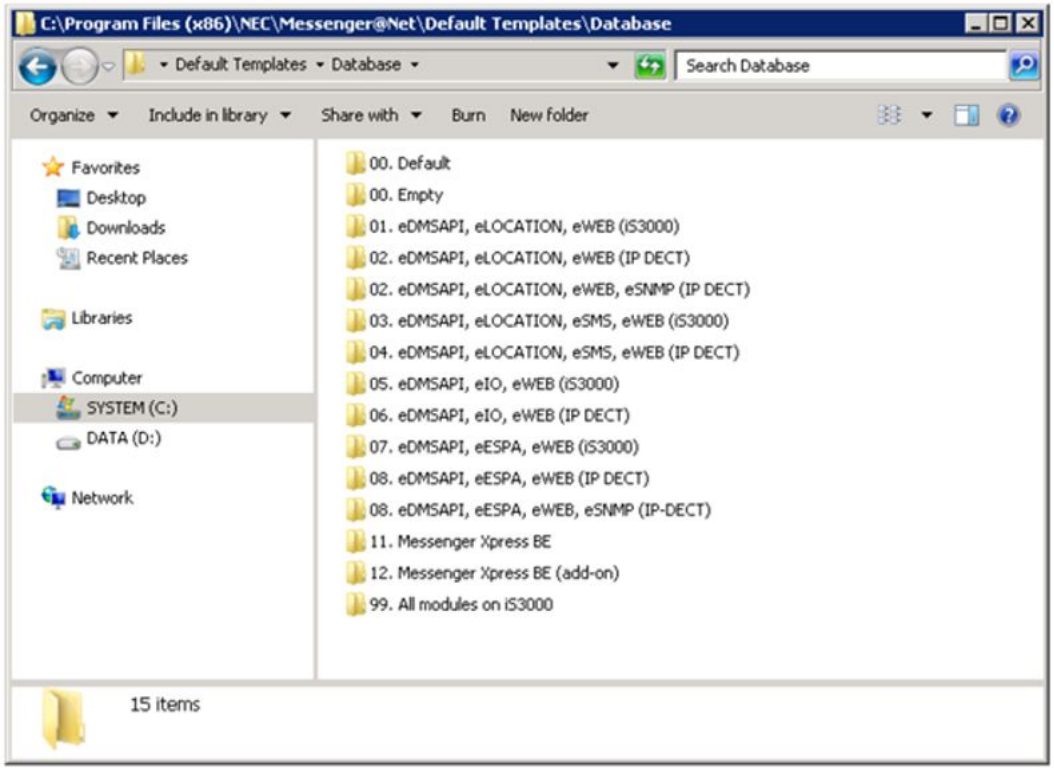


Figure 19: Default database templates

You can manually apply a configuration database or import one using Web Administrator.

To manually apply a configuration database:

1. Ensure that all DECT Messenger modules are stopped.
2. Open Windows Explorer.
3. Browse for the location of the desired template database on the server PC.

   For example: `C:\Program Files\Avaya\Avaya DECT Messenger \Default Templates\Database\02. eDMSAPI, eLOCATION, eWEB (IP DECT)`
4. Copy the chosen **MessengerConfig.mdb** template file.

5. Browse for the main database directory.

   For example: `C:\Program Files\Avaya\Avaya DECT Messenger \Database`.

   Rename the existing **MessengerConfig.mdb** to **MessengerConfig_old.mdb**, in case you need to return to the default configuration.

6. Paste the **MessengerConfig.mdb** template file into the main database directory.

To import a configuration database using Web Administrator:

1. Ensure that all DECT Messenger modules are stopped.
2. Open Web Administrator by going to **Start menu > Programs > Avaya DECT Messenger > Web Administrator**
3. Login with your username and password. (The default username is admin and the default password is admin).
4. On the left-side tree, navigate to **Configuration > Expert > Import**.
5. If eKERNEL or eGRID are running on the server, an error message displays one of the following messages:

   `Error. Configuration database is in use (eKERNEL)` or `Error. Configuration database is in use (eGRID).`

   If this happens, verify that the mentioned module is shut down on the server.

6. Select the configuration database to import, as shown in the following figure.

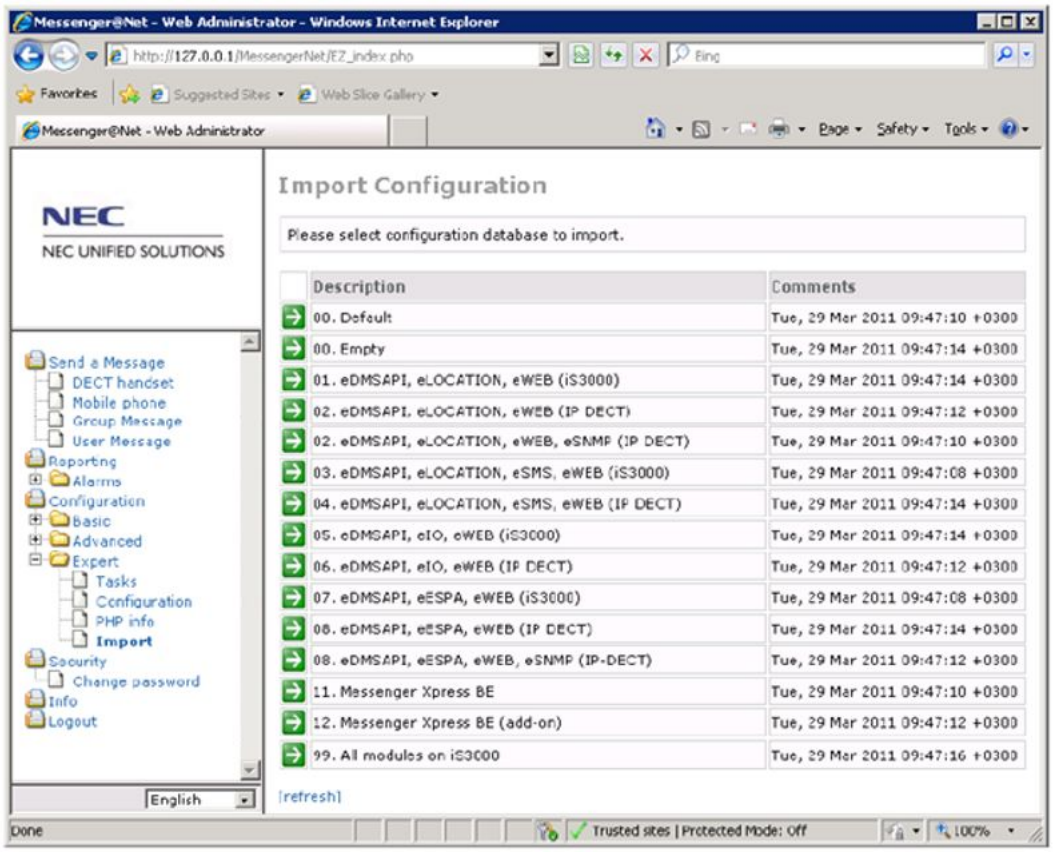


Figure 20: Web Administrator Import Configuration screen

7. A message appears stating the import was successful.

Important:

Standard configuration databases are not changed during the installation to match the environment. As a result, the file path and IP address of the DECT Messenger server PC are set to the default values by the installer. Be sure to use eGRID module to verify the values in the following tables and adjust them if necessary:

- eKERNEL_Site (columns: CFG_Connectionstring_CFG_str, CFG_Connectionstring_DATA_str, CFG_Log_path_str, CFG_eLOG_Path_str)
- eDMSAPI (columns eDMSAPI_API_address_str, eDMSAPI_External_address_str)
- eCSTA (column eCSTA_API_address_str)

Starting eKERNEL

To start eKERNEL, click the associated shortcut (**Start > Programs > Avaya DECT Messenger > eKERNEL**).

You can also use and adjust the shortcuts located at the `C:\SOPHO Messenger@Net\Lnk` directory or subdirectories.

When eKERNEL starts, it displays a main window similar to the one shown in the following figure:

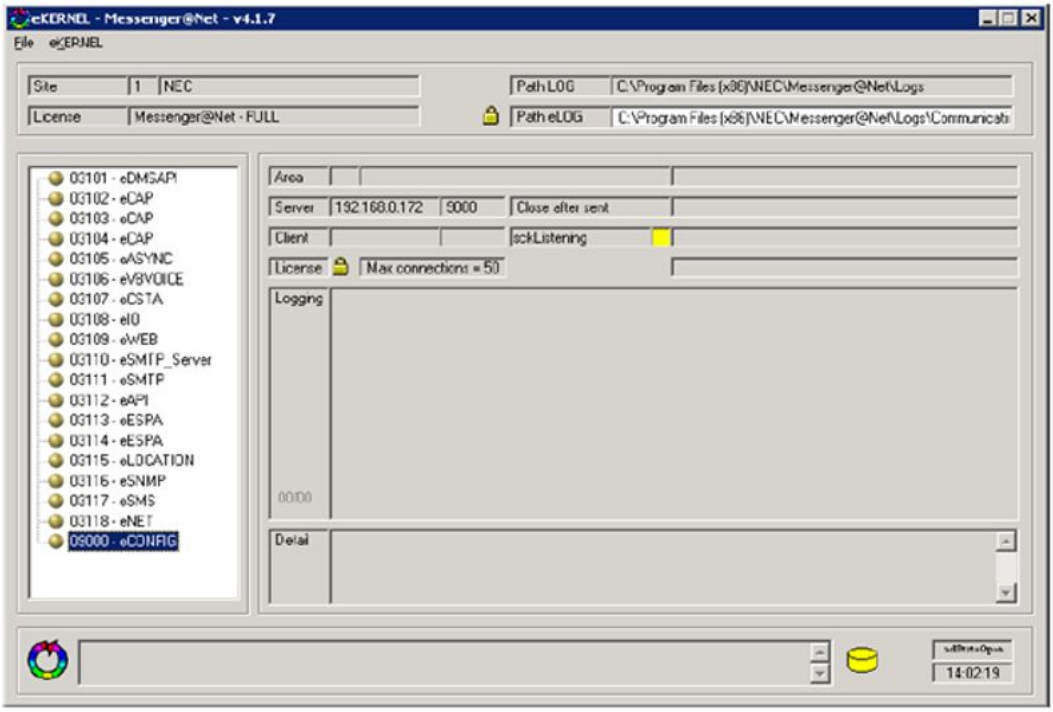


Figure 21: The eKERNEL main window

The list on the left indicates the configured modules, their status, and the connection information required to connect to eKERNEL.

Using eCONFIG (Configurator)

You can install and useCONFIG on the server or on one of the client PCs. However, when running from a client PC, you can only modify a limited number of properties (users, groups and devices). eCONFIG requires access to the configuration database, as well as an open TCP/IP connection to eKERNEL. Ensure that eKERNEL is running and configured to accept eCONFIG connections.

Note:

By default, eKERNEL listens for an eCONFIG connection on port 9000. The eCONFIG module needs to copy this database locally to make configuration changes.

Launching the eCONFIG configuration utility

1. Use the shortcut available in the Start Menu (**Start > Programs > Avaya DECT Messenger > eCONFIG**).
2. When prompted for login, enter the username and password. The default login that is available after installation is username: ‘admin’, password: ‘admin’.

3. When using eCONFIG from a client PC, you will be prompted for the location of MessengerConfig.mdb database. Browse to the network share of the server PC containing the **MessengerConfig.mdb** database.
4. If a working configuration database already exists (from a previous eCONFIG session), you will get a message box asking whether to use the locally modified copy of the MessengerConfig.mdb database previously used by eCONFIG, or to replace it with a fresh copy of the latest data, as shown in the following figure:

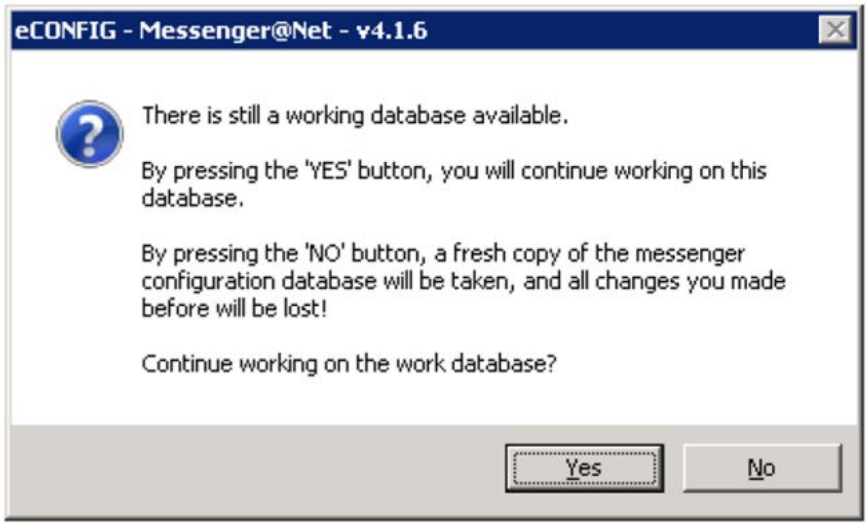


Figure 22: eCONFIG database message

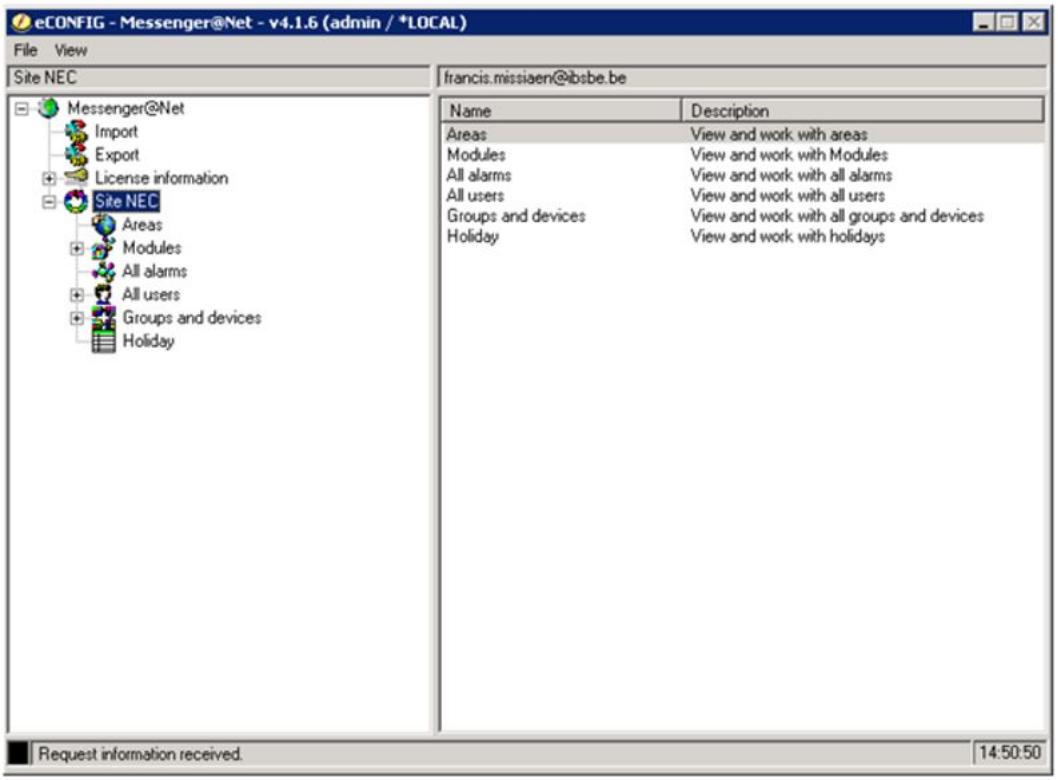
If you are not sure, choose **No**. This is the safest option; it discards the local (possibly outdated) database copy and replaces it with a copy of the server database.

5. If eKERNEL cannot be contacted, an error message appears as shown in the following figure:

Figure 23: eCONFIG and eKERNEL connection error

If this happens, verify that eKERNEL is running on the server PC and that the eKERNEL address and port number passed to eCONFIG correspond to those shown in the eKERNEL main window.

6. The main window appears, as shown in the following figure:



The left pane shows a tree with the available configuration items, while the right pane shows details on the currently selected configuration item.

Working with the eCONFIG user interface

- Left-click on any node in the tree to show more information on the right pane.
- Right-click on the nodes to show a pop-up menu with actions available for that node.
- Double-click on a node to show a screen where you can perform maintenance changes.
- Select **File > Exit** to close.

  You are asked whether to apply the changes (see Publishing or discarding changes on page 60).

Overview of DECT Messenger and eCONFIG concepts

A **site** represents a PC running a DECT Messenger Server (eKERNEL module). Normally, only one site should exist.

An **area** is a subdivision of a site. Multiple areas are normally only necessary when DECT Messenger is connecting to multiple iS3000 PBXs with DECT, or when using a mixed environment of iS3000/2000 IPS for DECT phones.

The **modules node** lists all modules configured, grouped by module type. Setting up module parameters requires advanced technical knowledge; in most cases, the provided defaults are sufficient. Refer to the Administrator Guide for detailed information on each module.

The **alarms** are notifications (also called messages) received by eKERNEL from input modules. Alarms received are processed according to rules specified in DECT Messenger and distributed using groups (containing group members) to output devices.

A more detailed presentation of concepts and their interaction is also available in the Administrator Guide.

Configuring site information

When using the default database configuration deployed by the installer, a Site NEC node should be visible in the left-side tree. You can use this site as a starting point of your configuration.

Configuring site information

1. Double-click the site node in the eCONFIG tree.

   The Site configuration window appears, as shown in the following figure:

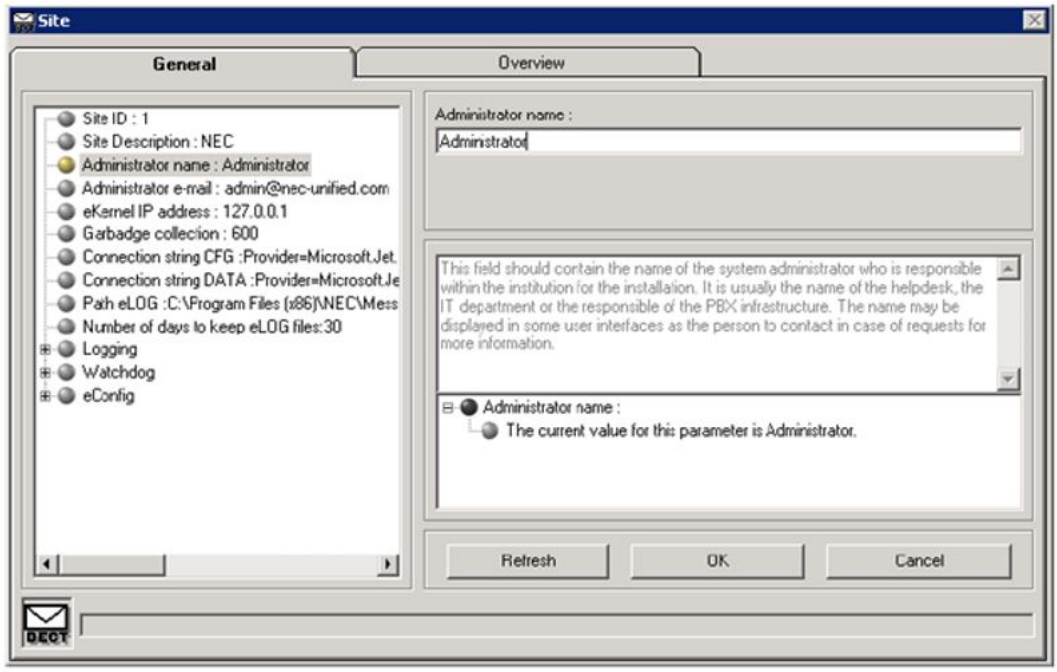


Figure 24: Site configuration window

2. From the list on the left, select the item to configure.
3. On the top-right, an input control allows you to modify the selected item.

4. On the middle-right, a description text explains the purpose of the selected item and how it should be configured.
5. On the bottom-right, the current value of the selected item is shown, as well as any validation errors related to the value entered in the input control.

   The following figure shows an example of a site configuration error:

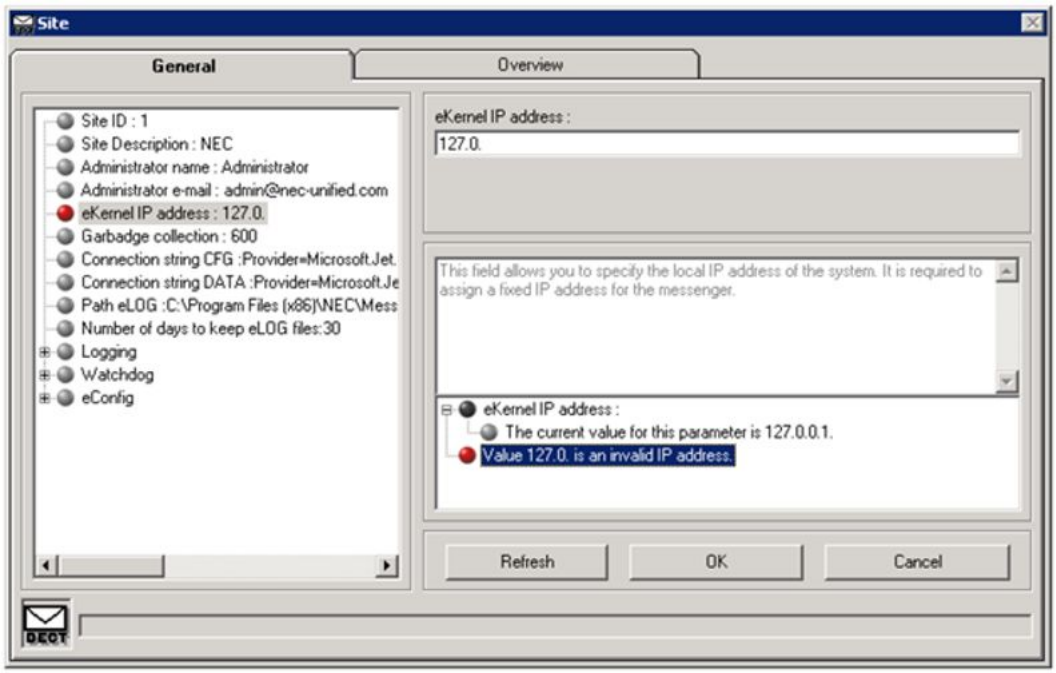


Figure 25: Site configuration error message

6. Enter a new value for the selected item.
7. If necessary, select another item and modify it using the same procedure.
8. Click **OK** to store the changes in the local copy of the database.

Note:

The type of input control (text box/combo box or grid) depends on the item selected.

Removing unused modules

The database configuration deployed by the installer contains a variety of input modules, already preset with working default parameters. For modules which you do not intend to use, or have not been installed, Avaya recommends to delete their configuration entry that appears in eCONFIG. This will avoid “could not launch module” errors that appear when using eTM (Task Manager) module during daily operation.

Deleting a module

1. Expand **Site NEC** node in the tree.
2. Expand the **Modules** node.
3. Under the **Modules** node, expand the node that represents the type of module which you do not intend to use.
4. Double-click the module name (e.g. ‘eESPA – area Hilversum’). The module details window appears – see Figure XY.
5. In the module configuration window, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the removal.

Configuring area information

When using the default database configuration deployed by the installer, two areas are already present. You can view them by expanding the **Site NEC** node and clicking the **Areas** node in the tree.

Modifying the description of an area

1. Expand the **Site NEC** node in the tree.
2. Select the **Areas** node.
3. Double-click the desired area listed in the right pane. The Area configuration window appears, as shown in the following figure:

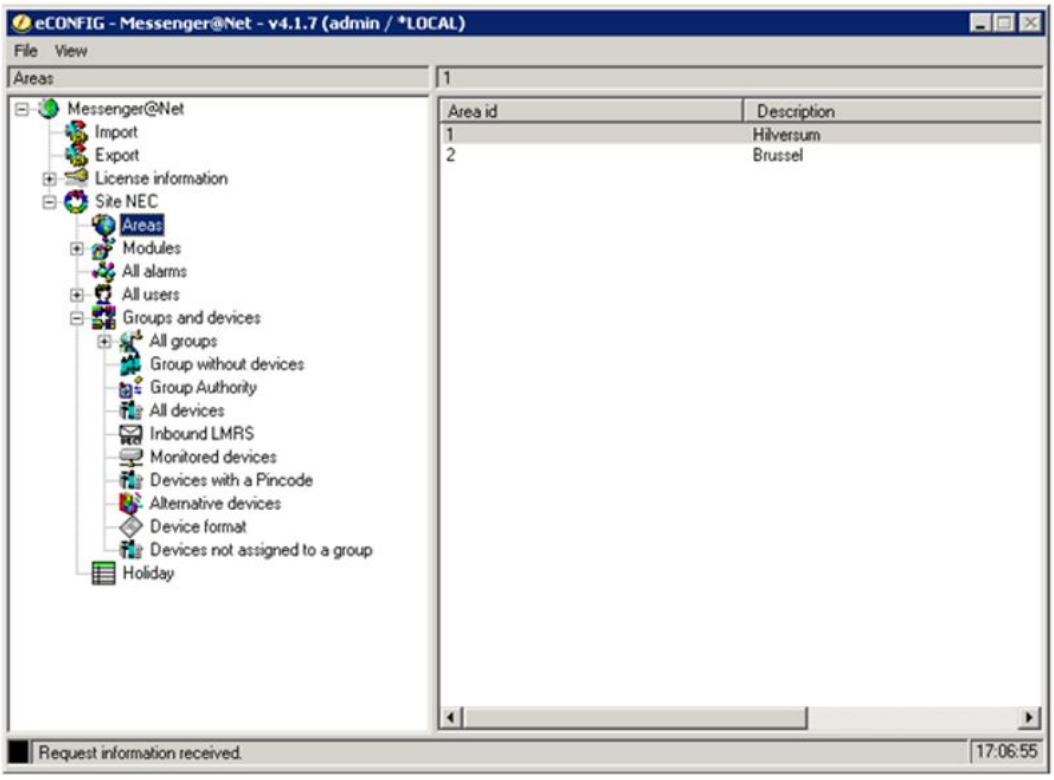


Figure 26: eCONFIG Area configuration window

4. Select the Description item and type a new description.
5. Click **OK** to save the changes.

Configuring PBX information

Note:

When using eDMSAPI, you also need to specify the IP address and port of the PBX (or IP-DECT DAP Controller PC) that the module should connect to.

Modifying the PBX information

1. Expand the **Modules** node.
2. Expand the **eDMSAPI** node.
3. Double-click the **eDMSAPI – area Hilversum** node to edit the preconfigured module.

   The module details window appears, as shown in the following figure:

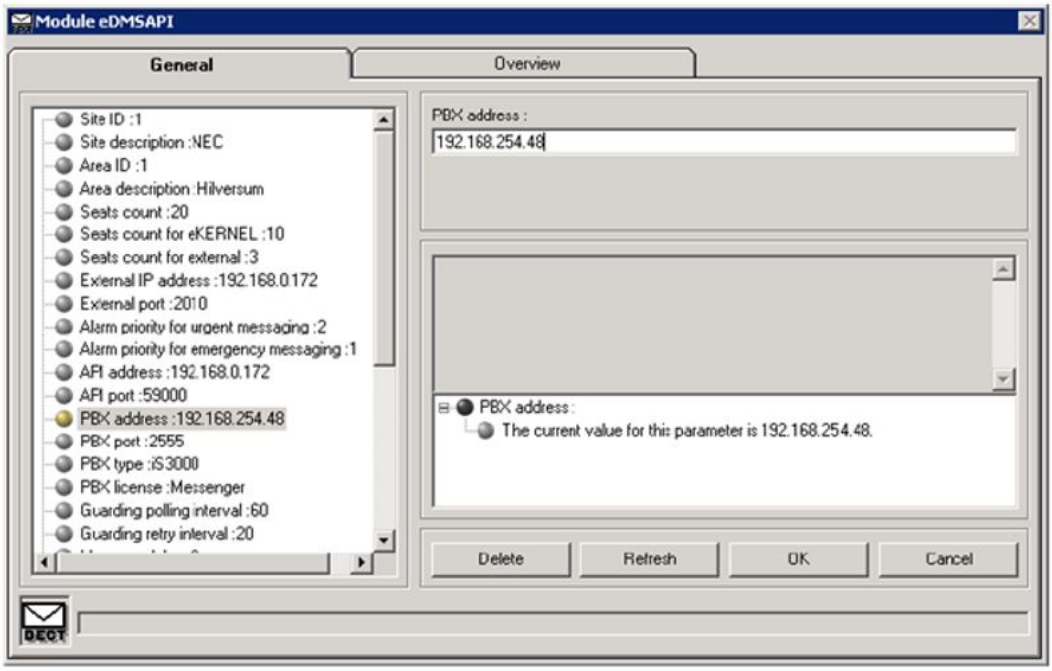


Figure 27: eDMSAPI module details window

4. Select and modify **PBX type** to indicate the type of device the module connects to (iS3000, Avaya or DAP Controller).
5. Select and modify **PBX address** to match the IP address of your target device.
6. Select and modify **PBX port** to match the port number of your target device.
7. Select and modify the **Seats count** according to the number of seat licenses available in your PBX.

   Note:

   Ensure that the seat count number plus the seats required by module eCSTA is equal or lower than the maximum number of seats licensed on the target PBX. Otherwise, you will not be able to make a LRMS message call.
8. Click **OK** to save the changes.

Adding a device

You require devices to output alarm messages notifications received from input modules. The following procedure describes how to add a device for sending alarm messages to DECT handsets. You can repeat the same procedure for other device types. You can also modify the sample devices installed by default with the product.

Adding a device

1. Expand the **Site NEC** node in the tree.
2. Expand **Groups and devices**.
3. Right-click **All devices** and select **New Device**.

   The device details screen appears, as shown in the following figure:

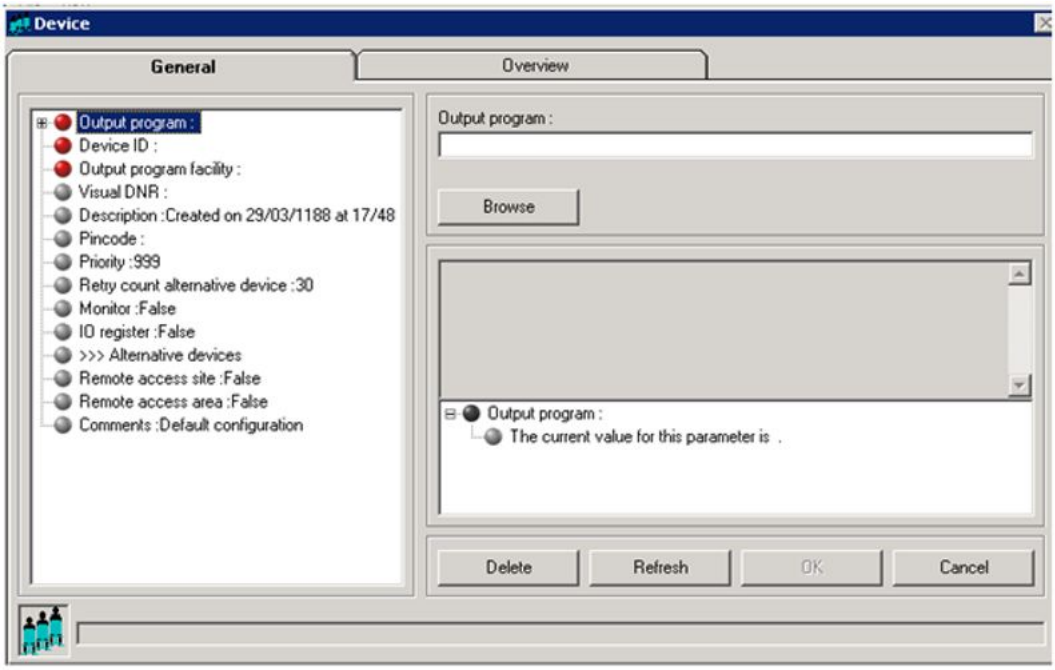


Figure 28: Device details window

4. Select the **Output Program** and click **Browse**.

   The Select Output Program window appears, as shown in the following figure:

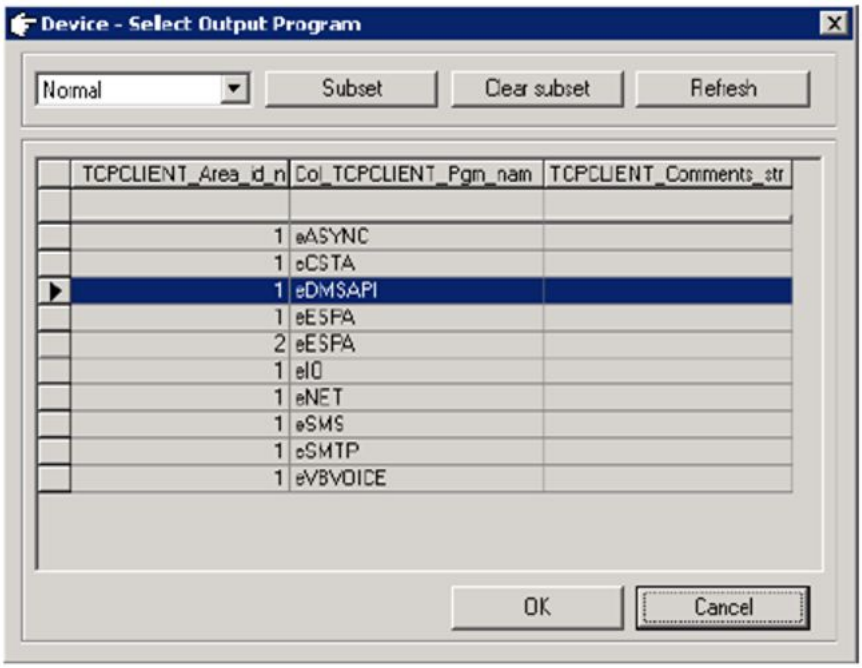


Figure 29: Select Output Program window

5. Select **eDMSAPI** by clicking the black arrow and click **OK**.
6. Select and specify a **Device ID** (the extension number, e.g. ‘2000’).
7. Select the **Output program facility** and click **Browse**.

   The Select Facility window appears, as shown in the following figure:

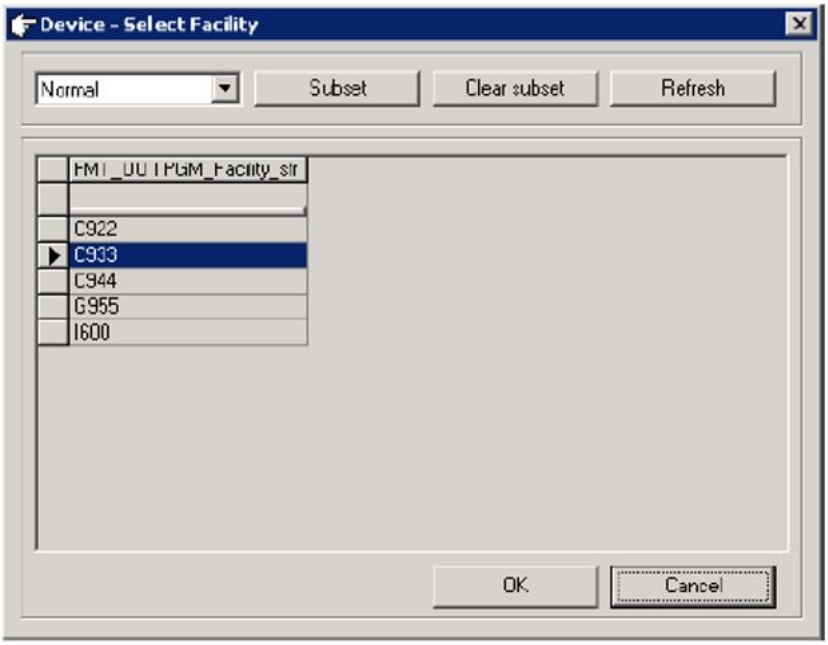


Figure 30: Select Facility window

8. Select a handset model and click **OK**.
9. Select and specify other properties if necessary (e.g. 'Description').
10. Click **OK** to save the new device.

Several devices can be added to a group, as group members. Working with groups and group members, as well as further details on supported output devices can be found in *DECT Messenger Fundamentals, NN43120–120*.

Publishing or discarding changes

When you are working with eCONFIG, changes to users, devices, and groups are published immediately. However, structural changes to sites, areas, and modules require the local database to be copied back over the production database on the DECT Messenger Server.

Note:

Publishing overwrites any changes made to the production database after eCONFIG created its local copy during start up. You must ensure that users did not make changes to the production database after the local copy was created.

When you want to end the configuration process, select **File > Exit** from the eCONFIG main menu. You are then asked if you want to apply the configuration changes. Click **Yes** to apply the changes or **No** to cancel.

Before you can publish the new configuration, you must ensure the current database is not in use. This means that all DECT Messenger modules must be stopped. Before stopping a production environment, take into consideration that all pending alarms are cleared at restart.

Also, while eKERNEL and associated modules are down, no input and output is performed, and alarm input and distribution is suspended.

Stopping active modules

1. On the server PC (where the database is located), check if eTM (Task Manager) module is running.
   - If eTM is not running, close every running module by clicking the box in the top-right corner of each module main window.
   - If eTM is running, right-click its icon in the Windows taskbar notification area and choose **Stop**. Then close any other running modules by clicking the box in the top-right corner of each module main window.
2. Right-click the **eTM** icon again and select **Exit**.

   A confirmation dialog appears.
3. Click **OK** to confirm.

Publishing changes

1. When asked whether to apply the new configuration, click **Yes**.
2. A warning message appears, advising you to close down eKERNEL and all other running DECT Messenger modules, as shown in the following figure:

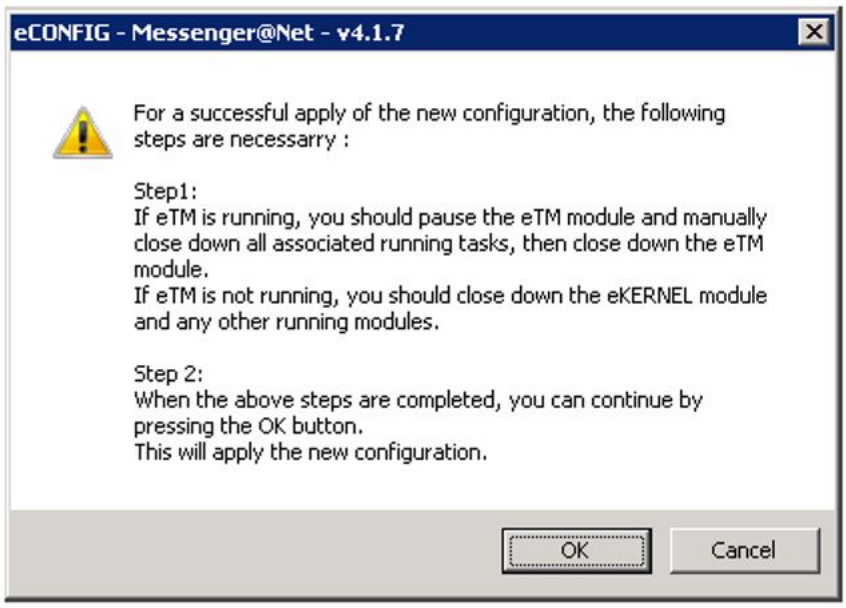


Figure 31: eCONFIG warning message

3. Ensure that all modules are stopped (refer to procedure above).
4. Click **OK** to apply the configuration.

   eCONFIG copies the local MessengerConfig.mdb database over the production configuration database. If there are errors copying the database, a warning box prompts you to close all applications and retry.

5. When the database is successfully published, a message appears stating that the registry files necessary for eTM module are created, as shown in the following figure:

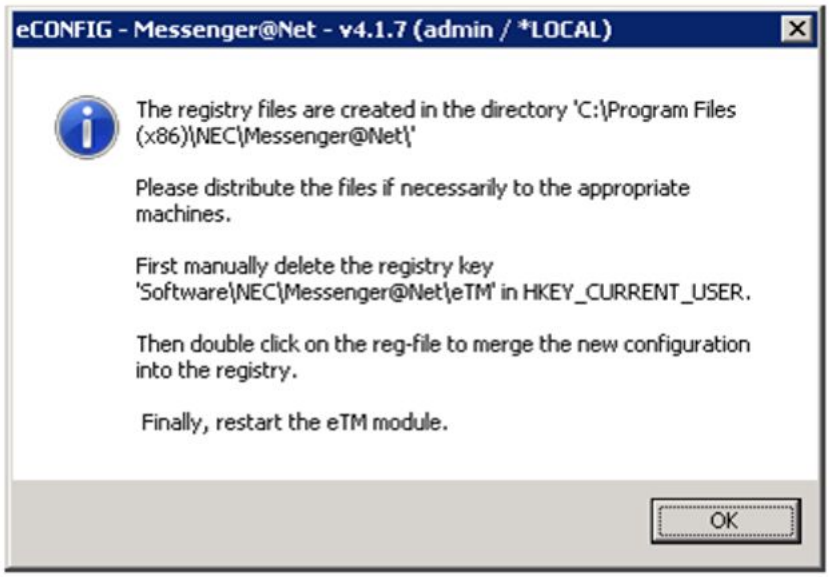


Figure 32: Successful publication window

6. Open the registry editor:
   - **Start menu > Run**
   - Type `regedit`
7. Locate the `HKEY_CURRENT_USER\Software\Avaya\Avaya DECT Messenger\eTM` registry key and delete it.
8. Locate the registry file in the directory mentioned in the message (by default, `C:\Program Files\ Avaya\Avaya DECT Messenger \Maintenance \Registry`).
9. Identify the file(s) corresponding to the DECT Messenger Server environment (Check the file names; they should be in the format eTM – Site <id> – Environment LOCAL.reg). Double-click the registry file(s) to merge the information into the server registry.

   A Registry Editor confirmation message appears asking for permission to merge the data into the registry.
10. Click **Yes** to confirm.

    A message box appears informing that the merge was successful.
11. Click **OK** to acknowledge and continue.

Note:

When configured for using a distributed environment with additional DECT Messenger clients installed on other PCs, eCONFIG generates additional registry files for each environment. You must copy these files and repeat the above steps

on the client PC corresponding to that specific environment, after stopping eTM and other associated processes.

12. Start eTM to bring the environment online.

    The following figure shows an example of eTM running configured modules:

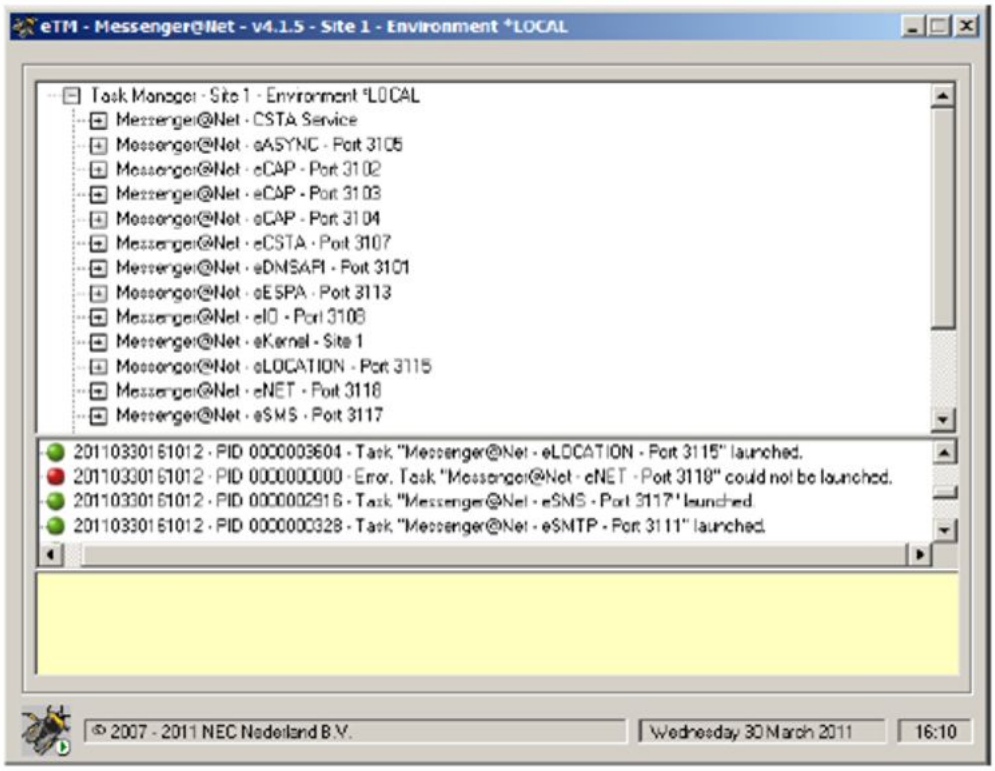


Figure 33: eTM with running modules

Discarding changes

1. When asked whether to apply the new configuration, click **No**.

   A window appears asking you to confirm that you want to exit the application.

2. Click **Yes**.

   Any changes are discarded without being published.

Using eCONFIG in a distributed environment

When eCONFIG is on the server PC, the path to the configuration database in the INI file is automatically populated by the installation program. You can run eCONFIG on another PC, but you can only add or modify users, groups and devices.

When running in a distributed environment with eCONFIG installed on a client PC, you require access to the configuration database located on the server PC. This means that the directory

where the MessengerConfig.mdb resides on the DECT Messenger Server must be shared over the network to enable remote machines to access the shared database.

Note:

Contact your network administrator for details on security and access privileges over the network.

To avoid browsing for the location of MessengerConfig.mdb, you can specify its location in the eCONFIG.ini file.

Configuring the eCONFIG.ini

1. Open Windows Explorer.
2. Browse to the directory with the eCONFIG executable (by default this is `C:\Program Files\Avaya\Avaya DECT Messenger\Configurator`).
3. Double-click the file to open it with the default text editor (typically Notepad).
4. Modify the **MessengerConfig** setting to indicate the location of MessengerConfig.mdb configuration database.
5. Save the changes and close the text editor.

   For example, if directory `C:\Program Files\ Avaya\Avaya DECT Messenger \Database` is shared as **MyNetworkShare** on eKERNEL system MyMessengerServer, the configuration setting would look as pictured in the following figure:

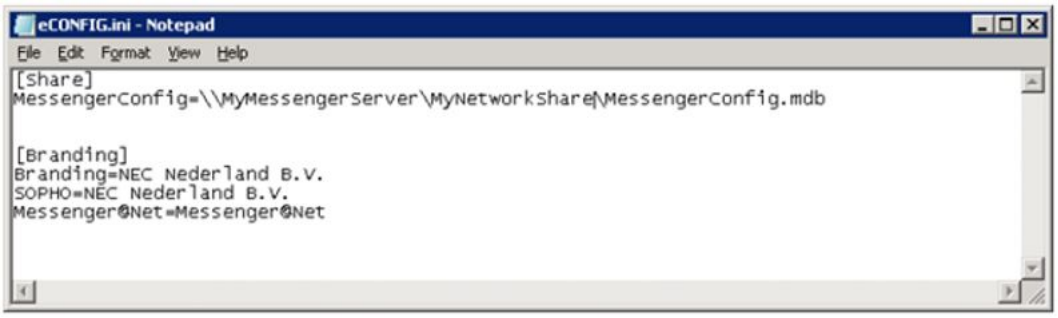


Figure 34: Example of an eCONFIG.ini file

Important:

To meet the requirement of eCONFIG to support both NEC and AVAYA environments, additional tags are available in the **eCONFIG.INI** file. These tags provide internal steering parameters for the user interface of the eCONFIG instance. Do not alter or remove these statements. Tampering with branding related information may violate copyright regulations.

Chapter 10: Module eAPI

Introduction

The module eAPI is not a real module, but rather a description of a public Application Program Interface (API) for third-party developers who want to communicate with DECT Messenger. This chapter is intended for developers who want to build an interface to the eKERNEL module.

The objective of this document is to describe how developers can integrate applications with DECT Messenger. Note that the eAPI interface has limited capabilities. An alternative to developing your own program is to contact Avaya and request the development of an integrated solution.

Limitations

Input program functionality only

The functionality implemented in the eAPI interface is limited to the sending of message requests to the eKERNEL module. This process is carried out through so-called message request (msgrqs) transactions. Therefore, third-party application programs that are created using eAPI technology are limited to input program functionality only.

No central configuration

A second limitation in eAPI is that there currently is no support for configuration request messages. In all other modules, there is a central configuration database, where all relevant parameters are centrally administered. This process is normally carried out through configuration requests (cfgrqs) from the module to eKERNEL and configuration replies (cfgrpy) from eKERNEL to the module. As a result, third-party developers must provide their own

configuration techniques (through registry, .INI files, database, command line parameters, and so on) to control the behavior of their applications.

Basic architecture

The architecture of eAPI is embedded in the eKERNEL module. The eAPI interface refers to the ability of eKERNEL to provide a TCP server, which listens to a specified port, and receives TCP sockets packages that contain message requests.

Therefore, when building an eAPI-based third-party product, you require an application that acts as TCP client and establishes a sockets connection to the eKERNEL module, which acts as TCP server.

Depending on the eKERNEL settings, the sockets connections are kept open or are closed after reception of a request. When the socket is kept open, the port remains allocated to the connected client. This is suitable for implementations where a dedicated connection is required. If multiple clients must address the same eKERNEL port, Avaya recommends that you close the socket after each ad hoc request. With this approach, a single port can serve to accept message requests from multiple input sources.

Message format

Message requests to eKERNEL must be formatted according to specific rules. A sample request is illustrated in Figure 35: Sample message request on page 66.

```
<xml><msgrqs><set_or_reset>*SET</
set_or_reset><group>1</group><alarmdescr>2</
alarmdescr><msg>3</msg><remove_after>*SENT</
remove_after></msgrqs></xml>
```

Figure 35: Sample message request

The following rules apply:

- The string must start with <xml><msgrqs> and end with </msgrs></xml> tags
- At the end of the string, a carriage return (ASCII 13) and line feed (ASCII 10) must be appended
- The message request must contain 5 parameters
  - The parameter set_or_reset must start with <set_or_reset> tag and end with the </set_or_reset> tag
  - The parameter group must start with <group> tag and end with the </group> tag

- The parameter alarmdescr must start with <alarmdescr> and end with the </alarmdescr> tag
- The parameter msg must start with <msg> tag and end with the </msg> tag
- The parameter remove_after must start with <remove_after> tag and end with the </remove_after> tag

- The parameter set_or_reset can supports the following values: *SET or *RESET
- The parameter group refers to a configured group defined in the eKERNEL_GROUP table
- The parameter alarm_descry refers to a configured alarm description, defined in the eKERNEL_ALARM table
- The parameter remove_after supports the following values: *SENT, *RESET or *CALC

Refer to the appropriate chapters of this document for more information on the tables.

Introduction to a sockets client

Refer to the documentation of your development environment for more information on sockets programming.

The code sample shown in Figure 36: Sample socket client code on page 69 describes an introduction for beginner programmers on how to build a very simple Visual Basic program that contacts the DECT Messenger eKERNEL module and delivers a message request. Note that the source code is provided for illustration only, and does not include error recovery.

Creating a basic sockets client using Visual Basic

Creating a basic sockets client using Visual Basic

1. Start Visual Basic, and open a new project of Standard .EXE type. In the menu, choose **Project > Components** and add the Microsoft Winsock Control component to the project. This component usually refers to C:\WINNT \system32\MSWINSCK.OCX.
2. Drag a CommandButton control to the form. You can use the default name Command1.
3. Drag a Winsock control to the form. You can use the default name Winsock1.
4. Add the code shown in Figure 36: Sample socket client code on page 69 in the Private Sub Command1_Click.
5. Specify the correct IP address (the IP address of the system where eKERNEL runs) and port number (the configured port for eAPI, as defined in eKERNEL_TCPCLIENT table).

6. Run the program. If you click the Command1 button, a message request is sent to eKERNEL.
7. You can alter the code shown in Figure 36: Sample socket client code on page 69 to specify the correct parameters for the parameters group (use one of the values specified in the eKERNEL_GROUP table), alarm description (use one of the values specified in the eKERNEL_ALARM table), and so on.

Note:

The code shown in Figure 36: Sample socket client code on page 69 is not intended to represent a reliable TCP client, and is meant only to illustrate how to start programming with eAPI using minimal code entry. A real-life program must take all necessary action to handle all error conditions.

The following issues usually require improvement:

- The sample code shown in Figure 36: Sample socket client code on page 69 does not respond on the asynchronous connection attempt by means of the Winsock1_Connect() event. The code assumes that the connect succeeds after doing a DoEvents(). The Winsock1.State must be 7 before a SendData can be requested.
- The sample code shown in Figure 36: Sample socket client code on page 69 includes appropriate error recovery, but does not respond to failed connection attempts.
- The sample code shown in Figure 36: Sample socket client code on page 69 assumes the data is actually transmitted with the SendData, and does not wait for the Winsock1_SendComplete() event.
- The values for IP address and port are hard-coded, and users must be able to set them as parameters in a real-word program.
- The values in the message request are hard-coded, and must be filled with actual alarm information and appropriate configured values, as defined in the configuration database.

```
Private Sub Command1_Click()
' close the socket
Winsock1.Close
' specify eKERNEL protocol, ip addres and port (hardcoded)
Winsock1.Protocol = sckTCPProtocol
Winsock1.RemoteHost = "10.110.50.138"
Winsock1.RemotePort = 3204
' connect to the eKERNEL server
Winsock1.Connect
' we wait indefinitely until asynchronous connect before sending
Do
Select Case Winsock1.State
Case Is <> 7
DoEvents
Case Else
' send the data when connect completes
Winsock1.SendData
"<xml><msgrqs><group>1</group><alarmdescr>2</alarmdescr><message>3</m
essage><set_or_reset>*set</set_or_reset><remove_after>*sent</remove_a
fter></msgrqs><xml>" + Chr$(13) + Chr$(10)
' allow asynchronous event to complete to purge data
DoEvents
Exit Do
End Select
Loop
' close the socket
Winsock1.Close
End Sub
```

Figure 36: Sample socket client code

More extended program

Refer to Module - eAPI sample on page 71 for a detailed source code listing of a more complete implementation of a Visual Basic program that implements eAPI functionality. The compiled program eAPI.exe and the source code eAPI.zip (zipped) are shipped with the DECT Messenger and the .exe is installed when you select eAPI module during custom install.

Note that this code is provided on as-is basis, and is not intended to be used without modification. Usage of the code is the responsibility of the third-party developer, as all aspects needed to make the code reliable are not implemented.

The eAPI program is designed to provide the same look and feel as is found in other DECT Messenger modules.

Some typical features include:

- Ability to specify certain runtime parameters of the program by means of the command line parameters in the shortcut, such as: /Site:2 /eKernel address:*LOCAL /eKernel port: 3209 /Log drive:C
- A menu that provides a queue (list) that the module can use to handle situations in which eKERNEL is temporarily unavailable.
- Logging facilities on-screen, with the option to left-click a log entry to see details.
- Logging facilities to disk, in the same directory structure mechanism as used for all other modules.

Real-world examples

Using eAPI, you can write external applications in your language of choice (Visual Basic, C++, Java, and so on). These applications can collect alarm information from external systems, for example by means of asynchronous communications or a network connection.

It is important however to realize that the scope of the eAPI interface to eKERNEL is limited, and there is for instance no ability to give feedback to eAPI (and the alarm system) upon successful or failed message delivery within DECT Messenger.

Avaya recommends investigating alternatives, such as reusing an existing module of DECT Messenger (for example, eCAP generic) or contacting Avaya to request the development of a new integrated module. There is a road-map procedure within the Avaya group that keeps track of all new requirements.

Chapter 11: Module - eAPI sample

```
eAPI_form - 1
Option Explicit
' ---------------------------------------
' This program requires a valid command$
' ---------------------------------------
' /Site:1
' /eKernel address:*LOCAL or value xxx.xxx.xxx.xxx
'/eKernel port:2001
' /Log drive:C
' ---------------------------------------
Private Function parse_cmd_line(keyword As String) As String
' This routine isolates the value of a keyword from the command$
Dim lcl_cmd As String
Dim lcl_str As Integer
Dim lcl_end As Integer
On Error Resume Next
Err = 0
lcl_cmd = g_command
lcl_str = InStr(1, UCase(lcl_cmd), / & UCase(keyword) & :)
If lcl_str = 0 Then
parse_cmd_line = N/A
log S, INF, Warning : parameter ' & keyword & ' not available in ' & lcl_cmd & '

Else
lcl_end = InStr(lcl_str + 1, UCase(lcl_cmd) +  /, /)
parse_cmd_line = Mid$(g_command + Space$(5), lcl_str + 2 + Len(keyword), lcl_end -
lcl_st
r - Len(keyword) - 3)
End If
If Err Then
MsgBox (Err.Description &  - Unexpected error in parse_cmd_line() function)
log E, ERR, Err.Description &  - Unexpected error in parse_cmd_line() function
End If
On Error GoTo 0
End Function
Private Function parse_xml(keyword As String, xml As String) As String
' Isolates the 'value' for a 'keyword' from a 'xml' string
' When no value is found, 'N/A' is returned
On Error Resume Next: Err = 0
Dim lcl_start As Integer
Dim lcl_end As Integer
Dim lcl_from As String
Dim lcl_to As String
Dim lcl_value As String
lcl_from = LCase$(< & keyword & >)
lcl_to = LCase$(</ & keyword & >)
lcl_start = InStr(1, LCase$(xml), lcl_from)
lcl_end = InStr(lcl_start + Len(lcl_from), LCase$(xml), lcl_to)
lcl_value = Mid$(xml, lcl_start + Len(lcl_from), 1 + lcl_end - lcl_start -
Len(lcl_to))
If Err Then
parse_xml = N/A
```

```
log S, INF, Warning : parameter ' & keyword & ' not available in ' & xml & '

Else
parse_xml = lcl_value
End If
On Error GoTo 0
End Function
Private Sub lab_message_Click()
End Sub
Private Sub cmd_transmit_Click()
Dim lcl_xml As String
' Validate
If Trim$(txt_group) =  Then
lab_msg =  Error. Group must be entered.
txt_group.SetFocus
Exit Sub
End If
If Trim$(txt_alarmdescr) =  Then
lab_msg =  Error. Alarm description must be entered.
txt_alarmdescr.SetFocus
eAPI_form - 2
Exit Sub
End If
If Trim$(txt_msg) =  Then
lab_msg =  Error. Message must be entered.
txt_msg.SetFocus
Exit Sub
End If
' Build XML string
lcl_xml = <xml><msgrqs> '<site> & g_site & </site>
lcl_xml = lcl_xml + <set_or_reset> & cbo_set_or_reset & </set_or_reset>
lcl_xml = lcl_xml + <group> & Trim$(txt_group) & </group>
lcl_xml = lcl_xml + <alarmdescr> & Trim$(txt_alarmdescr) & </alarmdescr>
lcl_xml = lcl_xml + <msg> & Trim$(txt_msg) & </msg>
lcl_xml = lcl_xml + <remove_after> & cbo_remove_after & </remove_after>
lcl_xml = lcl_xml + </msgrqs></xml>
' Submit request
eAPI_form.lst_ekernel_outq.AddItem lcl_xml
' Inform user
lab_msg =  Message submitted to eKERNEL.
End Sub
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Dim lcl_o As String
' Submit <pgmsts> shutdown request to ekernel if connected
On Error Resume Next: Err = 0
If ip_ekernel.State = 7 Then
lcl_o = <xml><pgmsts><value>Shutdown</value></pgmsts></xml>
ip_ekernel.SendData lcl_o + Chr$(13) + Chr$(10)
If Err Then
lab_msg =  Error  & Err &  -  & Err.Description
log E, ERR, TCP senddata error  & Err &  -  & Err.Description &  -  & l
cl_o &  could not be sent to eKERNEL
Else
lst_ekernel_outq.RemoveItem 0
log O, TCP, lcl_o
End If
On Error GoTo 0
End If
DoEvents
' log
log S, INF, Application ended
' end
End
```

```
End Sub
Private Sub lst_log_DblClick()
' show details
On Error Resume Next: Err = 0
txt_log.Text = lst_log.List(lst_log.ListIndex)
On Error GoTo 0
show_pages
End Sub
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
'F3=Exit
If KeyCode = 114 Then
Unload Me
'End
End If
End Sub
Private Sub Form_Load()
Dim lcl_rc As String
Dim lcl_o
Dim lcl_version
Dim Lcl_Msg As String
Dim lcl_h As Integer
' Set application title
Me.Caption = eAPI - SOPHO Messenger@Net - v & App.Major & . & App.Minor & . & App.R
evision
' Startup values required to enable logging
g_log_path = D:\SOPHO Messenger@Net
g_log_days = 14
lab_log_path =   & g_log_path
lab_log_days =   & g_log_days
' Default command line parameters
eAPI_form - 3
' /Site:1 /eKernel address:*LOCAL /eKernel port:3209 /Log drive:C
g_command = Command$
If g_command =  Then
Lcl_Msg = Warning: eAPI is started without command line parameters. + Chr$(10) + Ch
r$(10)
Lcl_Msg = Lcl_Msg + Check the command string in the target value in the properties o
f the shortcut. + Chr$(10) + Chr$(10)
Lcl_Msg = Lcl_Msg + Please confirm to start this session with the following replacem
ent values: + Chr(10) + Chr$(10)
g_command = /Site:2 /eKernel address:*LOCAL /eKernel port:3209 /Log drive:C
lcl_rc = InputBox(Lcl_Msg, Me.Caption, g_command)
If lcl_rc =  Then
End
Else
g_command = lcl_rc
End If
End If
'Initialise screen labels
lab_ekernel_remote_address =  N/A
lab_ekernel_remote_port =  N/A
lab_ekernel_local_address =  N/A
lab_ekernel_local_port =  N/A
' Get command line parameter
g_site = parse_cmd_line(Site)
g_ekernel_remote_address = parse_cmd_line(eKernel address)
g_ekernel_remote_port = parse_cmd_line(eKernel port)
g_log_drive = parse_cmd_line(Log drive)
' Handle special values
If g_ekernel_remote_address = *LOCAL Then g_ekernel_remote_address =
ip_ekernel.LocalIP
' Start
log S, INF, Application  & Me.Caption &  started with parameters  & g_command
```

```
' Terminate if undefined values
If g_site = N/A Then
lcl_rc = MsgBox(eAPI could not start. Parameter '/Site:xxx' missing in command strin
g., vbCritical, eAPI - SOPHO Messenger@Net)
Unload Me
End If
If g_ekernel_remote_address = N/A Then
lcl_rc = MsgBox(eAPI could not start. Parameter '/eKernel address:xxx.xxx.xxx.xxx' m
issing in command string., vbCritical, eAPI - SOPHO Messenger@Net)
Unload Me
End If
If g_ekernel_remote_port = N/A Then
lcl_rc = MsgBox(eAPI could not start. Parameter '/eKernel port:xxxxx' missing in com
mand string., vbCritical, eAPI - SOPHO Messenger@Net)
Unload Me
End If
If g_log_drive = N/A Then
lcl_rc = MsgBox(eAPI could not start. Parameter '/Log drive:x' missing in command st
ring., vbCritical, eAPI - SOPHO Messenger@Net)
Unload Me
End If
If Len(g_log_drive) <> 1 Then
lcl_rc = MsgBox(eAPI could not start. Parameter '/Log drive:x' is invalid in command
string., vbCritical, eAPI - SOPHO Messenger@Net)
Unload Me
End If
' Update screen labels
lab_ekernel_remote_address =   & g_ekernel_remote_address
lab_ekernel_remote_port =   & g_ekernel_remote_port
' Initialise eAPI screen fields
With cbo_set_or_reset
.Clear
.AddItem *SET
.AddItem *RESET
.ListIndex = 0
End With
With cbo_remove_after
.Clear
.AddItem *SENT
.AddItem *RESET
.AddItem *CALC
eAPI_form - 4
.ListIndex = 0
End With
' Set socket state indicator to defaults
lab_ekernel_state.Backcolour = RGB(0, 0, 0)
' Show copyright
lab_msg =   & App.LegalCopyright
' Initialise CFGRQS variables
g_log_path = g_log_drive + :\SOPHO Messenger@Net
g_log_days = 14
lab_log_path =   & g_log_path
lab_log_days =   & g_log_days
' Ininitialise guarding
g_guarding = Timer
' Enable timer for eKernel
tim.Interval = 100
tim.Enabled = True
End Sub
Private Sub ip_ekernel_Connect()
Dim lcl_version As String
Dim lcl_o As String
' Update screen
```

```
g_ekernel_local_address = ip_ekernel.LocalIP
lab_ekernel_local_address =   & g_ekernel_local_address
g_ekernel_local_port = ip_ekernel.LocalPort
lab_ekernel_local_port =   & g_ekernel_local_port
' log S, INF, TCP local port  & Format$(g_ekernel_local_port, 00000) &  connected
with remote port  & Format$(g_ekernel_remote_port, 00000) &  (eKERNEL)
End Sub
Private Sub ip_ekernel_DataArrival(ByVal bytesTotal As Long)
' ip data received
lab_msg =  Data arrival -  & bytesTotal &  bytes received from eKERNEL
Dim lcl_i As String
ip_ekernel.GetData lcl_i, vbString
' Append to buffer, and isolate a valid <xml>xxxx</xml> sockets data stream
g_ekernel_buffer = g_ekernel_buffer + lcl_i
Dim lcl_str_xml As Integer
Dim lcl_end_xml As Integer
Dim lcl_dta_xml As String
' Begin Loop
Do
' Check if <xml> string occurs
lcl_str_xml = InStr(g_ekernel_buffer, <xml>)
' Incomplete block without <xml> is not yet processed
If lcl_str_xml = 0 Then Exit Do
' Check if </xml> string occurs
lcl_end_xml = InStr(lcl_str_xml, g_ekernel_buffer, </xml> + Chr$(13) + Chr$(10)
)
' Incomplete block without </xml> is not yet processed
If lcl_end_xml = 0 Then Exit Do
Both <xml> and </xml> tags are found, isolate this data stream
lcl_dta_xml = Mid$(g_ekernel_buffer, lcl_str_xml, (lcl_end_xml - lcl_str_xml) + 8
)
' Keep remainder of this data stream (if any is available)
g_ekernel_buffer = Mid$(g_ekernel_buffer, lcl_str_xml + Len(lcl_dta_xml))
' Add to listbox
log I, TCP, lcl_dta_xml
' Submit request to ekernel jobqueue
lst_ekernel_jobq.AddItem lcl_dta_xml
' End loop
Loop
End Sub
Private Sub ip_ekernel_Error(ByVal number As Integer, Description As String, ByVal
Scode As L
ong, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDispl
ay As Boolean)
lab_msg =  Error  & number &  -  & Description
log E, ERR, TCP error  & number &  -  & Description &  (eKERNEL)
End Sub
Private Sub mnu_ekernel_disconnect_Click()
ip_ekernel.Close
lab_ekernel_state.Backcolour = RGB(0, 0, 0)
g_ekernel_local_address = N/A
g_ekernel_local_port = N/A
eAPI_form - 5
lab_ekernel_local_port =   & g_ekernel_local_port
lab_ekernel_local_address =   & g_ekernel_local_address
End Sub
Private Sub process_ekernel()
Dim lcl_o As String
Dim lcl_version As String
' Handle sockets status - continuously attempt to stay connected
Dim lcl_ekernel_cur_state As Integer
lcl_ekernel_cur_state = ip_ekernel.State
```

```
If lcl_ekernel_cur_state <> g_ekernel_prv_state Then
g_ekernel_prv_state = lcl_ekernel_cur_state
Select Case lcl_ekernel_cur_state
Case 0
lab_ekernel_msg =  Closed
lab_ekernel_state.BackColor = RGB(0, 0, 0)
Case 1
lab_ekernel_msg =  Open
lab_ekernel_state.BackColor = RGB(200, 130, 0)
Case 2
lab_ekernel_msg =  Listening
lab_ekernel_state.BackColor = RGB(255, 255, 0)
Case 3
lab_ekernel_msg =  Connection pending
lab_ekernel_state.BackColor = RGB(200, 130, 0)
Case 4
lab_ekernel_msg =  Resolving host
lab_ekernel_state.BackColor = RGB(200, 130, 0)
Case 5
lab_ekernel_msg =  Host resolved
lab_ekernel_state.BackColor = RGB(200, 130, 0)
Case 6
lab_ekernel_msg =  Connecting
lab_ekernel_state.BackColor = RGB(200, 130, 0)
Case 7
lab_ekernel_msg =  Connected
lab_ekernel_state.BackColor = RGB(0, 200, 0)
Case 8
lab_ekernel_msg =  Closing
lab_ekernel_state.BackColor = RGB(200, 130, 0)
Case 9
lab_ekernel_msg =  Error
lab_ekernel_state.BackColor = RGB(128, 0, 0)
Case Else
End Select
End If
' Only process if ekernel_outq is populated
If lst_ekernel_outq.ListCount = 0 Then Exit Sub
' Not yet connected
If ip_ekernel.State <> 7 Then
On Error Resume Next
Err = 0
If ip_ekernel.State <> sckClosed Then ip_ekernel.Close
g_ekernel_local_address = N/A
g_ekernel_local_port = N/A
lab_ekernel_local_address =   & g_ekernel_local_address
lab_ekernel_local_port =   & g_ekernel_local_port
ip_ekernel.RemoteHost = g_ekernel_remote_address
ip_ekernel.RemotePort = g_ekernel_remote_port
ip_ekernel.Connect
DoEvents
Exit Sub
On Error GoTo 0
End If
' Connected
g_ekernel_local_address = ip_ekernel.LocalIP
g_ekernel_local_port = ip_ekernel.LocalPort
lab_ekernel_local_address =   & g_ekernel_local_address
lab_ekernel_local_port =   & g_ekernel_local_port
'
--------------------------------------------------------------------------
' Handle requests in ekernel jobqueue
'
```

```
------------------------------------------------------------------------------
While lst_ekernel_jobq.ListCount > 0
process_ekernel_jobq lst_ekernel_jobq.List(0)
lst_ekernel_jobq.RemoveItem 0
eAPI_form - 6
Wend
'
------------------------------------------------------------------------------
' Handle requests in ekernel outq
'
------------------------------------------------------------------------------
On Error Resume Next: Err = 0
Do While lst_ekernel_outq.ListCount > 0
lcl_o = lst_ekernel_outq.List(0)
ip_ekernel.SendData lcl_o + Chr$(13) + Chr$(10)
If Err Then
lab_msg =  Error  & Err &  -  & Err.Description
log E, ERR, Error  & Err &  -  & Err.Description &  during SendData  & lcl_
o &  to eKERNEL
Exit Do
Else
lst_ekernel_outq.RemoveItem 0
log O, TCP, lcl_o
End If
Loop
On Error GoTo 0
'
------------------------------------------------------------------------------
' Close socket after send
'
------------------------------------------------------------------------------
DoEvents
ip_ekernel.Close
' Set socket state indicator to defaults
lab_ekernel_state.BackColor = RGB(0, 0, 0)
' Update screen
g_ekernel_local_address = ip_ekernel.LocalIP
lab_ekernel_local_address =   & g_ekernel_local_address
g_ekernel_local_port = ip_ekernel.LocalPort
lab_ekernel_local_port =   & g_ekernel_local_port
'------------------------------------------------------------------------------
End Sub
Private Sub process_ekernel_jobq(cmd As String)
Dim lcl_rc As Integer
' <xxxxxx>
If Left$(cmd + Space$(13), 13) = <xml><xxxxxx> Then
' TODO - you could add code here
End If
' <yyyyyy>
If Left$(cmd + Space$(13), 13) = <xml><yyyyyy> Then
' TODO : you could add code here
End If
End Sub
Sub show_pages()
lab_log = Format$(lst_log.ListIndex + 1, 00) & / & Format$(lst_log.ListCount, 00)
End Sub
Private Sub tim_Timer()
Dim lcl_guarding As Variant
' Disable timer to prevent recursive calls
tim.Enabled = False
' Update clock
lab_clock =   & Format$(Now, hh:nn:ss)
' Update guarding
```

```
lcl_guarding = Timer - g_guarding
If lcl_guarding < 0 Then lcl_guarding = lcl_guarding + 86400
If (lab_guarding <> Format$(lcl_guarding, 00000)) Then
lab_guarding = Format$(lcl_guarding, 00000)
End If
' Process ekernel
process_ekernel
' Enable timer to resume processing
tim.Enabled = True
End Sub
Private Sub txt_log_GotFocus()
lst_log.SetFocus
End Sub
Sub log(log_type As String, log_sts As String, log_dta As String)
Dim lcl_rc As Integer
' Check log_type
Select Case log_type
eAPI_form - 7
Case I
Case O
Case S
Case E
Case Else
lcl_rc = MsgBox(Invalid log type  & log_type)
Exit Sub
End Select
' Check log_sts
Select Case log_sts
Case TCP
Case COM
Case INF
Case ERR
Case Else
lcl_rc = MsgBox(Invalid log status  & log_sts)
Exit Sub
End Select
' Add log data to listbox
lst_log.AddItem log_type & : & log_sts & : & log_dta
Do While lst_log.ListCount > 99
lst_log.RemoveItem 0
Loop
lst_log.ListIndex = lst_log.ListCount - 1
'--------------------------------------------------------------------------------
' Add log data to logfile
'
--------------------------------------------------------------------------------
' do not log is g_log_days=0
If g_ekernel_remote_port =  Then Exit Sub
' build directory and file
Dim lcl_path As String
Dim lcl_file As String
' start error recovery
On Error Resume Next: Err = 0
' if specified drive is valid, try to toggle between C: drive and D: drive
Err = 0
Dim lcl_chk As Integer
lcl_chk = Len(Dir$(g_log_path, vbDirectory))
lcl_path = g_log_path
If Len(Dir$(lcl_path, vbDirectory)) = 0 Then
MkDir lcl_path
End If
lcl_chk = Len(Dir$(g_log_path, vbDirectory))
If ((Err = 52) Or (lcl_chk = 0)) Then
```

```
Select Case Left$(g_log_path, 3)
Case C:\
g_log_path = D:\ + Mid$(g_log_path, 4)
lab_log_path =   & g_log_path
Case D:\
g_log_path = C:\ + Mid$(g_log_path, 4)
lab_log_path =   & g_log_path
Case Else
g_log_path = C:\ + Mid$(g_log_path, 4)
lab_log_path =   & g_log_path
End Select
End If
Err = 0
' make D:\SOPHO Messenger@Net
lcl_path = g_log_path
If Len(Dir$(lcl_path, vbDirectory)) = 0 Then
MkDir lcl_path
End If
' make D:\SOPHO Messenger@Net\log
lcl_path = lcl_path + \log
If Len(Dir$(lcl_path, vbDirectory)) = 0 Then
MkDir lcl_path
End If
' make D:\SOPHO messenger@Net\log\02001_eAPI
lcl_path = lcl_path + \ + Format$(g_ekernel_remote_port, 00000) + _eAPI
If Len(Dir$(lcl_path, vbDirectory)) = 0 Then
MkDir lcl_path
End If
' Kill log-files older then x days if g_LastLogFile not today
If Mid$(g_LastLogFile, 1, 8) <> Format$(Now, yyyymmdd) Then
eAPI_form - 8
KILL_OLD_LOGFILES lcl_path
End If
' make 20001030.txt
lcl_file = Format$(Now, yyyymmdd) & .txt
g_LastLogFile = lcl_file
' open file D:\SOPHO messenger@Net\log\02001_eAPI\20001030txt
Open lcl_path & \ & lcl_file For Append As 1
' write log record
Print #1, Format$(Now, dd/mm/yyyy hh:mm:ss) &  -  & log_type & : & log_sts & : &
log_dta
' close log file
Close 1
' disable error recovery
On Error GoTo 0
End Sub
```

Chapter 12: Module - eASYNC

The module eASYNC consists of one program eASYNC.exe, written in Visual Basic.

Overview

eASYNC.exe

The eASYNC.exe is the Visual Basic component of the eASYNC module. The program communicates with two processes: the eKERNEL.exe and the asynchronous modem attached to a COM port. The eKERNEL.exe is the central engine that centralizes all database access and communication with input and output capable modules.

The eASYNC.exe communicates with eKERNEL.exe by means of TCP sockets. In this communication, eASYNC.exe is a TCP client software that connects to the other component, acting as TCP server software.

At startup, eASYNC.exe contacts the eKERNEL.exe by means of a socket connection. Startup parameters are required to identify eASYNC.exe, and locate the eKERNEL.exe program. These parameters are set in the Properties section of the shortcut that initiates eASYNC.exe. This shortcut is usually located in the Windows Startup group (click **Start**, and choose **Programs > Startup**).

```
"C:\SOPHO Messenger@Net\Exe\eASYNC.exe"
/Site:1
/eKernel port:3105
/eKernel address:*LOCAL
/Log drive:C
```

Figure 37: Typical parameters in the shortcut

In the example in Figure 37: Typical parameters in the shortcut on page 81, the eASYNC.exe identifies itself as belonging to Site 1, and specifies the location of eKERNEL through IP address *LOCAL and port 3105. The special value *LOCAL refers to the assigned IP address of the first NIC adapter found in the PC. You can find this use the IPCONFIG.exe command or in the appropriate sections of the Windows network settings. The keyword Log drive refers to the drive in which the logging data must be stored; usually this is the C:-drive, referring to C:\SOPHO Messenger@Net\Log\ structure.

At startup, the eASYNC.exe sends an XML string to eKERNEL.exe requesting a configuration. This step is needed for each module that interacts with eKERNEL.exe, because this approach allows central administration using a single database, even if some client modules are located on a distributed machine.

```
<xml>
<cfgrqs>
<appl>eASYNC</appl>
<site>1</site>
</cfgrqs>
</xml>

<xml>
<cfgrpy><interface_cnt>2</interface_cnt>
<com_port_01>COM02</com_port_01>
<settings_01>9600,N,8,1</settings_01>
<type_01>PAGING</type_01>
<provider_01>BELGACOM</provider_01>
<password>*NONE</password>
<number_01>00452500001</number_01>
<init_01>AT&C0S0=3</init_01>
<rty_intv_01>60</rty_intv_01>
<rty_cnt_01>1</rty_cnt_01>
<snd_depth_01>1</snd_depth_01>
<snd_time_01>600</snd_time_01>
<com_port_02>COM02</com_port_02>
<settings_02>9600,N,8,1</settings_02>
<type_02>SMS</type_02>
<provider_02>PROXIMUS</provider_02>
<password>proximus</password>
<number_02>00475161622</number_02>
<init_02>AT&C0S0=3</init_02>
<rty_intv_02>60</rty_intv_02>
<rty_cnt_02>2</rty_cnt_02>
<snd_depth_02>1</snd_depth_02>
<snd_time_02>600</snd_time_02>
<log_path>C:\SOPHO Messenger@net</log_path>
<log_days>1</log_days>
</cfgrpy>
</xml>
```

Figure 38: A typical cfgrqs configuration request and reply

Refer to the appropriate sections on the database tables that define the received parameters for more information on each value. The information in this document is provided for informational purposes; detailed description of these internal inter-process communications is beyond the scope of this document.

If the <cfgrpy> shown in Figure 38: A typical cfgrqs configuration request and reply on page 82 is received, a license for eASYNC is valid.

If the <cfgrpy> shown in Figure 39: eASYNC module receives this cfgrpy from the eKernel if no license is available for eASYNC on page 83 is received, no license is available, and the eASYNC module cannot connect to the eKernel module anymore.

```
<xml>
<cfgrpy>
<licence>No licence available></licence>
</cfgrpy>
</xml>
```

Figure 39: eASYNC module receives this cfgrpy from the eKernel if no license is available for eASYNC

The eASYNC Connections tab is shown in Figure 40: eASYNC Connections tab on page 83.

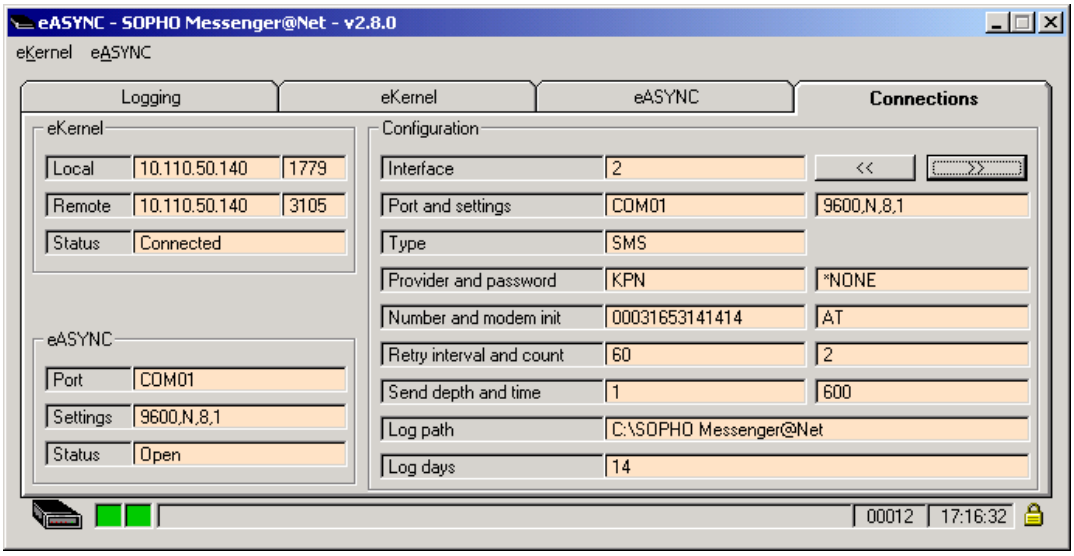


Figure 40: eASYNC Connections tab

The eASYNC module receives message requests from eKERNEL. After processing, feedback is sent from eASYNC to eKERNEL. Figure 41: Sample eKERNEL message request and eASYNC feedback on page 84 shows an example of a message request and the feedback generated by eASYNC.

```
<xml>
<msgrqs><id>00005</id>
<type>SMS</type>
<provider>PROXIMUS</provider>
<password>proximus</password>
<to>32475353215</to>
<pag_01>test message</pag_01>
<pag_more>N</pag_more>
</msgrqs>
</xml>


<xml>
<msgrpy>
<id>00005</id>
<sts>NACK - No carrier while waiting for connection^</sts>
</msgrpy>
</xml>
```

Figure 41: Sample eKERNEL message request and eASYNC feedback

During communications, eASYNC contacts the provider and handle the dialog that is required to deliver the message. The transactions are processed on a first-in first-out basis. However, configuration settings can be active that request a wait time or a queue depth that must be reached prior to initiating the communication process. This is especially relevant for SMS messaging to PROXIMUS or KPN, because these providers support the ability to deliver more than one SMS message during one single dial-up connection.

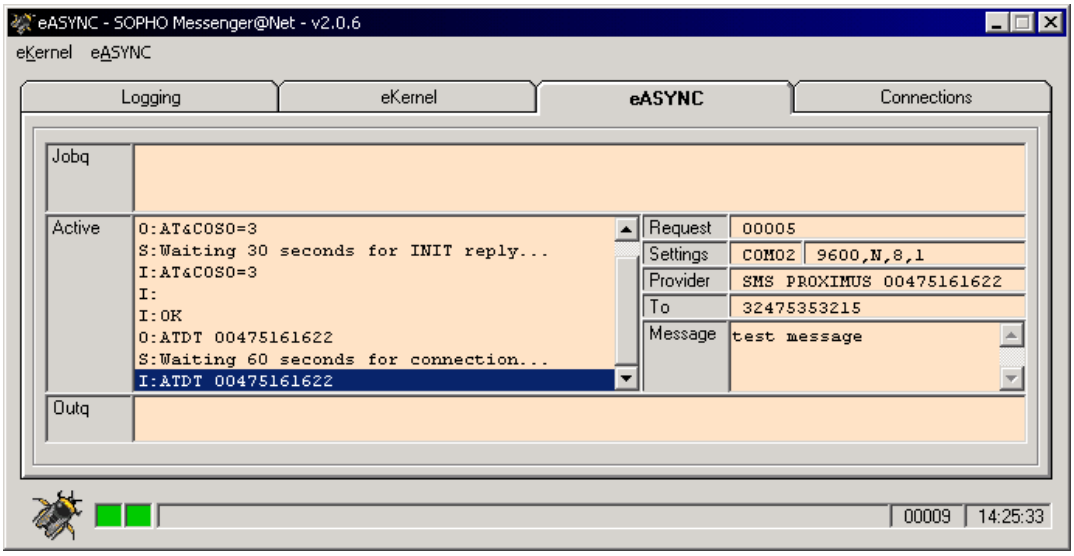


Figure 42: eASYNC tab

Logging

Logging information is available both on-screen and in logging files.

You can view on-screen logging through the Logging tab.

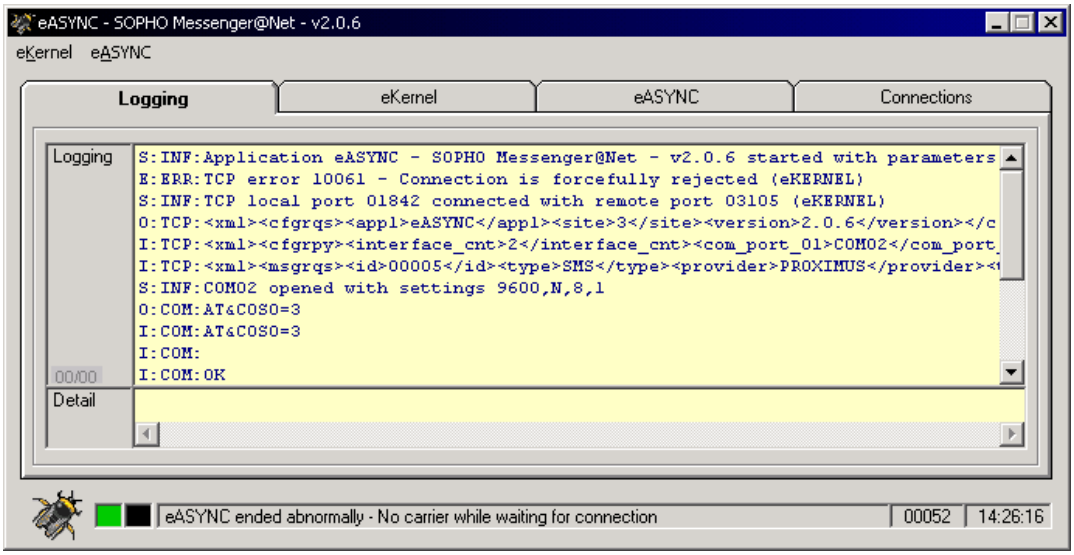


Figure 43: eASYNC Logging tab

```
19/03/2001 10:57:25 - S:INF:COM02 opened with settings 9600,N,8,1
19/03/2001 10:57:26 - O:COM:AT&C0S0=3
19/03/2001 10:57:26 - O:COM:ATDT 00475161622
19/03/2001 10:57:27 - I:COM:AT&C0S0=3
19/03/2001 10:57:27 - I:COM:
19/03/2001 10:57:27 - I:COM:OK
19/03/2001 10:57:27 - I:COM:ATDT 00475161622
19/03/2001 10:57:49 - I:COM:
19/03/2001 10:57:49 - I:COM:CONNECT 33600 V42bis
19/03/2001 10:57:50 - O:COM:_01/00121/O/01/32475353215//proximus/3/
534D5320746F2050726F78696D75732077697468205534F50484F204D657373656E67657
2404E6574/A3_
19/03/2001 10:57:54 - I:COM:connected
19/03/2001 10:57:54 - I:COM:_01/00019/R/01/A//69_
19/03/2001 10:57:55 - S:INF:Port closed
19/03/2001 10:57:55 - O:TCP:<xml><msgrpy><id>00002</id><sts>ACK^</
sts></msgrpy></xml>
```

Figure 44: Sample logging data for SMS to PROXIMUS

```
S:INF:COM03 opened with settings 9600,N,8,1
02-05-2002 13:57:37 - O:COM:AT
02-05-2002 13:57:38 - I:COM:AT
02-05-2002 13:57:38 - I:COM:
02-05-2002 13:57:38 - I:COM:OK
02-05-2002 13:57:39 - O:COM:ATDT 00653141414
02-05-2002 13:57:43 - I:COM:ATDT 00653141414
02-05-2002 13:58:16 - I:COM:
02-05-2002 13:58:16 - I:COM:CONNECT 33600 V42bis
02-05-2002 13:58:17 - O:COM:_01/00084/O/01/0620032328///3/
456D657267656E637920534F532031204576616375617469 6F6E/E2_
02-05-2002 13:58:33 - S:INF:Port closed
02-05-2002 13:58:33 - O:TCP:<xml><msgrpy><id>00142</id><sts>ACK^</
sts></msgrpy></xml>
```

Figure 45: Sample logging data for SMS to KPN

```
19/03/2001 15:56:08 - S:INF:COM02 opened with settings 9600,N,8,1
19/03/2001 15:56:09 - O:COM:AT&C0S0=3
19/03/2001 15:56:10 - I:COM:AT&C0S0=3
19/03/2001 15:56:10 - I:COM:
19/03/2001 15:56:10 - I:COM:OK
19/03/2001 15:56:09 - O:COM:ATDT 00452500001
19/03/2001 15:56:10 - I:COM:OK
19/03/2001 15:56:10 - I:COM:ATDT 00452500001
19/03/2001 15:56:34 - I:COM:
19/03/2001 15:56:34 - I:COM:CONNECT 14400 V42bis
19/03/2001 15:56:40 - I:COM:_
19/03/2001 15:56:40 - I:COM:WELCOME TO THE BELGACOM PAGING SERVICE.
19/03/2001 15:56:40 - I:COM:-------------------------------------
19/03/2001 15:56:40 - I:COM:You can call numbers in the range:
19/03/2001 15:56:40 - I:COM:2xxxxxx, 3xxxxxx, 8xxxxxx, 9xxxxxx
19/03/2001 15:56:40 - I:COM:Correction with backspace, delete or @
19/03/2001 15:56:40 - I:COM:Terminate each input with "RETURN-KEY"
19/03/2001 15:56:40 - I:COM:Disconnect with "ctrl-D"
19/03/2001 15:56:40 - I:COM:_
19/03/2001 15:56:40 - I:COM:****IMPORTANT INFORMATION****      +98+
19/03/2001 15:56:40 - I:COM:NEW  "Email Notification for paging"
19/03/2001 15:56:40 - I:COM:Interested: Send a mail to : Email.pag-
ing@belgacom.be
19/03/2001 15:56:41 - I:COM:or Contact 02/5406161(NL) - 02/5406302(FR)
19/03/2001 15:56:42 - I:COM:www.belgacom.be/cgi-bin/echannel/web/in-
dex.jsp?LANGUAGE=EN&DIVISION=RES
19/03/2001 15:56:42 - I:COM:Select:
19/03/2001 15:56:42 - I:COM:Catalog/Mobiles Solutions/Pagers   +99+
19/03/2001 15:56:42 - I:COM:_
19/03/2001 15:56:42 - I:COM:Type the 7 digits of the
19/03/2001 15:56:42 - I:COM:wanted pager-number:               +01+
19/03/2001 15:56:43 - O:COM:9789074
19/03/2001 15:56:44 - I:COM:.......
19/03/2001 15:56:44 - I:COM:9789074
19/03/2001 15:56:44 - I:COM:
19/03/2001 15:56:44 - I:COM:Type your alpha-numeric message.   +30+
19/03/2001 15:56:45 - O:COM:Test paging to Belgacom with SOPHO Messen-
ger@Net
```

continued on next page...

```
19/03/2001 15:56:46 -
I:COM:_[?7h...............................................................
..........................................................................
..............................
19/03/2001 15:56:46 - I:COM:_[A_[ATest paging to Belgacom with SOPHO Mes-
senger@Net
19/03/2001 15:56:47 - I:COM:
19/03/2001 15:56:47 - I:COM:CALL ACCEPTED.                     +80+
19/03/2001 15:56:48 - S:INF:Port closed
19/03/2001 15:56:48 - O:TCP:<xml><msgrpy><id>00001</id><sts>ACK^</
sts></msgrpy></xml>
```

Figure 46: Sample logging data for PAGING to BELGACOM

Chapter 13: Module - eBACKUP

You can use the eBACKUP module to make a backup of a predefined list of files.

The eBACKUP.exe must be started from a shortcut, which provides a number of command line parameters. Figure 47: eBACKUP shortcut with required line parameters on page 89 shows an example of a shortcut with the required command line parameters:

```
"C:\SOPHO Messenger@Net\Exe\eBACKUP.exe"
/Path:C:\SOPHO Messenger@Net
/Log drive:C
/Site:1
/Batch:N
```

Figure 47: eBACKUP shortcut with required line parameters

The following keywords are available:

- Path specifies the default path where the MDB subdirectory resides.
- Log drive specifies the letter of the drive in which logging information resides.
- Site specifies the site identifier to be saved.
- Batch specifies whether the application runs in interactively or in batch. In batch mode you do not need to click the Backup site Close button to close the program after execution. Batch is typically used in environments in which automated backup is scheduled at set intervals.

You can use the eBACKUP application to back up the files that are configured in the BACKUP table of the configuration database.

Table 4: eBACKUP sample data on page 89 shows sample data.

Table 4: eBACKUP sample data

Site	From path	From file	To path	To file
3	C:\Php	php.ini	C:\Temp\[weekday]\php	php.ini
3	C:\Program Files\Apache group\Apache\conf	httpd.conf	C:\Temp\[weekday]\Program Files\Apache Group\Apache \conf	httpd.conf
3	C:\SOPHO Messenger@Net \Exe	csta.dll	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	csta.dll
3	C:\SOPHO Messenger@Net \Exe	CSTA_Service.ex e	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	CSTA_Service .exe

Site	From path	From file	To path	To file
3	C:\SOPHO Messenger@Net\Exe	eAPI.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eAPI.exe
3	C:\SOPHO Messenger@Net\Exe	eASYNC.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eASYNC.exe
3	C:\SOPHO Messenger@Net\Exe	eBACKUP.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eBACKUP.exe
3	C:\SOPHO Messenger@Net\Exe	eCAP.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eCAP.exe
3	C:\SOPHO Messenger@Net\Exe	eDMSAPI.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eDMSAPI.exe
3	C:\SOPHO Messenger@Net\Exe	eGRID.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eGRID.exe
3	C:\SOPHO Messenger@Net\Exe	eIO.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eIO.exe
3	C:\SOPHO Messenger@Net\Exe	eKERNEL.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eKERNEL.exe
3	C:\SOPHO Messenger@Net\Exe	eSMTP.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eSMTP.exe
3	C:\SOPHO Messenger@Net\Exe	eSMTP_ server.exe	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	eSMTP_ server.exe
3	C:\SOPHO Messenger@Net\Exe	omnithread_rt.dll	C:\Temp\[weekday]\SOPH O Messenger@Net\Exe	omnithread_rt .dll
3	C:\SOPHO Messenger@Net\Mdb	Messenger_CFG .mdb	C:\Temp\[weekday]\SOPH O Messenger@Net\Mdb	Messenger_CF G.mdb
3	C:\SOPHO Messenger@Net\Mdb	Messenger_Data .mdb	C:\Temp\[weekday]\SOPH O Messenger@Net\Mdb	Messenger_Da ta.mdb

From_path and From_file specify the path and the name of the file that are copied to the To_path and To_file.

When eBACKUP is started, a blank window with one button is shown, for example, Backup site 3.

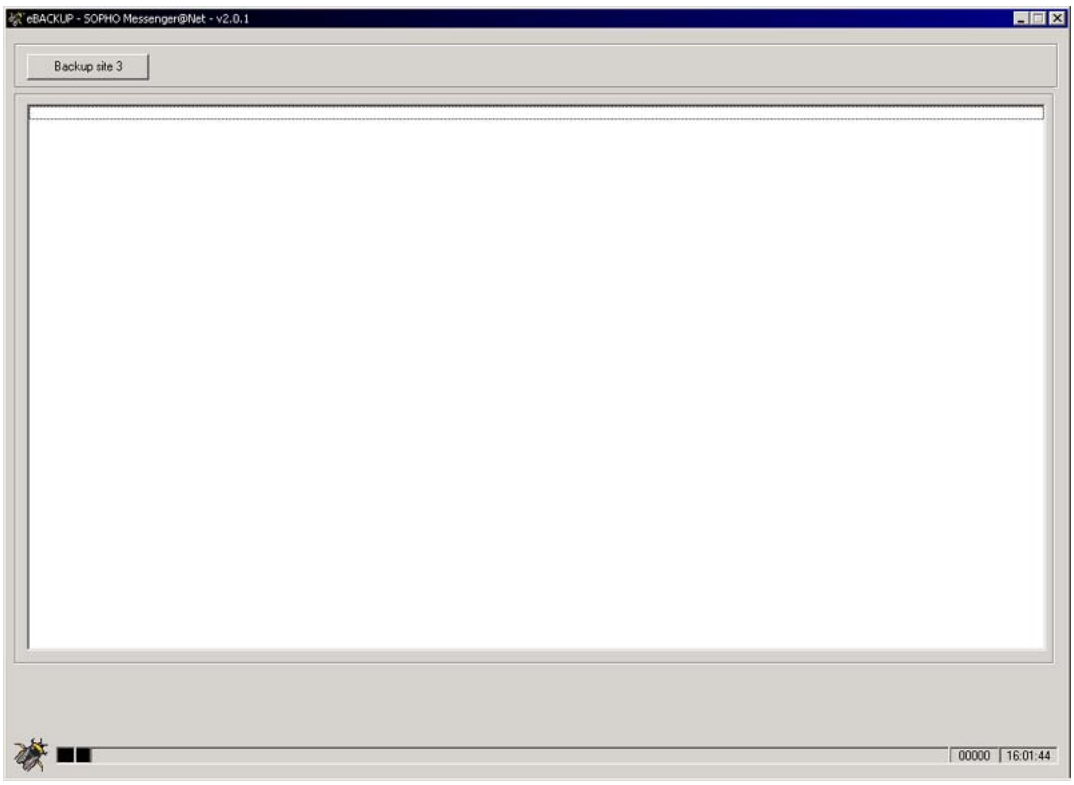


Figure 48: Backup start window

Click Backup to begin the backup procedure.

When all the files are successfully copied, the window becomes green.

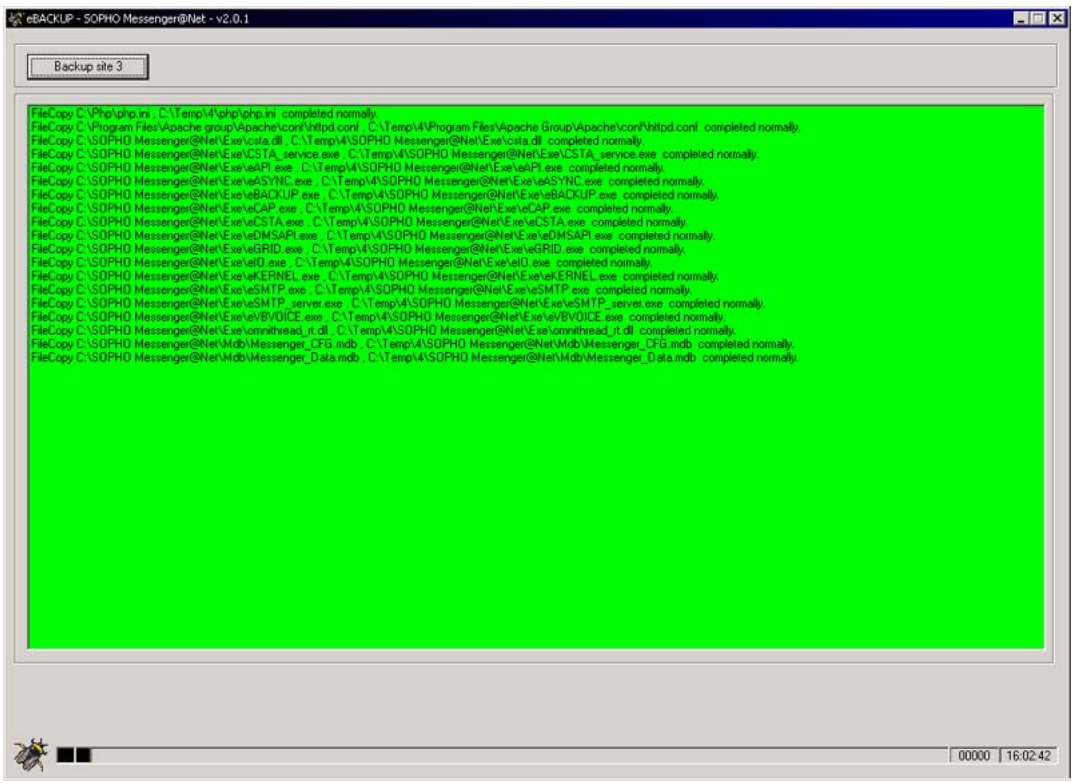


Figure 49: Backup successful

If one or more files are not copied, the window becomes red.

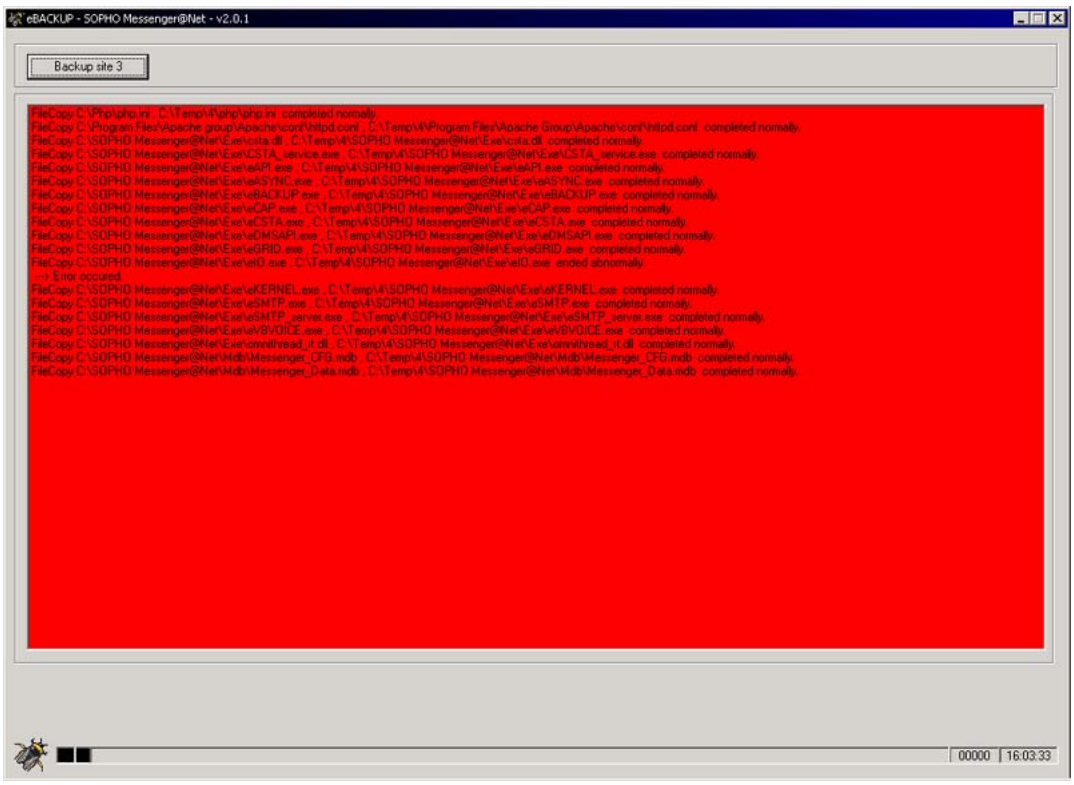


Figure 50: Backup error window

During backup, logging information is written to the hard disk, an example of which is shown in Figure 51: Sample backup log on page 94. Note that in the example, the file eIO.exe was not saved.

```
25/10/2001 16:06:20 - S:INF:Application eBACKUP - SOPHO Messenger@Net -
v2.0.1 started with parameters /Path:C:\SOPHO Messenger@Net /Log drive:C
/Site:3 /Batch:N
25/10/2001 16:06:22 - S:INF:FileCopy C:\Php\php.ini ,
C:\Temp\4\php\php.ini  completed normally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\Program Files\Apache
group\Apache\conf\httpd.conf , C:\Temp\4\Program Files\Apache
Group\Apache\conf\httpd.conf  completed normally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\csta.dll
, C:\Temp\4\SOPHO Messenger@Net\Exe\csta.dll  completed normally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Exe\CSTA_service.exe , C:\Temp\4\SOPHO Messen-
ger@Net\Exe\CSTA_service.exe  completed normally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eAPI.exe
, C:\Temp\4\SOPHO Messenger@Net\Exe\eAPI.exe  completed normally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Exe\eASYNC.exe , C:\Temp\4\SOPHO Messenger@Net\Exe\eASYNC.exe
completed normally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eBACK-
UP.exe , C:\Temp\4\SOPHO Messenger@Net\Exe\eBACKUP.exe  completed nor-
mally.
25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eCAP.exe
, C:\Temp\4\SOPHO Messenger@Net\Exe\eCAP.exe  completed normally.

25/10/2001 16:06:22 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eDMSA-
PI.exe , C:\Temp\4\SOPHO Messenger@Net\Exe\eDMSAPI.exe  completed nor-
mally.
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Exe\eGRID.exe , C:\Temp\4\SOPHO Messenger@Net\Exe\eGRID.exe
completed normally.
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eIO.exe
, C:\Temp\4\SOPHO Messenger@Net\Exe\eIO.exe  ended abnormally.
25/10/2001 16:06:23 - S:INF: ---> Error occured.
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eKER-
NEL.exe , C:\Temp\4\SOPHO Messenger@Net\Exe\eKERNEL.exe  completed nor-
mally.

continued on next page...
```

Figure 51: Sample backup log

```
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messenger@Net\Exe\eS-
MTP.exe , C:\Temp\4\SOPHO Messenger@Net\Exe\eSMTP.exe  completed normal-
ly.
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Exe\eSMTP_server.exe , C:\Temp\4\SOPHO Messen-
ger@Net\Exe\eSMTP_server.exe  completed normally.
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Exe\omnithread_rt.dll , C:\Temp\4\SOPHO Messen-
ger@Net\Exe\omnithread_rt.dll  completed normally.
25/10/2001 16:06:23 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Mdb\Messenger_CFG.mdb , C:\Temp\4\SOPHO Messen-
ger@Net\Mdb\Messenger_CFG.mdb  completed normally.
25/10/2001 16:06:25 - S:INF:FileCopy C:\SOPHO Messen-
ger@Net\Mdb\Messenger_Data.mdb , C:\Temp\4\SOPHO Messen-
ger@Net\Mdb\Messenger_Data.mdb  completed normally.
25/10/2001 16:06:30 - S:INF:Application ended
```

Avaya recommends that you close all the DECT Messenger applications before starting the backup procedure. In release 2, the file copy procedure is implemented by means of a Windows API-call, as shown with the code excerpt in Figure 52: File copy example on page 95:

```
:
Declare Function apiCopyFile Lib "kernel32" Alias "CopyFileA" (ByVal
lpExistingFileName As String, ByVal lpNewFileName As String, ByVal bFail-
IfExists As Long) As Long
:
Result = apiCopyFile(SourceFile, DestFile, False)
:
```

Figure 52: File copy example

Important:

To ensure a complete and consistent image, you must close all applications before backup. The code shown in Figure 52: File copy example on page 95 can back up files, even if they are open. Therefore, you can initiate the eBACKUP while, for instance, eKERNEL is active and the Messenger_CFG.mdb database is open. Although you can use the eBACKUP to save the open files, Avaya does not guarantee that the copied file is a complete image or a consistent database image. During activity of eKERNEL, parts of the Access 2000 database are sometimes in use and transactions are pending. Saving open files is not officially supported.

For more information, visit the Microsoft web site at: http://support.microsoft.com/support/kb/articles/Q207/7/03.asp

Chapter 14: Module - eCAP

The module eCAP consists of the program eCAP.exe, written in Visual Basic. In general, DECT Messenger programs reside in the default directory C:\SOPHO Messenger@Net\Exe, unless otherwise implemented in your environment.

Overview

eCAP.exe

The eCAP.exe is a Visual Basic component of the eCAP module. The program communicates with two processes: the eKERNEL.exe and an external alarm interface. The eKERNEL.exe is the central engine that centralizes all database access and communication with input and output capable modules.

The eCAP.exe communicates with eKERNEL.exe by means of TCP sockets. In this communication, eCAP.exe is a TCP client software that connects to the eKERNEL component, acting as TCP server software.

At startup, eCAP.exe contacts the eKERNEL.exe by means of a socket connection. Startup parameters identify eCAP.exe, and locate the eKERNEL.exe program. These parameters are set in the Properties section of the shortcut that initiates eCAP.exe. This shortcut is usually located in the Windows Startup group (click **Start** on the Windows toolbar, and choose **Programs > Startup**).

```
"C:\SOPHO Messenger@Net\Exe\eCAP.exe"
/Site:1
/eKernel address:*LOCAL
/eKernel port:3102
/Log drive:C
```

Figure 53: Typical parameters in the shortcut

In the example shown in Figure 53: Typical parameters in the shortcut on page 97, the eCAP.exe identifies itself as belonging to site 1, and specifies the location of eKERNEL through IP address *LOCAL and port 3102. The special value *LOCAL refers to the assigned IP address of the first NIC adapter found in the PC. You can determine the IP address using the IPCONFIG.exe command or in the appropriate sections of the Windows network settings. The

keyword Log drive refers to the drive where the logging data must be stored. Usually this is C:\SOPHO Messenger@Net\Log\.

At startup, the eCAP.exe sends an XML string to eKERNEL.exe requesting a configuration. This step is needed for each module that interacts with eKERNEL.exe, because this approach allows central administration using a single database, even if some client modules are located on a distributed machine.

```
<xml>
<cfgrqs>
<appl>eCAP</appl>
<site>1</site>
</cfgrqs>
</xml>

<xml>
<cfgrpy>
<manufacturer>ELDAD</manufacturer>
<model>L:48-0:RC-1:SR-2:SS-3:SS-4:SR </model>
<bidir>N</bidir>
<link_type>RS232</link_type>
<resource>COM01</resource>
<settings>9600,N,8,1</settings>
<descr>Eldad DP6000</descr>
<log_path>c:\SOPHO Messenger@net</log_path>
<log_days>30</log_days>
</cfgrpy></xml>
```

Figure 54: A typical cfgrqs configuration request and its received cfgrpy configuration reply

Note:

The generic eCAP configuration sends extra keywords and values, as defined in the eCAP_generic table.

Refer to the chapters of this document that describe the database tables for more information on each value. Detailed descriptions of these internal inter-process communications is beyond the scope of this chapter.

When the configuration is received, the **Connection** tab displays information similar to what is pictured in Figure 55: eCAP Connection tab on page 99.

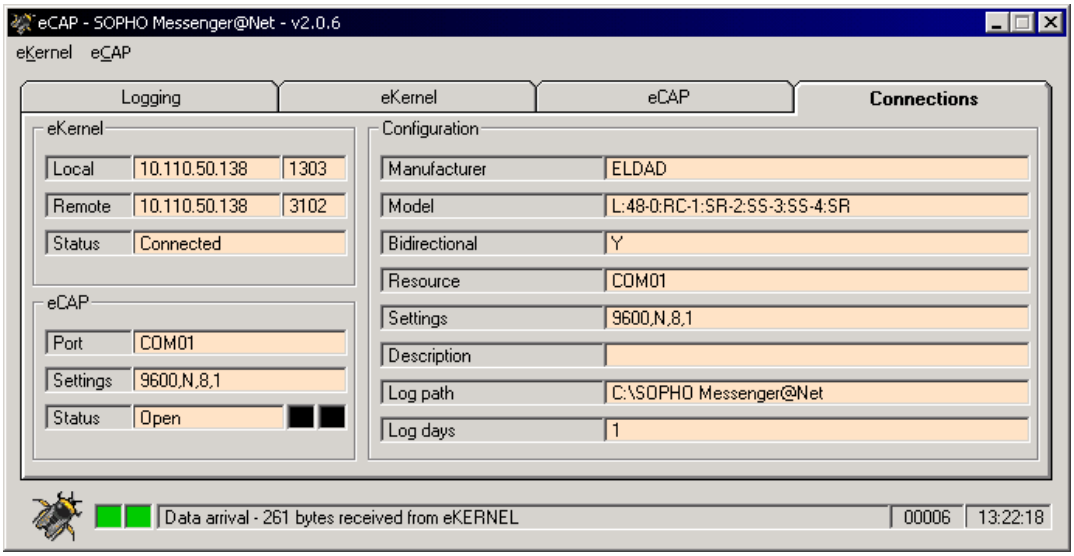


Figure 55: eCAP Connection tab

Because the eCAP is designed to handle asynchronous serial communications with a number of alarm systems, the eCAP requires configuration settings to start processing. These values are returned through the <cfgrpy> reply that is sent on return of the <cfgrqs> request. Some parameters refer to asynchronous communication settings (for example, port number, baud rate, data bits, parity bits, stop bits, and so on); others refer to general information settings (for example, logging parameters); the rest are parameters that actually determine the alarm system (for example, manufacturer, model, bidirectional, and so on).

Note:

The values shown in Figure 55: eCAP Connection tab on page 99 are received from the DECT Messenger database: from the eKERNEL_INPGM table, eCAP_generic table, and the eKERNEL_SITE table.

At startup, the eCAP.exe appears the specified COM port with the specified settings. The COM port specified must be available, be set to use a valid baud rate, and so on. A physical connection must exist between the specified COM port and the external alarm system through a properly wired serial cable. In many cases, alarm systems support a limited number of control signals (for example, ground and send), so consult the alarm system vendor on cable specifications. In most cases, you can use a standard null-modem cable. If no more COM ports are available, extra hardware (such as DigiBoard PC/4e or DigiBoard PC/8e) is needed to provide extra serial ports. Check compatibility issues (supported by operating system, driver available, and so on) and hardware requirements (memory, available slots, IRQ conflicts, and so on) before ordering or configuring a system.

In many cases the distance between the DECT Messenger and the external-alarm system is relatively small, so no extra hardware is needed. In some conditions hardware is needed, such as, when RS-232-C limitations apply (for example, at 9600 baud maximum limit of 9 metres). In some cases galvanic isolation is requested, or base-band modems, SOPHO LAM, CISCO equipment, and so on are needed to bridge the distance between the DECT Messenger and the alarm system.

Once a link is established between eCAP and the alarm system, the eCAP handles further communications and informs eKERNEL when relevant information is to be exchanged.

Functional description

In general, eCAP is designed to provide eKERNEL with alarm information. This is carried out using a <msgrqs> message request. For some interfaces eKERNEL must send feedback to the alarm system, a process that is handled through <msgrpy> message reply request.

Important:

The eCAP module is compatible with a number of alarm system installations. However, many of the supported vendors offer a broad variety of hardware and software environments, all of which are not necessarily compatible with eCAP. For example, the fact that one NIRA serial protocol is implemented does not mean that every version of serial input from NIRA is compatible. Ensure that a specific alarm-system model is supported by DECT Messenger before purchasing or installing it. In most cases manufacturers are not using the same standard for all of their equipment, so obtain information on protocols and specifications. Avaya recommends pre-sales consultation. If necessary, a modification of the current release of eCAP can be made to embed new protocols.

The most typical protocols are listed in Table 5: Supported manufacturer/model protocols on page 100 and described in more detail in the pages following the table. Refer to the protocol specifications of each vendor for more information, as detailed protocol issues are beyond the scope of this document. The information in Table 5: Supported manufacturer/model protocols on page 100 provides a list of supported manufacturers and models. This information is provided on an as-is basis, to illustrate the eCAP module.

Table 5: Supported manufacturer/model protocols

Manufacturer	Model
ARGINA	*BASE
ARITECH	*BASE
ARITECH	1
BEMAC	DIANA 1
BEMAC	DIANA 2
ELDAD	L:48-0:RC-1:SR-2:SS-3:SS-4:SR
GENERIC	*BASE
GENT	3400
GENT	VIGILON EN54
M-TECH	ESPRESSO
NIRA	*BASE

Manufacturer	Model
TELEVIC	PROTOCOL CONVERTOR – L:03
TYCO	MINERVA 80
VSK	DE LICHTERVELDE
VSK	OLV VAN VREDE
VSK	ST-JOZEF
WORMALD	*BASE
WORMALD	L:01
WORMALD	G:EIPM

ARGINA

The valid manufacturer is ARITECH and the valid model is *BASE.

Argina *BASE is based upon installation Maas en Kempen-Campus Bree.

Table 6: Sample ARGINA protocol data

#BAL, Z002,D003,00000,00000,00000,Inkomhal

Argina alarms are always sent to group ARGINA, since no pager information is available in the data-stream.

Messages are sent with an alarm description based upon the first part before the comma. The # character is omitted. This means #BAL results in an alarm description BAL. The message is based upon the sixth element, for example, Inkomhal, omitting the leading and trailing spaces. Alarms are sent with the option: remove after *sent.

ARITECH

The valid manufacturer is ARITECH and the valid model is *BASE and 1.

Aritech *BASE is based upon the installation Floreal Nieuwpoort.

```
19980218 08:49:41 ----------------------------------------
19980218 08:49:41 Fout     :1      Gebeu. :1258 Aktief
19980218 08:49:41 Poort    :SER1   Printer afgekoppeld
19980218 08:49:41                  18/02/98  08:19:31 P:1
19980218 08:49:41 ----------------------------------------
19980218 08:49:41 Actie    :4      Gebeu. :1259 Gelogd
19980218 08:49:41                  Stop Zoemer
19980218 08:49:41                  18/02/98  08:19:35 P:1
19980218 08:49:42 ----------------------------------------
19980218 08:49:42 Fout     :2      Gebeu. :1260 Gelogd
19980218 08:49:42                  Deur contact
19980218 08:49:43                  18/02/98  08:19:47 P:1
19980218 08:49:43 ----------------------------------------
19980218 08:49:43 Actie    :5      Gebeu. :1261 Gelogd
19980218 08:49:43                  Stop Zoemer
19980218 08:49:43                  18/02/98  08:19:48 P:1
19980218 08:49:43 ----------------------------------------
19980218 08:49:43 Actie    :6      Gebeu. :1262 Gelogd
19980218 08:49:43                  Deurcontact gesloten
19980218 08:49:43                  18/02/98  08:21:09 P:1
19980218 08:49:43 ----------------------------------------
19980218 08:49:43 Fout     :2      Gebeu. :1263 Gelogd
19980218 08:49:43                  Deur contact
19980218 08:49:43                  18/02/98  08:21:11 P:1
19980218 08:49:43 ----------------------------------------
19980218 08:49:43 Actie    :7      Gebeu. :1264 Gelogd
19980218 08:49:43                  HERSTEL
19980218 08:49:43                  18/02/98  08:22:35 P:1
19980218 14:28:29 ----------------------------------------
19980218 14:28:29 Alarm    :1      Gebeu. :1509 Aktief
19980218 14:28:30 Zone     :2      Gebied :0
19980218 14:28:30 Adres    :3/4    Brand
19980218 14:28:30 DKKV             18/02/98  14:24:05 P:1
19980218 14:28:30 C TELEFOONCENTRALE
19980218 14:28:46 ----------------------------------------
19980218 14:28:46 Actie    :3      Gebeu. :1510 Gelogd
19980218 14:28:46                  Stop Zoemer
19980218 14:28:46                  18/02/98  14:24:23 P:1
```

Figure 56: Sample Aritech protocol data

Aritech alarms are always sent to group ARITECH, because no pager information is available in the datastream. Messages are sent with alarm description ARITECH. An alarm is set only when Gebeur occurs in the datastream. When BRAND occurs the message is BRAND; in other cases the message is ARITECH. When HERSTEL occurs a general reset of all ARITECH alarms is issued.

Note:

Use of this protocol usually requires consulting services and customizing.

```
' ================================================================
' SCENARIO 1 : Brand
' actie *SET
' message "T112 det"
' groep "ARITECH_B"
' alarm description "ARITECH_B"
' ----------------------------------------------------------------
' Alarm :1 Gebeu. :2109 Aktief
' Zone :42 Geb. :0
' Adres :3/1 Brand
' OPT 04/07/05 10:19:03 P:6
' T122 det
' ================================================================
' ================================================================
' SCENARIO 2 : Fout
' actie *SET
' message "Printer afgekoppeld G:1"
' groep "ARITECH_F"
' alarm description "ARITECH_F"
' ----------------------------------------------------------------
' Fout :1 Gebeu. :661 Aktief
' Poort :SER2 Printer afgekoppeld
' 04/07/05 10:08:39 G:1
' ================================================================
' ================================================================
' SCENARIO 3 : Fout - communicatie fout
' actie *SET
' message "T122 det P6"
' groep "ARITECH_C"
' alarm description "ARITECH_C"
' ----------------------------------------------------------------
' Fout :1 Gebeu. :2106 Aktief
' Zone :42 Geb. :0
' Adres :3/1 Communicatie fout
' OPT 04/07/05 09:59:54 P:6
' T122 det
' ================================================================
' ================================================================
' SCENARIO 4 : Herstel
' "*RESET" naar alle input
' ----------------------------------------------------------------
' Actie :1 Gebeu
' Herstel
' 04/07/05 10:30:19 P:8
' ================================================================
```

Figure 57: ARITECH Model 1 alarm (based upon installation RUCA)

Aritech model 1 alarms are always sent to group ARITECH_F, ARITECH_C, or ARITECH_B, depending on the datastream.

Messages are sent with alarm description ARITECH_F, ARITECH_C, or ARITECH_B, depending on the datastream.

Alarms are *SET with option *CALC.

To define alarms, remove after *RESET.

When HERSTEL occurs a general reset of all ARITECH alarms is issued for *ALL groups and *ALL alarm descriptions.

Note:

This protocol usually requires project-based consulting services and customizing.

BEMAC

Valid manufacturer is BEMAC, valid model is DIANA 1 and DIANA 2.

Bemac is based upon installation Clinique St-Vincent Rocourt.

```
20000913 15:07:09 I:<866/LOC 101B/0>
20000913 15:09:09 I:<861/LOC 222B/0>
20000913 15:09:31 I:<861/LOC 333B/0>
20000913 15:10:47 I:<861/LOC 444B/0>
20000913 15:11:34 I:<999/RESET/0>
20000913 15:12:04 I:<999/RESET/0>
20000913 15:12:39 I:<999/RESET/0>
```

Figure 58: Sample Bemac protocol data

Bemac alarms contain three fields, a pager number, a message and a tone code. Alarms are sent with group equal to the first parameter (for example, 866). The message is retrieved from the second parameter (for example, LOC 101B). When the third parameter is 0 the message is reset. When the message is RESET all messages for all groups are reset; in other cases only the specific message for the specified group is reset. When the third parameter is a value other than 0, the message is set. During set the alarm description is BEMAC_x, where x is the specified tone code (for example, tone code 3 sets message with alarm description BEMAC_3.

ELDAD

Valid manufacturer is ELDAD, valid model is specified through a special syntax, for example, L:48-0:RC-1:SR-2:SS-3:SS-4SR. Note that the model is built upon components having syntax A:BB and separated with a hyphen (-).

- L:xx denotes that the length of an alpha-message is xx bytes. For example, L:48 means that alpha-messages are 48 bytes long, L:24 denotes alpha-messages are 24 bytes long.
- 0:xx specifies behavior of tone code 0, 1:xx specified behavior of tone code 1, 2:xx specifies behavior of tone code 2, and so on.

For each tone code, the syntax ends on two characters. The first character can be S or R. S denotes set of alarm, R denotes reset of alarm. The second character can be S or R or C. The value S refers to remove after *SENT, the value R refers to remove after *RESET, the C refers to remove after *CALC.

For example, L:48-0:RC-1:SR-2:SS-3:SS-4SR means the alpha-messages are 48 bytes long, tone code 0 denotes *RESET alarm remove after *CALC, tone code 1 and 4 denote *SET alarm remove after *RESET, and tone code 2 and 3 denote *SET alarm remove after *SENT.

Bemac is based upon installation Sint-Franciskus-Ziekenhuis Heusden-Zolder.

The syntax is

```
'STX' + XXXXTZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZYYYYYP + 'LF' + 'CR'
```

- XXXX = pager number
- T = Tone-code
- ZZ.. = Alpha-message (usually 24 or 48 bytes)
- YYYYY = 5-digit information (= message if ZZ.. are all ...)
- P = present flag

```
20000419 00:00:45 I:_01241NUR A203                00000P
20000419 00:00:45 O: _(ACK)
20000419 00:01:35 I:_01640NUR A203                00000P
20000419 00:01:35 O: _(ACK)
20000419 00:03:38 I:_01244ASS A203                00000P
20000419 00:03:38 O: _(ACK)
20000419 00:03:50 I:_01244ASS A203                00000P
20000419 00:03:50 O: _(ACK)
20000419 00:07:09 I:_01142TECHNISCH               00000P
20000419 00:07:09 O: _(ACK)
20000419 00:11:41 I:_01643BRAND                   00000P
20000419 00:11:41 O: _(ACK)
```

Figure 59: Sample Eldad protocol data

Eldad messages are either *SET or *RESET based upon tone code and model configuration. The group is located from the datastream (4-byte pager number). The message is retrieved from the alpha-message, but if the alpha-message is a string of period (.) characters then 5-byte digit information is used instead. The alarm description is the tone code (for example, 1 or 2 or 3 and so on).

GENT

The valid manufacturer is GENT and the valid model is 3400 and VIGILON EN54.

Model 3400

The GENT model 3400 transmits binary datastreams of 56 bytes. The first bytes identify the alarm type. Currently, only Fire and Super Fire are processed.

```
' ---------------------------------------------------------
' Byte 0 and 1 - Event code
' ---------------------------------------------------------
' MSB LSB Event description
' 0 1 Fire reset
' 0 2 All faults cleared
' 0 3 All disablements cleared
' 0 4 Alarms silenced
' 0 5 Alarms sounded
' 2 1 Supervisory on
' 2 2 Supervisory off
' 4 X Fault - System
' 5 X Fault - Outstanding/Loop
' 7 X Disablement
' 9 X Fire
' 10 X Super Fire
' ---------------------------------------------------------
' Byte 25-56 - Outstanding/Zole/Supervisory/Panel label
' ---------------------------------------------------------
```

Figure 60: GENT Model 3400

The message is retrieved from bytes 25 to 56. The group is always FIRE and the alarm description is always FIRE. Alarms are *SET with the option: remove after *sent.

Model VIGILIN EN54

```
Time:14:38.35 Thu 29 June 2006
Call Point Operated
MCP;FAP 1 BLOCK 2 LOWER GROUND FLOOR Z
Number 149 on Loop 1
Sector 1 Group 1 Zone 1
---------------------------------------
Time:14:38.35 Thu 29 June 2006
Call Point Operated
MCP;FAP 1 BLOCK 2 LOWER GROUND FLOOR Z
Number 149 on Loop 1
Sector 1 Group 1 Zone 1
---------------------------------------
Time:14:38.35 Thu 29 June 2006
Call Point Operated
MCP;FAP 1 BLOCK 2 LOWER GROUND FLOOR Z
Number 149 on Loop 1
Sector 1 Group 1 Zone 1
---------------------------------------
Time:14:38.35 Thu 29 June 2006
Call Point Operated
FIRE;FAP 1 BLOCK 2 LOWER GROUND FLOOR Z
Number 149 on Loop 1
Sector 1 Group 1 Zone 1
---------------------------------------
```

Figure 61: Sample Model VIGILIN EN54

Currently, only records starting in 3rd line MCP or FIRE are processed.

For FIRE alarms, the group is FIRE and the alarm description is FIRE. Messages are sent with the option: remove after *sent.

For MCP alarms, the group is MCP and the alarm description is MCP. Messages are sent with the option: remove after *sent.

The message is retrieved from the remaining part of the line after FIRE or MCP. For example, FIRE;FAP 1 BLOCK 2 LOWER GROUND FLOOR Z returns the message FAP 1 BLOCK 2 LOWER GROUND FLOOR Z.

M-TECH

The valid manufacturer is M-TECH and the valid model is ESPRESSO.

Table 7: Sample M-TECH data

<00001/This is a sample message>

The first field starts between < and /, in the example 00001.

The second field starts between / and > in the sample M-TECH data message.

The group is identified by the first fields, for example, 00001, and the message is identified by the second field (This is a sample message). Currently, alarms are sent with alarm description P0 and sent with option Remove after *sent.

NIRA

The valid manufacturer is NIRA and the valid model is *BASE.

Nira is based upon installation Eeuwfeestkliniek Antwerpen.

```
20010110 00:00:02 I: 22:37 | 0781 | 10 | | 6 | | - | 416-C
20010110 00:00:12 I: 22:37 | 0784 | 10 | | 6 | | - | 419-C
20010110 00:00:12 I: Tuesday
9-JANUARY-2001
20010110 00:00:12 I:
20010110 00:00:12 I: TIME ADDR. LINK S/P CODE CALL-INFO ID MESSAGE
20010110 00:00:12 I:-------+------+----+---+----+---------+-----+-------
------
----------------------
20010110 00:00:12 I: 22:37 | 0777 | 08 | | 5 | | - | 336-C----A
20010110 00:00:33 I: 22:38 | 0777 | 08 | | 5 | | - | 336-C----A
20010110 00:00:33 I: 22:38 | 0781 | 10 | | 6 | | - | 416-C
20010110 02:02:17 I: 00:39 | 0777 | 08 | | 5 | | - | 336-C----A
20010110 02:02:18 I: 00:39 | 0784 | 10 | | 7 | | - | 520-A
20010110 02:02:38 I: 00:40 | 0777 | 08 | | 5 | | - | 336-C----A
20010110 02:02:59 I: 00:40 | 0777 | 08 | | 5 | | - | 336-C----A
```

Figure 62: Sample NIRA protocol data

Alarms are always *SET, and repeat mechanisms are configured to indicate if an alarm is no longer active (logical *RESET is based upon no longer repeating). For this reason, Avaya recommends that the alarm repeat interval be specified, for example, to 30 seconds.

The room number is retrieved from the data-stream and is used as a group, for example, 416, 419, 336, and so on. The message is 4-bytes long and consists of the room number, followed by either a C or an A indication. Datastreams in the format XXX-C and XXX----A are considered as C and datastreams in the format XXX-A and XXX---AA and XXX—AAA are considered A type alarms.

Message text is 416C or 520A. The alarm description is NIRA_C for messages that end with C and NIRA_A for alarms that end with A. All messages are removed after *SENT.

Note:

This protocol typically requires project-based consulting services and modifications.

STEAFA

Note:

The Landis-Steafa interface was not ordered nor implemented and protocol information is provided on an as-is basis. Contact your software vendor for implementation.

```
19970403000053 COM    03/04/97  Prio 1 CONTROLLER 00   SYSTEM STAEFA
test alarm
19970403000054 COM    00:05:00      COM. NICO-RS MODU(U)L(EN) NORMAAL
UIT
19970403000054 COM                                                  00
COS   2000
19970403000154 COM ____
19970403000154 COM    03/04/97  Prio 1 CONTROLLER 00   SYSTEM STAEFA
test alarm
19970403000154 COM      00:06:01          GEEN COMMUNICATIE NICO-RS
MODU(U)L(EN)   IN
19970403000154 COM                                                  00
COS   2000
19970403000252 COM ____
19970403000253 COM    03/04/97  Prio 1 CONTROLLER 00   SYSTEM STAEFA
test alarm
19970403000253 COM    00:07:00      COM. NICO-RS MODU(U)L(EN) NORMAAL
UIT
19970403000253 COM                                                  00
COS   2000
19970403000354 COM ____
```

Figure 63: Sample Steafa protocol data

TELEVIC

The valid manufacturer is TELEVIC and the valid model is PROTOCOL CONVERTOR–L:xx, with xx between 01 and 99.

Note:

The extension – L:xx is new in release 2, and must be specified. The functionality is introduced to obtain more flexibility in message handling. The following string handling is performed:

- Remove leading spaces of the message

  NUR K100 -> NUR K100

- Append a trailing space to the result

BEERPUT -> BEERPUT

- Look up the occurrence of the first space character

  NUR K100 -> 4

  BEERPUT -> 8

- Keep the leading non-blank characters

  NUR K100 and 4 -> NUR

  BEERPUT and 8 -> BEERPUT

- Keep the leading characters only, with length specified in L:xx

  NUR and L:03 -> NUR

  BEERPUT and L:03 -> BEE

Table 8: Televic Example

Length	Original message	Resulting alarm type	Length
L:01	NUR K100	N	1
	WC 120	W	1
	BEERPUT	B	1
L:02	NUR K100	NU	2
	WC 120	WC	2
	BEERPUT	BE	2
L:03	NUR K100	NUR	3
	WC 120	WC	2
	BEERPUT	BEE	3
L:04	NUR K100	NUR	3
	WC 120	WC	2
	BEERPUT	BEER	4
L:05	NUR K100	NUR	3
	WC 120	WC	2
	BEERPUT	BEERP	5

Refer to the official specifications on Televic Protocol Convertor. You can obtain these specifications from the manufacturer, or through Avaya sales support. Detailed information is beyond the scope of this document.

Televic is based upon installation CAZK campus Groeninghe Kortrijk.

```
07/01/2001 00:02:15 - I:COM:_O0103224210          8POORT100Y4C_
07/01/2001 00:02:15 - O:COM:_A0103Y1A_
07/01/2001 00:02:22 - O:COM:_T010312421052_
07/01/2001 00:02:23 - I:COM:_A0103Y1A_
07/01/2001 00:02:24 - O:COM:_T010312421052_
07/01/2001 00:02:25 - I:COM:_A0103Y1A_
07/01/2001 00:11:17 - I:COM:_O0104225091          1NUR PALIY52_
07/01/2001 00:11:17 - O:COM:_A0104Y1D_
07/01/2001 00:11:23 - O:COM:_T01041250915F_
07/01/2001 00:11:24 - I:COM:_A0104Y1D_
07/01/2001 00:11:25 - O:COM:_T01041250915F_
07/01/2001 00:11:26 - I:COM:_A0104Y1D_
07/01/2001 00:16:39 - I:COM:_O0105225091          1NUR PALIY53_
07/01/2001 00:16:39 - O:COM:_A0105Y1C_
```

Figure 64: Sample Televic protocol data

Televic is an extended two-direction protocol, which provides a number of protocol rules to keep the communication secure; for example, through sequencing each packet, requesting acknowledge string, handshake through clear-string, return of feedback on message delivery through what is called terugmelding string, error detection through checksum, and so on. Refer to the protocol specifications for details on Televic protocol.

Regarding configuration, the following details are important:

The datastream contains a pager indication. This pager indication is used as a group indication.

Alarms can either be sent as type 1 (*SET of an alarm that has a *RESET type 0), type 0 (*RESET of an alarm that was previously *SET through code 1) and finally a type 2 (*SET of an alarm that does receive a *RESET). As a result, messages can be *SET or *RESET with remove after *SENT or remove after *RESET.

The DECT Messenger implements three distinct alarm descriptions.

1. The highest priority is assigned to alarm types that are configured through the specified length L:xx of the alarm text (For example, L:03 : NUR, ASS, SAN, REA, MUG, and so on). If this definition is found, the alarm attributes are fetched there.
2. When the first definition is unavailable, alarm types are fetched equal to the tone code (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). If this definition is found, the alarm attributes are fetched there.
3. As the last option, the alarm types are fetched through *OTHER special value. If this definition is found, alarm attributes are fetched there. In absence of any of the three definitions, the alarms are ignored.

The alarm message is retrieved from the datastream (NUR PAL, POORT 100, and so on).

TYCO

The valid manufacture is TYCO and the valid model is MINERVA 80.

```
   FIRE     EVACUATE    GROUND FLOOR           12:13:14 22 Nov 05
                        BGU.FRONT HALL EAST    Zn 0110  F   63
  CLEAR                 GROUND FLOOR           12:13:24 22 Nov 05
                        BGU.FRONT HALL EAST    Zn 0110  F   63
  SYSTEM    SILENCE     Fire System Zone       12:13:54 22 Nov 05
                        Silence                Zn 0300 LB00 S02
  SYSTEM     RESET                             12:13:56 22 Nov 05
                        Non Addressable Pnt.   Panl 01 MP
  SYSTEM     RESET      All Zones              12:13:56 22 Nov 05
                        Unknown                Panl 03 MP
ISOLATE STATUS  ON      System Zone            12:13:58 22 Nov 05
                        Non Addressable Pnt.   Zn 0100
PALM BEAMS ISOLATED                            12:13:58 22 Nov 05
                        Non Addressable Pnt.   Panl 01 MP   R02
 PRINTER     OFFLINE    System Zone            12:13:58 22 Nov 05
                        Non Addressable Pnt.   Zn 0400 LB00 R04
   FIRE      ALARM      GROUND FLOOR           12:16:20 22 Nov 05
                        SD.SAFE DEPOSIT ROOM   Zn 0110  F   7
  SYSTEM    SILENCE     Fire System Zone       12:16:30 22 Nov 05
                        Silence                Zn 0300 LB00 S02
  SYSTEM     RESET                             12:16:38 22 Nov 05
                        Non Addressable Pnt.   Panl 01 MP
  SYSTEM     RESET      Remote Panel Zone      12:16:38 22 Nov 05
                        Remote Panel Point     Panl 03 MP
ISOLATE STATUS  ON      System Zone            12:16:40 22 Nov 05
                        Non Addressable Pnt.   Zn 0100
PALM BEAMS ISOLATED                            12:16:40 22 Nov 05
                        Non Addressable Pnt.   Panl 01 MP   R02
 PRINTER     OFFLINE    System Zone            12:16:44 22 Nov 05
                        Non Addressable Pnt.   Zn 0400 LB00 R04
```

```
   FIRE     EVACUATE    GROUND FLOOR           12:13:14 22 Nov 05
                        BGU.FRONT HALL EAST    Zn 0110  F   63
```

Figure 65: Sample MINERVA 80 protocol data

An alarm is sent to the group MINERVA and the alarm description FIRE_EVACUATE, as well as a remove after *SENT and message BGU.FRONT HALL EAST. The group is always MINERVA and the alarm description comes from the first line when _ symbol appears. The message is retrieved from the second line, for example, BGU.FRONT HALL EAST.

VSK

Valid manufacturer is VSK.

Valid models are:

- DE LICHTERVELDE
- OLV VAN VREDE
- ST-JOZEF

VSK is based upon the three installations defined in the model, with a different implementation for each, as illustrated in Figure 66: Sample DE LICHTERVELDE protocol data on page 113, Figure 67: Sample OLV VAN VREDE protocol data on page 114, and Figure 68: Sample ST-JOZEF protocol data on page 115.

```
19990329 11:58:29 R:----------------------
19990329 11:58:29 R:D:HOOFDBORD+CENTR.L707
19990329 11:58:29 R:Z:043 D:0945
19990329 11:58:29 R:29-03-99 11:59 DET.FOUT
19990329 11:59:47 R:----------------------
19990329 11:59:47 R:S:GENERAL RESET
19990329 11:59:47 R:S:9130
19990329 11:59:47 R:29-03-99 12:00 SYS.FOUT
19990329 12:08:27 R:----------------------
19990329 12:08:27 R:S:GENERAL RESET
19990329 12:08:27 R:S:9130
19990329 12:08:27 R:29-03-99 12:09 SYS.FOUT
19990329 12:08:34 R:----------------------
19990329 12:08:34 R:S:GEEN PAPIER MEER !
19990329 12:08:34 R:S:9098
19990329 12:08:34 R:29-03-99 12:09 SYS.FOUT
19990329 12:08:59 R:----------------------
19990329 12:08:59 R:S:BUZZER RESET
19990329 12:08:59 R:S:9131
19990329 12:08:59 R:29-03-99 12:09 SYS.FOUT
19990329 12:21:06 R:----------------------
19990329 12:21:06 R:D:GANG            G710
19990329 12:21:06 R:Z:043 D:0931
19990329 12:21:06 R:29-03-99 12:21 DET.FOUT
19990329 12:21:35 R:----------------------
```

Figure 66: Sample DE LICHTERVELDE protocol data

```
19981117 00:14:30 R:S:GENERAL RESET
19981117 00:14:30 R:S:9130
19981117 00:14:30 R:17-11-98 00:11 SYS.FOUT
19981117 00:14:53 R:-----------------------
19981117 00:14:53 R:D:BEZOEKZAAL       303
19981117 00:14:53 R:Z:003 D:0193
19981117 00:14:53 R:17-11-98 00:11 DET.FOUT
19981117 00:16:03 R:-----------------------
19981117 00:16:03 R:D:BEZOEKZAAL       303
19981117 00:16:03 R:Z:003 D:0193
19981117 00:16:03 R:17-11-98 00:13 DET.FOUT
19981117 00:21:41 R:-----------------------
19981117 00:21:41 R:S:BUZZER RESET
19981117 00:21:41 R:S:9131
19981117 00:21:41 R:17-11-98 00:18 SYS.FOUT
19981117 05:47:14 R:-----------------------
19981117 05:47:14 R:S:NACHTBEL       POORT
19981117 05:47:14 R:S:9176      GP01
19981117 05:47:14 R:17-11-98 05:44 SYS.FOUT
19981117 05:47:51 R:-----------------------
19981117 05:47:51 R:S:BUZZER RESET
19981117 05:47:51 R:S:9131
19981117 05:47:51 R:17-11-98 05:44 SYS.FOUT
19981117 06:32:10 R:-----------------------
19981117 06:32:10 R:S:NACHTBEL       POORT
19981117 06:32:10 R:S:9176      GP01
19981117 06:32:10 R:17-11-98 06:29 SYS.FOUT
19981117 06:40:25 R:-----------------------
```

Figure 67: Sample OLV VAN VREDE protocol data

```
19980318 11:30:44 ----------------------
19980318 11:30:44 D:KAMER 1
19980318 11:30:45 ZONE #00
19980318 11:30:45 Z:000 D:0001
19980318 11:30:46 26-01-98 04:13 BRAND
19980318 11:40:54 S:VERWIJDER CFG. STRAP
19980318 11:40:54 S:9146
19980318 11:40:55 26-01-98 04:12 SYS.FOUT
19980318 11:40:57 D:KAMER 56
19980318 11:40:57 Z:001 D:0056
19980318 11:40:58 26-01-98 04:12 ISOLATIE
19980318 11:40:58 ----------------------
19980318 11:40:59 D:KAMER 1
19980318 11:40:59 ZONE #00
19980318 11:41:00 Z:000 D:0001
19980318 11:41:00 26-01-98 04:13 BRAND
19980318 11:42:48 S:VERWIJDER CFG. STRAP
19980318 11:42:49 S:9146
19980318 11:42:49 26-01-98 04:12 SYS.FOUT
19980318 11:44:00 D:KAMER 56
19980318 11:44:01 Z:001 D:0056
19980318 11:44:01 26-01-98 04:12 ISOLATIE
19980318 11:44:02 ----------------------
19980318 11:44:02 D:KAMER 1
19980318 11:44:03 ZONE #00
19980318 11:44:03 Z:000 D:0001
19980318 11:44:04 26-01-98 04:13 BRAND
19980318 11:44:23 ----------------------
19980318 11:44:24 S:NETONDERBREKING
19980318 11:44:24 S:9048      L1.0
19980318 11:44:25 26-01-98 04:13 SYS.FOUT
19980318 11:44:37 ----------------------
19980318 11:44:38 D:KAMER 49
19980318 11:44:38 Z:001 D:0049
19980318 11:44:39 26-01-98 04:14 DET.FOUT
19980318 11:46:14 ----------------------
19980318 11:46:14 S:VERWIJDER CFG. STRAP

continued on next page...

19980318 11:46:15 S:9146
19980318 11:46:15 26-01-98 04:17 SYS.FOUT
19980318 11:46:29 S:VERWIJDER CFG. STRAP
19980318 11:46:29 S:9146
19980318 11:46:30 26-01-98 04:12 SYS.FOUT
19980318 11:46:34 D:KAMER 56
19980318 11:46:34 Z:001 D:0056
19980318 11:46:35 26-01-98 04:12 ISOLATIE
19980318 11:46:35 ----------------------
```

Figure 68: Sample ST-JOZEF protocol data

Fire alarms are sent to group VSK_F (fire alarm) with alarm description VSK_F, system errors are sent to group VSK_S (system errors) with alarm description VSK_S, detector errors are sent to group VSK_D (detector errors) with alarm description VSK_D.

All alarms are *SET with remove after *RESET. When GENERAL RESET occurs in the datastream, all active alarms for the VSK input program are reset for all groups.

Small differences between the three models are found, for example, in the level of detail in the messages that are sent (for example, FOUT BRANDCENTRALE in DE LICHTERVELDE, VSKFOUT in ST JOZEF and all details in OLV VAN VREDER). Also, fire alarm messages are formatted slightly differently between the models (BRAND or BR*xxxxx where xxxxx denotes a location).

Note:
Use of this protocol usually requires consulting services and customizing.

WORMALD

The valid manufacturer is WORMALD and the valid model is *BASE, L:xx (new in release 2.9.11), or G:xxxx (new in release 3.1).

Wormald is based upon installation Alexianen Bouchout and RUCA Antwerpen and is compatible with both versions through automatic-protocol-recognition programming.

The model L:xx defines the group. The model *BASE, identifies the group WORMALD_F or WORMALD_P depending on the type of alarm (see the examples Table 9: Alexianen on page 116 and Table 12: RUCA on page 118).

If the model = L:xx, the first xx characters of the message define the group.

The model G:xxxx is new in release 3.1.0 and defines the group used for all the alarms. The group is xxxx, for example, in G:EIPM, no message is sent to the group EIPM.

If the group is not defined in the Messenger_CFG.mdb database, the eKernel application sends the request to the group identified with a question mark (?), if one is defined in the table eKERNEL_GROUP (field GRP_Descr_str). This feature is supported for eCAP WORMALD only when model = L:xx.

Examples:

Table 9: Alexianen

```
27/04/99 15:30:23
ALARM 001-084 DK
8000 REV EVACUA
```

This datastream results in *SET of alarm with alarm description WORMALD_F and message F8000 REV EVACUA, remove after *RESET.

The group varies depending on the model

Table 10: F8000 REV EVACUA groups

Model	Group
*BASE	WORMALD_F.
L:01	8 (group description: field GRP_Descr_str from table eKERNEL_GROUP)
L:02	80
L:03	800
L:04	8000
G:BRAND	BRAND (for all alarms, including fire and pre-alarms)

```
27/04/99 19:55:52
VOORALARM 0001-024 ION
9003 KEU EETPL
```

This datastream results in *SET of alarm with alarm description WORMALD_P and message P9003 KEU EETPL, remove after *RESET.

The group varies depending on the model.

Table 11: P9003 KEU EETPL groups

Model	Group
*BASE	WORMALD_P.
L:01	9
L:02	90
L:03	900
L:04	9003
G:BRAND	BRAND (for all alarms including fire and pre-alarms)

```
27/04/99 19:56:12
Reset
```

This datastream results in a *RESET of alarm description *ALL for group *ALL and message *ALL.

Table 12: RUCA

```
0000.  Brand                MG 099  MLD 009              01:01:00 01.01.97
       Boodschap
```

This datastream results in a *SET of alarm description WORMALD_F, with message Boodschap and remove after *RESET.

The group varies depending on the model

Table 13: F8000 REV EVACUA groups

Model	Group
*BASE	WORMALD_F.
L:01	B
L:02	Bo
L:03	Boo
L:04	Bood
G:EIPM	EIPM for all alarms

```
0001. Reset
```

This datastream results in *RESET of alarm description *ALL for group WORMALD_F with message *ALL and remove after *RESET.

Note:
Use of this protocol usually requires consulting services and customization.

Generic

Valid manufacturer is GENERIC, valid model is *BASE.

This new manufacturer and model combination is implemented in release 2, to handle fixed-formatted serial inputs that are built upon single lines and separated with a fixed character.

There are many parameters available to define the interpretation of the datastreams that are received through the generic eCAP implementation. These parameters define criteria for retrieval of group, message, alarm description, set_or_reset and remove_after parameters. There are also default values available if some parameters are missing. Another set of parameters describes record separators, field separators, and so on.

Chapter 15: Module - eESPA

The module eESPA consists of one program. The program is eESPA.exe and is written in Visual Basic (v6.0).

In general, the programs reside in the default directory C:\SOPHO Messenger@Net\Exe, unless otherwise implemented in your environment.

Note:

In release 4.0 an additional implementation is available that you can activate by specifying manufacturer ESPA and model VSK. In most cases, the new VSK implementation is likely to be more compatible than the *BASE system and therefore the preferred method to configure the eESPA module.

Manufacturer ESPA and model BASE

For backwards compatibility, to activate the original implementation of module eESPA by specifying manufacturer ESPA and model *BASE.

Overview

eESPA.exe

The eESPA.exe is a Visual Basic component of the eESPA module. The program communicates with two processes: the eKERNEL.exe and external paging equipment. The eKERNEL.exe is the central engine that centralizes all database access and communication with input and output capable modules.

The eESPA.exe communicates with eKERNEL.exe by means of TCP sockets. In this communication, eESPA.exe is a TCP client software that connects to the eKERNEL component, acting as TCP server software.

At start up, eESPA.exe contacts the eKERNEL.exe by means of a socket connection. eESPA.exe requires at start up a few parameters, so that eESPA can identify itself and locate the eKERNEL.exe program. This is carried out by means of parameters in the properties

section of the shortcut that initiates eESPA.exe. This shortcut is usually located in the Windows Startup group.

```
"C:\SOPHO Messenger@Net\Exe\eESPA.exe"
/Site:1
/eKernel address:*LOCAL
/eKernel port:3114
/Log drive:C
/SleepBeforeAnswer:0
```

Figure 69: Typical parameters in the shortcut

In the example shown in Figure 69: Typical parameters in the shortcut on page 120, the eESPA.exe identifies itself as belonging to site 1, and specifies the location of eKERNEL through IP address *LOCAL and port 3114. The special value *LOCAL refers to the assigned IP address of the first NIC adapter found in the PC. You can determine the IP address using the IPCONFIG.exe command or in the appropriate sections of the Windows network settings. The keyword Log drive refers to the drive where the logging data must be stored. Usually this is C:\SOPHO Messenger@Net\Log\.

The keyword SleepBeforeAnswer refers to the number of milliseconds the system waits before sending an answer to the linked station. Some systems block the answer if the answer is sent immediately (less than 6 milliseconds after receipt). Use this parameter to set a delay of x milliseconds before the output is sent.

Note:

The maximum supported value is 150 milliseconds

Set this value as low as possible, because during the delay time, the application is inactive, so high values can disrupt the operation of the system.

At start up, the eESPA.exe sends an XML string to eKERNEL.exe requesting a configuration. This step is needed for each module that interacts with eKERNEL.exe, because this approach allows central administration from a single database, even if some client modules are located on a distributed machine.

```
<xml>
<cfgrqs>
<appl>eESPA</appl>
<site>1</site>
</cfgrqs>
</xml>

<xml>
<cfgrpy>
<resource>COM01</resource>
<settings>9600,N,8,1</settings>
<link_type>RS232</link_type>
<ControlStation>Y</ControlStation>
<polling_intv>300</polling_intv>
<Polling_address_list>2</Polling_address_list>
<localAddress>1</localAddress>
<externalAddress>2</externalAddress>
<dataId_Group>1</dataId_Group>
<group_default>eESPA</group_default>
<dataId_Msg>2</dataId_Msg>
<msg_default>Default ESPA msg</msg_default>
<dataId_Ala_descr>3</dataId_Ala_descr>
<ala_descr_default>1</ala_descr_default>
<remove_after>*RESET</remove_after>
<nak_retry_cnt>2</nak_retry_cnt>
<timeout>600</timeout>
<handshaking>0</handshaking>
<log_path>C:\SOPHO Messenger@net</log_path>
<log_days>14</log_days>
</cfgrpy>
</xml>
```

Figure 70: Typical cfgrqs configuration request and its received cfgrpy configuration reply

Refer to the appropriate sections on the database tables that define the received parameters for more information on each value. The information in this document is provided for informational purposes only; a detailed description of these internal inter-process communications is beyond the scope of this document.

If no valid license is available to run an eESPA module, the eKERNEL sends a reply <xml><pgmsts>NO LICENSE AVAILABLE</pgmsts></xml> and close its port. If you receive this message upon connection to the eKernel, use Avaya License Manager to see if there is an available license.

When the configuration is received, the **Connection** tab displays information similar to what is pictured in Figure 71: eESPA Connection tab showing a valid configuration on page 122.

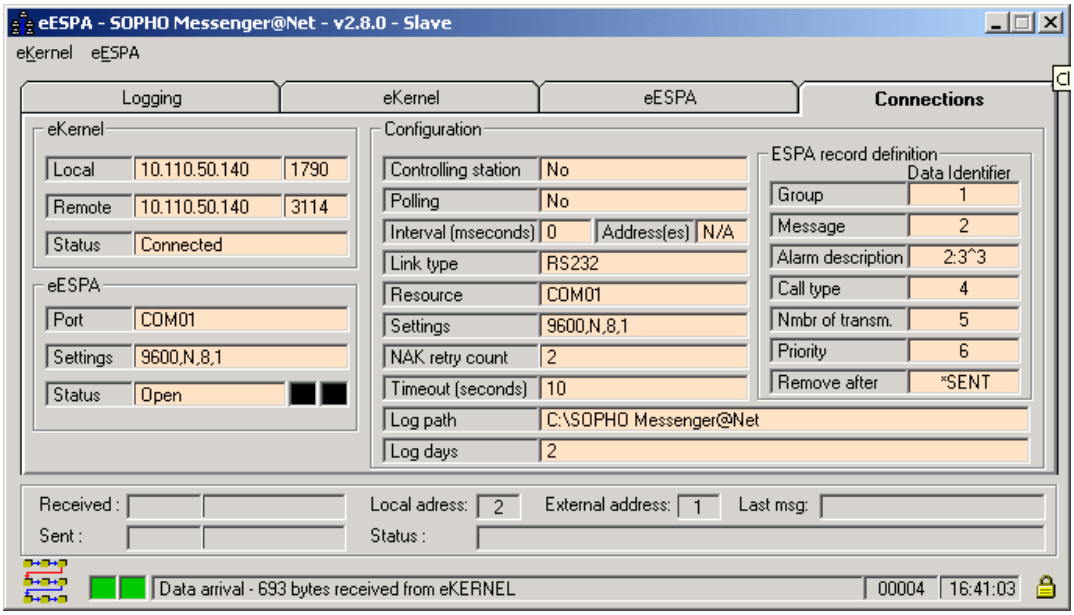


Figure 71: eESPA Connection tab showing a valid configuration

Because the eESPA is designed to handle serial data communications with a number of paging systems, the eESPA must be configured before use. These values are returned through the <cfgrpy> reply that is sent on return of the <cfgrqs> request.

Note:

The values shown in Figure 71: eESPA Connection tab showing a valid configuration on page 122 are retrieved from the DECT Messenger database; from the KERNEL_INPGM table, and from the eESPA, eESPA_OUTBOUND_CFG, and the eKERNEL_SITE tables.

At start up, the eESPA.exe appears the specified COM port with the specified settings.

Note:

The COM port you identify must be available, and you must specify a valid baud rate, and so on.

As well, a physical connection between the specified COM port and the external paging system must be available, which requires a serial cable. In most cases, you can use a standard null-modem cable. However, in some cases, alarm systems support a limited number of control signals (for example, ground, send and receive), so consult the alarm system vendor for cable specifications.

If no COM ports are available, extra hardware (such as DigiBoard PC/4e and DigiBoard PC/8e) is required to provide extra serial ports. Investigate compatibility issues (operating system support, driver available, and so on) and hardware requirements (memory, available slots, IRQ conflicts, and so on) before purchasing equipment, or configuring the system.

In some cases, additional hardware is needed, for example, when RS-232-C limitations apply (for example, at 9600 baud maximum limit of 9 metres). In some cases galvanic isolation is requested, or base-band modems, SOPHO LAM, CISCO equipment, and so on are needed to bridge the distance between the DECT Messenger and the alarm system. In many cases

the distance between the DECT Messenger and the external-paging system is relatively small, so no extra hardware is needed.

Once a link is established between eESPA and the paging system, the eESPA handles further communications and informs eKERNEL when relevant information is to be exchanged.

For a detailed description of the ESPA4.4.4 protocol, refer to the proposal for serial data interface for paging equipment (Nov. 1984), reference JMJ182/NB/12B, and ISO1745 Information processing, basic mode control procedures for data communication systems.

Functional description

In general, eESPA is designed to provide eKERNEL with paging information. This is carried out using a <msgrqs> message request. For some interfaces eKERNEL must send feedback to the paging system, a process that is handled through <msgrpy> message reply request.

```
<xml><msgrqs>
<id>00786</id>
<group>1234</group>
<call_type>3</call_type>
<transmission_nmbr>1</transmission_nmbr>
<alarm_cnt>1</alarm_cnt>
<message_01>NURSE ROOM45</message_01>
<beep_code_01>1</beep_code_01>
<priority_01>2</priority_01>
</msgrqs></xml>
```

Figure 72: msgrqs message request

Every <msgrqs> message belongs to a group and has a specific group ID. A group contains one or more requests. For every message request in a group, a data block is created. A data block consists of a header, record separators, unit separators, and data that is retrieved from the message request. Every data block also contains a specifically calculated checksum (block check character ISO 1155). After sending the data over the serial line, the receiving side uses the block check character (BCC) to determine whether data has arrived properly or not. In the event of a successful delivery, the receiving side answers with an ACK.

```
<xml><msgrpy>
<id>00786</id>
<sts_cnt>1</sts_cnt>
<sts_01>ACK^</sts_01>
</msgrpy></xml>
```

Figure 73: msgrpy message ACK

If an incorrect BCC is found, delivery fails, and the receiving side sends a NACK, which is prefixed with error code 1 (Transmission error, corrupt characters or corrupt BCC received by the station).

eESPA handles only data blocks of type 1, Call to Pager data blocks. If another type of data block comes in, eESPA reacts by sending an ACK, but the data block is not processed.

Delivery can also fail if a timeout occurs while sending the data block. The temporary master station, which is always the sending side, expects to receive an ACK within a timeout of eESPA_Timeout_n seconds. In the event of a timeout on sending a data block, the sending side tries to re-send the data block. This retry is attempted x times (where x is the defined value of the field eESPA_NAK_retry_cnt_n in the eESPA table). After retrying x times and not receiving an ACK, the temporary master station decides that the transaction is unsuccessful.

```
<xml><msgrpy>
<id>00786</id>
<sts_cnt>1</sts_cnt>
<sts_01>NACK^</sts_01>
</msgrpy></xml>
```

Figure 74: msgrpy message NACK

Data flow

The ESPA4.4.4 protocol prescribes a controlling station that polls devices on the communication line. Polling means sending out requests for data. The polling device, which is also called controlling station or master, sends out requests to the other devices available on the communication line. Every device on the line has a specific address. The characters 0 to 9 are available as addresses. Avaya recommends that you assign the character 1 to the controlling station. In the field eESPA_Polling_address_list_str of the table eESPA you can define multiple addresses of slave devices. At least one address (that represents a slave) is required in this field. Multiple addresses are separated by ^.

For example: the value 2^4^5 in eESPA_Polling_address_list_str defines three slaves that are polled by the controlling station.

Define a control station by placing eESPA_ControlStation_b in the eESPA table on True. eESPA_LocalAddress_n and eESPA_ExternalAddress_n must be filled up respectively with the local address of the module (a controlling station prefers a local address 1), and the address of an external eESPA module or device, with which the module communicates.

The controlling station polls every address with an enquiry. To extend the example, the controlling station sends 2ENQ, 4ENQ, and 5ENQ. The field eESPA_Polling_intv_n in the eESPA table defines the time between sequencing polls.

A slave whose address is polled reacts by replying with either a nothing-to-transmit (EOT), or an enquiry (<master address>ENQ) that tells the controlling station that this slave wants to transmit some data. If a slave receives data that is not assigned to the slave address, the slave ignores the data. If a slave does not respond to an enquiry within eESPA_Timeout_n seconds, the controlling station places this slave in a special offline list. When polling, the controlling station reads the offline list to determine if a slave is online or not, before sending an enquiry. After 60 seconds, the slave is removed from the offline list, and polling to the IP address of the slave is restarted. If the slave does not react, the slave is put back in the offline list. By using

an offline list (also known as a black list) the polling interval is not disturbed by repeated timeouts of some slaves.

The controlling station stops polling when data is waiting to be sent to one of the slaves, or when a slave has indicated that data is ready to send.

The controlling station stops polling when a message request is received from the eKERNEL. The controlling station then creates a data block and sends this to the appropriate slave address. A slave station sends data in the same fashion, but with one difference: the slave first has to wait to be polled to tell the controlling station that data is ready to send. When a slave is polled and has data to send, the slave tells the controlling station to stop polling. The controlling station accepts this request by sending an ACK to the slave that wants to send data. When it receives an ACK from the controlling station, the slave becomes temporarily master station. Only a master station is able to send data blocks. A master station always sends data; a slave always receives. In this scenario, the controlling station becomes (temporarily) a slave.

Every eESPA module has a status bar with some additional information about the actual communication situation.

- The label Receiving shows the timestamp and the latest incoming databits. Values that can appear on this label are as follows: ACK, NAK, EOT, address + ENQ, SOH (start of data), and data.
- The label Sent shows the timestamp and the databit that was last sent. Values that can appear on this label are as follows: ACK, NAK, EOT, address + ENQ, SOH (start of data), and data.
- The Local address field shows the defined value of eESPA_LocalAddress_n in the eESPA table. Avaya recommends that you set the controlling station local address to 1. The field eESPA_ControlStation_b in the eESPA table defines whether this module is a controlling station or not.

  In the DECT Messenger model, a module always communicates with one other module (master to slave or point to point). The address of the other station is defined in the eESPA_ExternalAddress_n field in the eESPA table. In the case of a non-controlling module (defined by setting eESPA_ControlStation_b on No in the database), for example, you can use a local address of 2 and an external address of 1. In this case, the module communicates directly with the controlling station.
- The label Status shows log information to allow users to see special actions.

Figure 75: Status of master/slave on page 125 shows an example of the eESPA status bar.

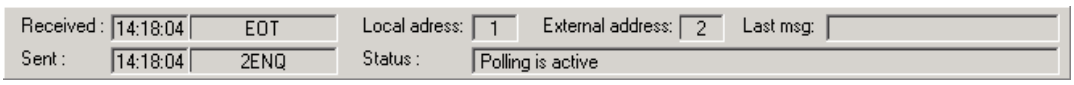


Figure 75: Status of master/slave

Logging

In the **Logging** tab, only basic (default) logging is shown.

Basic logging: incoming and outgoing data on the communication port (I:COM and O:COM), incoming and outgoing data on the socket communication with the eKernel (I:TCP and O:TCP), and warning information.

To show and log additional information, choose the menu item **eESPA > Logging > Detailed**. The additional information is set in bold in Figure 76: Detailed logging information on page 126.

```
I:INF:--> Created datablock is: _1_156789-2eAPI Koekoek-51-63_O
O:COM:_1_156789-2eAPI Koekoek-51-63_O
I:INF:--> ACK received on sending datablock
O:COM:EOT
I:INF:--> Concluded transaction with sending an EOT
I:COM:ACK
I:COM:EOT
I:INF:--> Received EOT character after sending messages.
O:COM:EOT
O:TCP:<xml><msgrpy><id>00124</id><sts_cnt>1</sts_cnt>
<sts_01>ACK^</sts_01></msgrpy></xml>
```

Figure 76: Detailed logging information

The label Last message shows the last sent or received message.

Manufacturer ESPA and model ASCOM

In release 4.0 an additional implementation of ESPA 4.4.4 protocol is available. You can activate it by defining manufacturer ESPA and model ASCOM. For further information, contact Unified Communications Professional Services (UCPS) division for more information.

Manufacturer ESPA and model VSK

In release 4.0 an additional implementation of ESPA 4.4.4 protocol is available, and is activated by defining the manufacturer ESPA and model VSK.

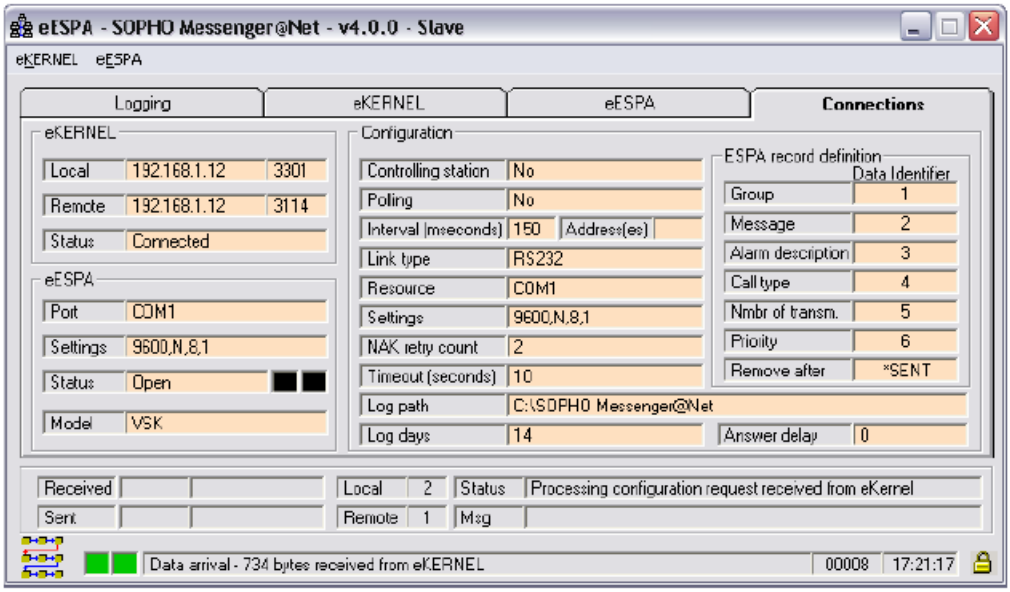


Figure 77: Field model containing VSK

Restrictions:

- Configure eESPA as Control station value No. The remote system is then control station and handles polling.
- Set the field Polling to No.
- Only one data block is received at a time.

In the example shown in Figure 78: 2[ENQ] with response [EOT] on page 127 it is assumed:

- The remote system (master) is using address 1
- The local system (slave) is using address 2

Different application logic and code is used when model VSK is used, mainly because different polling is supported. The original code assumes polling is 2[ENQ] with the response [EOT].

```
06/06/2007 18:38:11 - I:COM:2
06/06/2007 18:38:11 - I:COM:[ENQ]
06/06/2007 18:38:11 - O:COM:[EOT]
```

Figure 78: 2[ENQ] with response [EOT]

VSK implementation also supports local polling of the control station. Note that 1[ENQ][EOT] is received and no response is made.

```
06/06/2007 18:38:11 - I:COM:1
06/06/2007 18:38:11 - I:COM:[ENQ]
06/06/2007 18:38:11 - I:COM:[EOT]
```

Figure 79: 1[ENQ][EOT], no response

For example, the FalconNet fire detection system alternatively polls itself and then the remote system (eESPA), as shown in Figure 80: FalconNet fire detection system on page 128.

```
06/06/2007 18:38:01 - I:COM:1
06/06/2007 18:38:01 - I:COM:[ENQ]
06/06/2007 18:38:01 - I:COM:[EOT]
06/06/2007 18:38:03 - I:COM:2
06/06/2007 18:38:03 - I:COM:[ENQ]
06/06/2007 18:38:03 - O:COM:[EOT]
```

Figure 80: FalconNet fire detection system

To improve problem determination, a more detailed and human readable logging is enabled when the model VSK is used.

```
06/06/2007 18:37:37 - S:INF:COM01 opened with settings 9600,N,8,1
06/06/2007 18:38:06 - I:COM:1
06/06/2007 18:38:06 - I:COM:[ENQ]
06/06/2007 18:38:06 - I:COM:2
06/06/2007 18:38:06 - I:COM:[ENQ]
06/06/2007 18:38:06 - O:COM:[ACK]
06/06/2007 18:38:10 - I:COM:2
06/06/2007 18:38:10 - I:COM:[ENQ]
06/06/2007 18:38:10 - O:COM:[EOT]
06/06/2007 18:38:11 - I:COM:2
06/06/2007 18:38:11 - I:COM:[ENQ]
06/06/2007 18:38:11 - O:COM:[EOT]
06/06/2007 18:38:12 - I:COM:2
06/06/2007 18:38:12 - I:COM:[ENQ]
06/06/2007 18:38:12 - O:COM:[EOT]
06/06/2007 18:38:13 - I:COM:1
06/06/2007 18:38:13 - I:COM:[ENQ]
06/06/2007 18:38:13 - I:COM:[EOT]
06/06/2007 18:38:15 - I:COM:2
06/06/2007 18:38:15 - I:COM:[ENQ]
06/06/2007 18:38:15 - O:COM:[EOT]
06/06/2007 18:38:17 - I:COM:1
06/06/2007 18:38:17 - I:COM:[ENQ]
06/06/2007 18:38:17 - I:COM:[EOT]
06/06/2007 18:38:17 - I:COM:2
06/06/2007 18:38:17 - I:COM:[ENQ]
06/06/2007 18:38:17 - O:COM:[EOT]
06/06/2007 18:38:18 - I:COM:1
06/06/2007 18:38:18 - I:COM:[ENQ]
06/06/2007 18:38:18 - I:COM:[EOT]
06/06/2007 18:38:18 - I:COM:2
06/06/2007 18:38:18 - I:COM:[ENQ]
06/06/2007 18:38:18 - O:COM:[EOT]
06/06/2007 18:38:19 - I:COM:1
06/06/2007 18:38:19 - I:COM:[ENQ]
06/06/2007 18:38:19 - I:COM:[EOT]
06/06/2007 18:38:19 - I:COM:2
06/06/2007 18:38:19 - I:COM:[ENQ]
06/06/2007 18:38:19 - O:COM:[EOT]
06/06/2007 18:38:20 - I:COM:1
06/06/2007 18:38:20 - I:COM:[ENQ]
06/06/2007 18:38:20 - I:COM:[EOT]
06/06/2007 18:38:20 - I:COM:2
06/06/2007 18:38:20 - I:COM:[ENQ]
06/06/2007 18:38:20 - O:COM:[EOT]
06/06/2007 18:38:23 - I:COM:[EOT]
06/06/2007 18:38:23 - I:COM:2
06/06/2007 18:38:23 - I:COM:[ENQ]
06/06/2007 18:38:23 - O:COM:[EOT]
06/06/2007 18:38:24 - I:COM:[EOT]
06/06/2007 18:38:24 - I:COM:2
06/06/2007 18:38:24 - I:COM:[ENQ]
06/06/2007 18:38:24 - O:COM:[EOT]
06/06/2007 18:38:26 - I:COM:1
06/06/2007 18:38:26 - I:COM:[ENQ]
```

Figure 81: Logging using model VSK

```
06/06/2007 18:38:26 - I:COM:[EOT]
06/06/2007 18:38:27 - I:COM:2
06/06/2007 18:38:27 - I:COM:[ENQ]
06/06/2007 18:38:27 - O:COM:[EOT]
06/06/2007 18:38:28 - I:COM:1
06/06/2007 18:38:28 - I:COM:[ENQ]
06/06/2007 18:38:28 - I:COM:2
06/06/2007 18:38:28 - I:COM:[ENQ]
06/06/2007 18:38:28 - O:COM:[ACK]
06/06/2007 18:38:30 - I:COM:[SOH]
06/06/2007 18:38:30 - I:COM:1
06/06/2007 18:38:30 - I:COM:[STX]
06/06/2007 18:38:30 - I:COM:1
06/06/2007 18:38:30 - I:COM:[US]
06/06/2007 18:38:30 - I:COM:G
06/06/2007 18:38:30 - I:COM:R
06/06/2007 18:38:30 - I:COM:O
06/06/2007 18:38:30 - I:COM:E
06/06/2007 18:38:30 - I:COM:P
06/06/2007 18:38:30 - I:COM:B
06/06/2007 18:38:30 - I:COM:+
06/06/2007 18:38:30 - I:COM:3
06/06/2007 18:38:30 - I:COM:[RS]
06/06/2007 18:38:30 - I:COM:2
06/06/2007 18:38:30 - I:COM:[US]
06/06/2007 18:38:30 - I:COM:T
06/06/2007 18:38:30 - I:COM:6
06/06/2007 18:38:30 - I:COM::
06/06/2007 18:38:30 - I:COM:
06/06/2007 18:38:30 - I:COM:*
06/06/2007 18:38:30 - I:COM:A
06/06/2007 18:38:30 - I:COM:S
06/06/2007 18:38:30 - I:COM:Y
06/06/2007 18:38:30 - I:COM:S
06/06/2007 18:38:30 - I:COM:T
06/06/2007 18:38:30 - I:COM:O
06/06/2007 18:38:30 - I:COM:L
06/06/2007 18:38:30 - I:COM:I
06/06/2007 18:38:30 - I:COM:E
06/06/2007 18:38:30 - I:COM:
06/06/2007 18:38:30 - I:COM:H
06/06/2007 18:38:31 - I:COM:F
06/06/2007 18:38:31 - I:COM:
06/06/2007 18:38:31 - I:COM:0
06/06/2007 18:38:31 - I:COM:
06/06/2007 18:38:31 - I:COM:,
06/06/2007 18:38:31 - I:COM:
06/06/2007 18:38:31 - I:COM:M
06/06/2007 18:38:31 - I:COM:a
06/06/2007 18:38:31 - I:COM:r
06/06/2007 18:38:31 - I:COM:i
06/06/2007 18:38:31 - I:COM:e
06/06/2007 18:38:31 - I:COM:
```

```
06/06/2007 18:38:31 - I:COM:3
06/06/2007 18:38:31 - I:COM:4
06/06/2007 18:38:31 - I:COM:3
06/06/2007 18:38:31 - I:COM:/
06/06/2007 18:38:31 - I:COM:1
06/06/2007 18:38:31 - I:COM:
06/06/2007 18:38:31 - I:COM:M
06/06/2007 18:38:31 - I:COM:a
06/06/2007 18:38:31 - I:COM:s
06/06/2007 18:38:31 - I:COM:c
06/06/2007 18:38:31 - I:COM:h
06/06/2007 18:38:31 - I:COM:e
06/06/2007 18:38:31 - I:COM:l
06/06/2007 18:38:31 - I:COM:e
06/06/2007 18:38:31 - I:COM:i
06/06/2007 18:38:31 - I:COM:n
06/06/2007 18:38:31 - I:COM:[RS]
06/06/2007 18:38:31 - I:COM:4
06/06/2007 18:38:31 - I:COM:[US]
06/06/2007 18:38:31 - I:COM:3
06/06/2007 18:38:31 - I:COM:[RS]
06/06/2007 18:38:31 - I:COM:3
06/06/2007 18:38:31 - I:COM:[US]
06/06/2007 18:38:31 - I:COM:1
06/06/2007 18:38:31 - I:COM:[RS]
06/06/2007 18:38:31 - I:COM:6
06/06/2007 18:38:31 - I:COM:[US]
06/06/2007 18:38:31 - I:COM:1
06/06/2007 18:38:31 - I:COM:[ETX]
06/06/2007 18:38:31 - I:COM:,
06/06/2007 18:38:31 - S:INF:Processing of received data
[SOH]1[STX]1[US]GROEPB+3[RS]2[US]T6: ***ASYSTOLIE HF 0 , Marie 343/1
Masschelein[RS]4[US]3[RS]3[US]1[RS]6[US]1[ETX],
06/06/2007 18:38:31 - O:COM:[ACK]
06/06/2007 18:38:31 -
O:TCP:<xml><msgrqs><set_or_reset>*SET</set_or_reset><msg>T6:
***ASYSTOLIE
HF 0 , Marie 343/1
Masschelein</msg><alarmdescr>T</alarmdescr><group>GROEPB+3</group><remov
e_afte
r>*SENT</remove_after></msgrqs></xml>
06/06/2007 18:38:34 - I:COM:[EOT]
```

Chapter 16: Module - eESPA - sample

MASTER (address 1)		SLAVE (address 2)
No data to be transferred		
2ENQ		
		EOT
Master has data to be transferred		
1ENQ (I want to send something)		
2ENQ (Destination address)		
		ACK (I am ready to receive data)
Data Block1		
		1NAK
Data Block1		
		ACK
EOT		
		EOT
2ENQ (polling)		
Master has data to be transferred (Slave is not ready to receive data)		
1ENQ (I want to send something)		
2ENQ (Destination address)		
		1NAK (Transmission error)
EOT		
1ENQ		
2ENQ		

MASTER (address 1)		SLAVE (address 2)
		1NAK
EOT		
Slave has data to be transferred		
2ENQ		
		1ENQ (I have data for address 1)
ACK (I am ready to receive data)		
		DATA Block1
ACK		
		DATA Block2
1NAK (Transmission error)		
		DATA Block2
ACK		
		EOT
2ENQ (Polling)		
		EOT
2ENQ		
		EOT
...		

Chapter 17: Module - eDMSAPI

The module eDMSAPI consists of two separate programs. One program is eDMSAPI and is written in Visual Basic. The other program is called CSTA_Service.exe and is written in C++.

In general, both programs reside in the default directory C:\SOPHO Messenger@Net\Exe, unless otherwise implemented in your environment.

Overview

eDMSAPI.exe

The eDMSAPI is the Visual Basic component of the eDMSAPI module. The program communicates with two processes: the eKERNEL and the CSTA Service. The eKERNEL is the central engine that centralizes all database access and communication with input and output capable modules.

The eDMSAPI communicates with both eKERNEL and CSTA_Service.exe by means of TCP sockets. In both communications, eDMSAPI is a TCP client software that connects to the two other components, acting as TCP server software.

For external clients (eWEB), the eDMSAPI acts as a multiple socket server.

At start up, eDMSAPI contacts the eKERNEL by means of a socket connection. For the eDMSAPI module to locate the eKERNEL, the eDMSAPI must start up with parameters that identify the eDMSAPI module and locate the eKERNEL program. These parameters are provided to eDMSAPI in the Properties section of the shortcut that initiates eDMSAPI. This shortcut is usually located in the Windows Startup group (click **Start** on the Windows task bar and choose **Programs > Startup**).

```
"C:\SOPHO Messenger@Net\Exe\eDMSAPI.exe"
/Site:1
/eKernel port:3101
/eKernel address:*LOCAL
/Log drive:C
```

Figure 82: Typical parameters in the shortcut

In the example shown in Figure 82: Typical parameters in the shortcut on page 135, eDMSAPI identifies itself as belonging to Site 1, and specifies the location of eKERNEL through IP address *LOCAL and port 3101. The special value *LOCAL refers to the assigned IP address

of the first NIC adapter found in the PC. You can determine the IP address using the IPCONFIG.exe command or in the appropriate sections of the Windows network settings. The keyword Log drive refers to the drive where the logging data must be stored; usually this is C:\SOPHO Messenger@Net\Log\.

At startup, the eDMSAPI sends an XML string to eKERNEL requesting a configuration. This step is needed for each module that interacts with eKERNEL, because this approach allows central administration using a single database, even if some client modules are located on a distributed machine.

```
<xml>
<cfgrqs>
<appl>eDMSAPI</appl>
<site>1</site>
</cfgrqs>
</xml>

<xml>
<cfgrpy>
<seat_cnt>20</seat_cnt>
<msg_dly>3</msg_dly>
<csta_api_address>10.110.50.140</csta_api_address>
<csta_api_port>59000</csta_api_port>
<external_seat_cnt>3</external_seat_cnt>
<external_port>2010</external_port>
<csta_pbx_address>10.110.49.171</csta_pbx_address>
<csta_pbx_port>2555</csta_pbx_port>
<csta_licence>Messenger</csta_licence>
<guarding_intv>60</guarding_intv>
<guarding_Retry_intv>20</guarding_Retry_intv>
<eKernel_Seat_cnt>10</eKernel_Seat_cnt>
<GeneralTimeOut>15</GeneralTimeOut>
<Ack2TimeOut>30</Ack2TimeOut>
<DataPathDelay>2</DataPathDelay>
<network>ETHERNET_DMC</network>(new in Msg@Net R3.0)
<pbxtype>DMC</pbxtype>(new in Msg@Net R3.0)
<IoReg_cnt>3</IoReg_cnt>
<IoReg_0001>863</IoReg_0001>
<IoReg_0002>123</IoReg_0002>
<IoReg_0003>914</IoReg_0003>
><keepalive>60</keepalive>
<log_path>C:\SOPHO Messenger@net</log_path>
<log_days>14</log_days>
</cfgrpy>
</xml>
```

Figure 83: A typical cfgrqs configuration request and its received cfgrpy configuration reply

Table 14: Possible values for the network and pbxtype tags

eDMSAPI_PBX_type_str(eDMSAPI	<pbxtype>	<network>
DAP controller	DAP Controller	dasgif

DMC	DMC	ETHERNET_DMC
AVAYA	AVAYA	Dasgif

Refer to the chapters of this document that deal with the database tables for more information on each value. A detailed description of these internal inter-process communications is beyond the scope of this document.

The parameter <IoReg_cnt> specifies how much DECT extension is IoRegistered (can send data messages to the DECT Messenger application).

The parameter <IoReg_xxxx> (where xxxx starts with 0001 until IoReg_cnt) specifies the DECT extensions that must be IoRegistered.

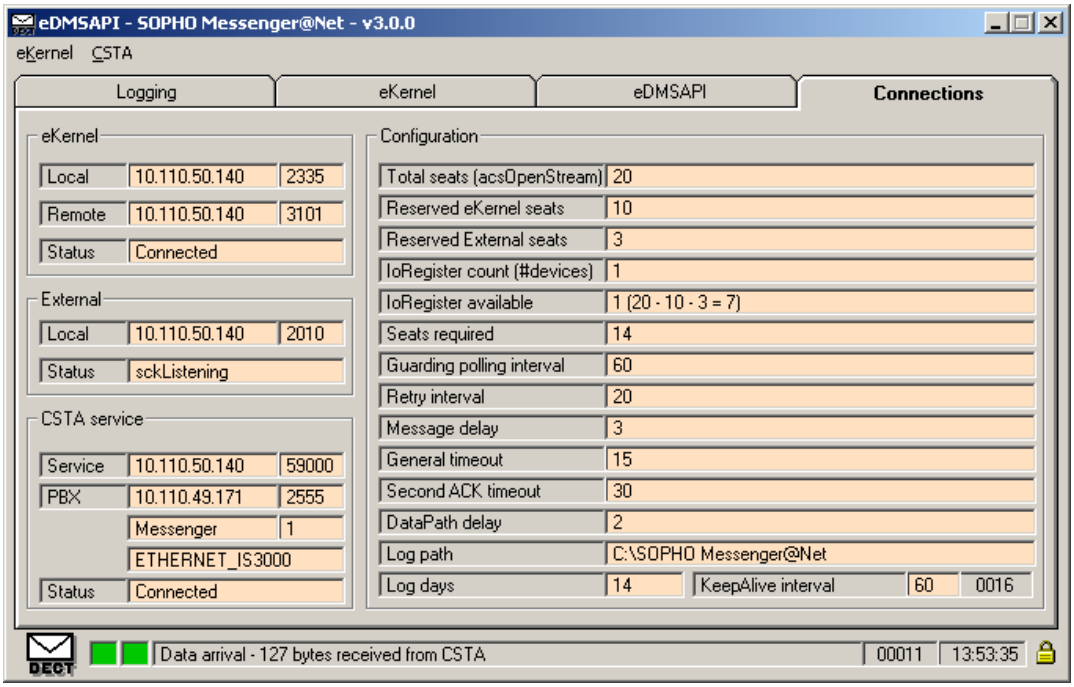


Figure 84: eDMSAPI Connections

In the left panel of the window shown in Figure 84: eDMSAPI Connections on page 137, the configuration and state of the different socket connections is shown.

On the right side, the configuration parameters received (<cfgrpy>) from the eKernel are shown.

Note:

When the eDMSAPI functionality is not licensed, the eKernel sends the following configuration reply: <xml><cfgrpy><license>NO LICENSE AVAILABLE</license></cfgrpy></xml>. After sending this reply to eDMSAPI, the port through which the eDMSAPI is communicating is closed on the eKernel side.

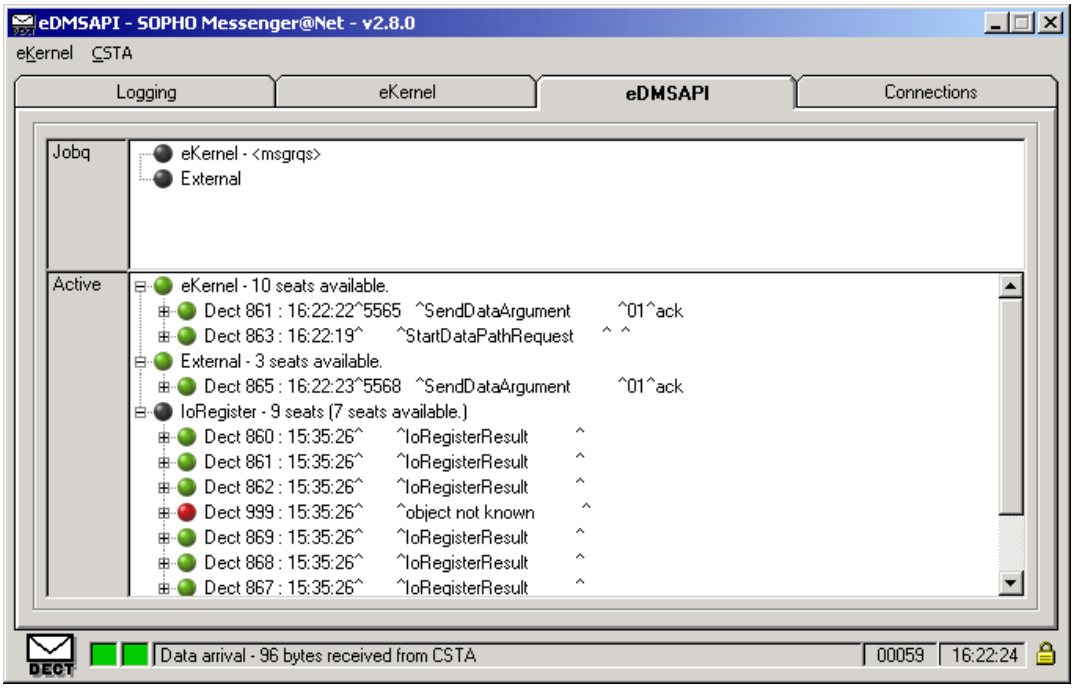


Figure 85: eDMSAPI tab

Note that during call handling, the eDMSAPI tab shows an overview of current call states of each device. The window consists of two sections:

- The Jobq section contains two job queues. One jobq is for the request from eKernel, and the other is for requests from external clients.

  This area is used to temporarily store requests that are waiting to be executed. For instance, when all the data paths are in use, new requests must wait until resources are available.

  Requests can come from the eKernel (<msgrqs>) or from an external client (SNDNMSG|ID|DNR|MESSAGE<cr><lf> or SNDUMSG|ID|DNR|MESSAGE<cr><lf>)

  The functionality to receive requests from external clients is supported only for the DECT Messenger application (internal use only).

- The Active section contains active jobs. This area is used to handle currently active requests. There are three different sections in this area: the eKernel, the External and the IoRegister.

  The eKERNEL and External areas show the active extensions. Active requests wait for acknowledge from the CSTA Service. A normal message receives an ACK or NAK (StopDataPathRequest) reply. An urgent message receives an initial ACK or NAK. Urgent messages that receive ACK wait <Ack2TimeOut> seconds for a second ACK or a NAK.

Overview of CSTA_Service.EXE

The CSTA_Service runs behind the scenes and communicates through CSTA.DLL with a PBX. The CSTA_Service acts as TCP server towards eCSTA. Communications to a PBX is done through CSTA.DLL, based upon Ethernet iS-Link CSTA interface on Ethernet. Details on these communications are beyond the scope of this document.

The CSTA_Service has no user interface, but an icon appears in the tray when it is running. When you right-click on the icon, a pop-up windows appears and shows the following options.

- About CSTA Service shows a banner with a copyright information panel similar to the one shown in the next figure.

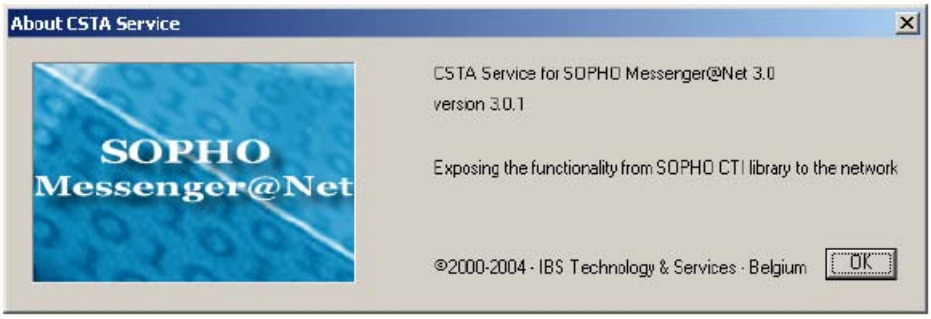


Figure 86: About CSTA service

- You can use Kill all clients to disconnect all TCP connections, for example the TCP/IP connection to eCSTA. Do not perform this function unless instructed to do so by service support.
- You can use End CSTA Service to close the CSTA_Service program.

Logging

The eDMSAPI application provides logging to both the window and to logging files on disk.

You can view the on-screen log through the Logging tab:

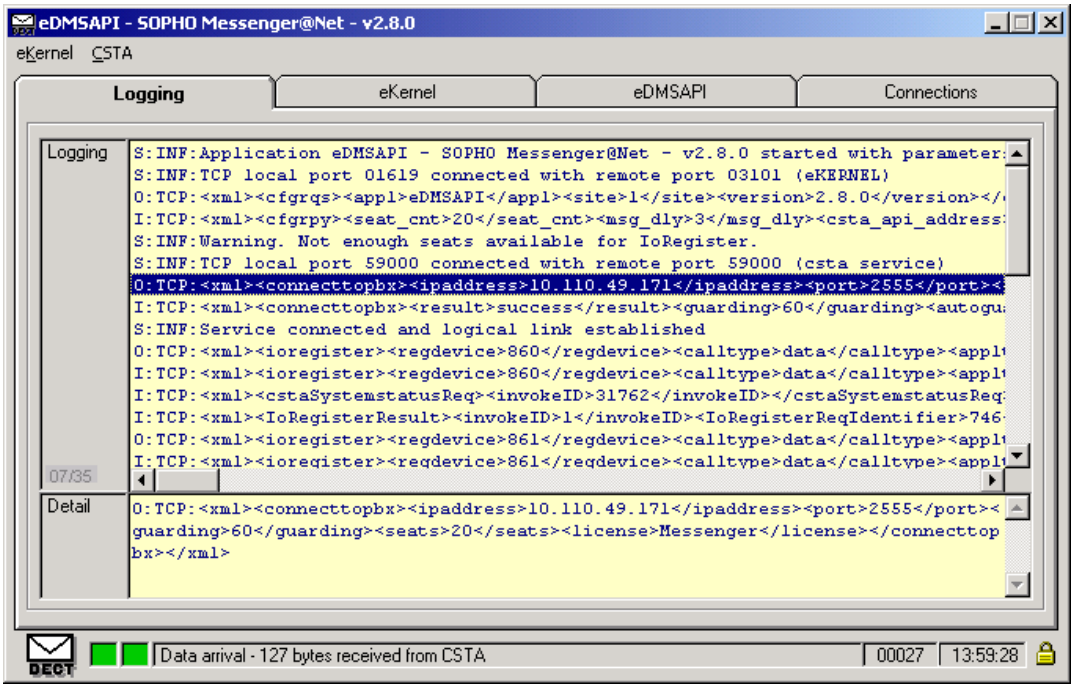


Figure 87: Logging information

Sample logging data is shown in Figure 88: Log example: initialization procedure on page 141.

```
03/10/2002 13:58:52 - S:INF:Application eDMSAPI - SOPHO Messenger@Net -
v2.8.0 started with parameters /Site:1 /eKernel address:*LOCAL /eKernel
port:3101 /Log drive:C

03/10/2002 13:58:53 - S:INF:TCP local port 01619 connected with remote
port 03101 (eKERNEL)

03/10/2002 13:58:53 - O:TCP:<xml><cfgrqs><appl>eDMSAPI</appl><site>1</
site><version>2.8.0</version></cfgrqs></xml>

PBX type:DMC
28/06/2004 13:58:55 - I:TCP:<xml><cfgrpy><seat_cnt>20</
seat_cnt><msg_dly>3</msg_dly><csta_api_address>10.110.50.140</
csta_api_address><csta_api_port>59000</
csta_api_port><external_seat_cnt>3</
external_seat_cnt><external_port>2010</
external_port><csta_pbx_address>10.110.49.171</
csta_pbx_address><csta_pbx_port>2555</csta_pbx_port><csta_licence>Mes-
senger</csta_licence><guarding_intv>60</
guarding_intv><guarding_Retry_intv>20</
guarding_Retry_intv><eKernel_Seat_cnt>10</eKernel_Seat_cnt><GeneralTim-
eOut>15</GeneralTimeOut><Ack2TimeOut>30</Ack2TimeOut><DataPathDelay>2</
DataPathDelay><network>ETHERNET_DMC</network><pbxtype>DMC</pbx-
type><IoReg_cnt>9</IoReg_cnt><IoReg_0001>860</
IoReg_0001><IoReg_0002>861</IoReg_0002><IoReg_0003>862</
IoReg_0003><IoReg_0004>999</IoReg_0004><IoReg_0005>869</
IoReg_0005><IoReg_0006>868</IoReg_0006><IoReg_0007>867</
IoReg_0007><IoReg_0008>866</IoReg_0008><IoReg_0009>865</
IoReg_0009><log_path>C:\SOPHO Messenger@net</log_path><log_days>14</
log_days></cfgrpy></xml>

PBX type:DAP controller
28/06/2004 13:58:55 - I:TCP:<xml><cfgrpy><seat_cnt>20</
seat_cnt><msg_dly>3</msg_dly><csta_api_address>10.110.50.140</
csta_api_address><csta_api_port>59000</
csta_api_port><external_seat_cnt>3</
external_seat_cnt><external_port>2010</
external_port><csta_pbx_address>10.110.49.171</

continued on next page...
```

Figure 88: Log example: initialization procedure

```
csta_pbx_address><csta_pbx_port>28001</csta_pbx_port><csta_licence>Mes-
senger</csta_licence><guarding_intv>60</
guarding_intv><guarding_Retry_intv>20</
guarding_Retry_intv><eKernel_Seat_cnt>10</eKernel_Seat_cnt><GeneralTim-
eOut>15</GeneralTimeOut><Ack2TimeOut>30</Ack2TimeOut><DataPathDelay>2</
DataPathDelay><network> dasgif</network><pbxtype> DAP Controller</pbx-
type><IoReg_cnt>9</IoReg_cnt><IoReg_0001>860</
IoReg_0001><IoReg_0002>861</IoReg_0002><IoReg_0003>862</
IoReg_0003><IoReg_0004>999</IoReg_0004><IoReg_0005>869</
IoReg_0005><IoReg_0006>868</IoReg_0006><IoReg_0007>867</
IoReg_0007><IoReg_0008>866</IoReg_0008><IoReg_0009>865</
IoReg_0009><log_path>C:\SOPHO Messenger@net</log_path><log_days>14</
log_days></cfgrpy></xml>

PBX type: CS 1000
28/06/2004 13:58:55 - I:TCP:<xml><cfgrpy><seat_cnt>20</
seat_cnt><msg_dly>3</msg_dly><csta_api_address>10.110.50.140</
csta_api_address><csta_api_port>59000</
csta_api_port><external_seat_cnt>3</
external_seat_cnt><external_port>2010</
external_port><csta_pbx_address>10.110.49.171</
csta_pbx_address><csta_pbx_port>28001</csta_pbx_port><csta_licence>Mes-
senger</csta_licence><guarding_intv>60</
guarding_intv><guarding_Retry_intv>20</
guarding_Retry_intv><eKernel_Seat_cnt>10</eKernel_Seat_cnt><GeneralTim-
eOut>15</GeneralTimeOut><Ack2TimeOut>30</Ack2TimeOut><DataPathDelay>2</
DataPathDelay><network> dasgif</network><pbxtype>NORTEL</pbx-
type><IoReg_cnt>9</IoReg_cnt><IoReg_0001>860</
IoReg_0001><IoReg_0002>861</IoReg_0002><IoReg_0003>862</
IoReg_0003><IoReg_0004>999</IoReg_0004><IoReg_0005>869</
IoReg_0005><IoReg_0006>868</IoReg_0006><IoReg_0007>867</
IoReg_0007><IoReg_0008>866</IoReg_0008><IoReg_0009>865</
IoReg_0009><log_path>C:\SOPHO Messenger@net</log_path><log_days>14</
log_days></cfgrpy></xml>


03/10/2002 13:58:55 - S:INF:Warning. Not enough seats available for
IoRegister.
```

continued on next page...

```
03/10/2002 13:59:00 - S:INF:TCP local port 59000 connected with remote
port 59000 (csta service)

PBX type:DMC
03/10/2002 13:59:00 - O:TCP:<xml><connecttopbx><ipad-
dress>10.110.49.171</ipaddress><port>2555</port><guarding>60</guard-
ing><seats>20</seats><licence>Messenger</licence>
<network>ETHERNET_DMC</network></connecttopbx></xml>

PBX type:DAP controller
03/10/2002 13:59:00 - O:TCP:<xml><connecttopbx><ipad-
dress>10.110.49.171</ipaddress><port>28001</port><guarding>60</guard-
ing><seats>20</seats><licence>Messenger</licence><network>dasgif</
network></connecttopbx></xml>

PBX type:CS 1000
03/10/2002 13:59:00 - O:TCP:<xml><connecttopbx><ipad-
dress>10.110.49.171</ipaddress><port>28001</port><guarding>60</guard-
ing><seats>20</seats><licence>Messenger</licence><network>dasgif</
network></connecttopbx></xml>


03/10/2002 13:59:00 - I:TCP:<xml><connecttopbx><result>success</re-
sult><guarding>60</guarding><autoguarding>1</autoguarding></connecttop-
bx></xml>

03/10/2002 13:59:00 - S:INF:Service connected and logical link estab-
lished

03/10/2002 13:59:00 - O:TCP:<xml><ioregister><regdevice>860</regde-
vice><calltype>data</calltype><appltype>messaging</appltype></ioregis-
ter></xml>

03/10/2002 13:59:00 - I:TCP:<xml><IoRegisterResult><invokeID>1</in-
vokeID><IoRegisterReqIdentifier>746</IoRegisterReqIdentifier></IORegis-
terResult></xml>
03/10/2002 13:59:00 - I:TCP:<xml><cstaSystemstatusReq><invokeID>31762</
invokeID></cstaSystemstatusReq></xml> (Guarding)
>
</xml>
```

```
03/10/2002 12:08:57 - I:TCP:<xml><msgrqs><id>00774</id> <ext>861</ext>
<ext_prty>6</ext_prty><pag_01>Guarding AM TELEVIC             </
pag_01><prty_01>N</prty_01><pag_more>N</pag_more><format>16^16^0^5^2</
format>
</msgrqs></xml>

PBX type: DMC and DAP controller: NORMAL and/or URGENT messages
03/10/2002 12:08:57 - O:TCP:<xml><startdatapathdevice><deviceID>861</de-
viceID><pathtype>text</pathtype>
<dirtype>bi</dirtype><homelocationnumber>1<homelocationnumber> </start-
datapathdevice></xml>

PBX type: CS 1000: NORMAL and/or URGENT messages
03/10/2002 12:08:57 - O:TCP:<xml><startdatapathdevice><deviceID>861</de-
viceID><pathtype>text</pathtype>
<dirtype>bi</dirtype><homelocationnumber>0<homelocationnumber> </start-
datapathdevice></xml>

PBX type: DMC and DAP controller: also Emergency messages
03/10/2002 12:08:57 - O:TCP:<xml><startdatapathdevice><deviceID>861</de-
viceID><pathtype>text</pathtype>
<dirtype>bi</dirtype><callcategory>emergency</callcategory> <homeloca-
tionnumber>1<homelocationnumber></startdatapathdevice></xml>

PBX type: CS 1000: Emergency messages
03/10/2002 12:08:57 - O:TCP:<xml><startdatapathdevice><deviceID>861</de-
viceID><pathtype>text</pathtype>
<dirtype>bi</dirtype><callcategory>emergency</callcategory><homeloca-
tionnumber> 0<homelocationnumber></startdatapathdevice></xml>

03/10/2002 12:08:58 - I:TCP:<xml><StartDataPathResult><IoCrossRefIdenti-
fier>5697</IoCrossRefIdentifier>
<invokeID>58</invokeID></StartDataPathResult></xml>

03/10/2002 12:08:58 - O:TCP:<xml><senddata><iocrossrefid>5697</iocross-
refid>
<Text>Guarding AM TELEVIC            01/01</Text><originator>pbx</orig-
inator>

continued on next page...
```

Figure 89: Log example: Message handling

```
<msgtype>normal</msgtype><direction>outbound</direction></senddata></
xml>

03/10/2002 12:08:58 - I:TCP:<xml><SendDataResult><invokeID>59</in-
vokeID></SendDataResult></xml>

03/10/2002 12:09:00 - I:TCP:<xml><SendDataArgument><invokeID>31206</in-
vokeID>
<ioCrossRefIdentifier>5697</ioCrossRefIdentifier><Provider>1</Provider>
<Text><ack></Text></SendDataArgument></xml>

03/10/2002 12:09:00 - O:TCP:<xml><senddataresult><invokeID>31206</in-
vokeID></senddataresult></xml>

03/10/2002 12:09:00 - O:TCP:<xml><stopdatapath><iocrossrefid>5697</
iocrossrefid>
<originator>pbx</originator></stopdatapath></xml>

03/10/2002 12:09:00 - I:TCP:<xml><stopdatapath><iocrossrefid>5697</
iocrossrefid>
<originator>pbx</originator></stopdatapath><invokeID>61</invokeID></
xml>

03/10/2002 12:09:06 - I:TCP:<xml><StopDataPathResult><InvokeID>61</In-
vokeID>nodata
</StopDataPathResult></xml>
```

Chapter 18: Module - eFR

Important:

Due to the ongoing development of the DECT Messenger product suite, some modules that provide additional functionality may become available after the initial release of DECT Messenger 4.0.

The following modules are described in this document but are not available at initial General Availability.

- eFR
- eLICENSE
- eLOCATION
- eSMS
- eSNMP
- eVBVOICE

The eFR module is an add-on module and is licensed separately through the eLICENSE module. Some of the modules listed in this attention box are available only on a site-specific basis.

Introduction

Module eFR is a Windows-based software module that monitors a number of items.

The letters FR in Module eFR stand for fault reporting. The eFR module can run standalone or can run in an environment powered by DECT Messenger.

When DECT Messenger infrastructure is available, additional functionality becomes available.

Basic overview

The eFR module consists of two major parts: monitoring and notification.

Overview of monitoring

The monitoring section covers four items.

- OM section

  The OM section is not applicable for traditional and SIP DECT systems.

- DISK section

  The DISK section verifies the state of a hard disk drive and notifies you on:

  - whether the disk drive is ready or not,
  - how much disk space is available.

    The status information is compared with a previous run for the same drive. You are notified if different information is found. You are notified when the drive is not ready and you are also notified when the drive is ready.

- PING section

  The PING section verifies if the PING command is successful in finding a destination address. For the destination addresses, you can enter:

  - The loopback adapter addresses or local IP addresses to verify local TCP stack and network resources.
  - A remote IP addresses to ensure that the network is responding on ICMP level.

    The status is compared with a previous run for the same destination. You are notified if different information is found. You are also notified when the PING is not responding. You are notified when the PING reestablishes a response.

- NETSTAT section

  The NETSTAT section verifies if a TCP Server is in a "listening" state, by determining whether a defined port has a socket in a "listening" state. For example, when an SMTP server is present "listening" on socket 25, it can be verified if a socket is "listening" for inbound connections or not.

  Similar tests are possible for all kinds of TCP Servers on well known ports, such as Telnet 23, HTTP 80, HTTPS 443, and so on, or user defined ports. It is also possible to verify that a TCP connection is established on a predefined port.

  Typical usage verifies that a connection that needs to be established continuously is available. The status is compared with a previous run for the same port. You are notified if

different information is found. You are notified when the socket is no longer "listening" or established, and you are notified when the socket returns "listening" or established.

Overview of notification

The notification part covers the following items.

- The SMTP destinations refer to the notification based upon e-mail by means of SMTP protocol. SMTP destinations requires an e-mail server infrastructure that is configured to support inbound SMTP connections originated from the eFR module. Depending on the e-mail infrastructure, you may need to configure the e-mail server to enable SMTP, allow inbound SMTP connections from the IP address where eFR runs, and allow relaying of e-mail request to defined destinations.

  You can configure the SMTP notification to send notification messages when changes occur in the state of monitoring items, such as DISK, NETSTAT, PING, start of error condition, and resume of normal condition.

- The SNMP destinations refer to notification based upon the SNMP protocol. In SNMP destinations, SNMPv1 traps can be sent based upon the UDP protocol stack. Since UDP protocol is datagram oriented, SNMPv1 traps can be sent even if the SNMP receiver infrastructure is not in place on the destination system. When the SNMP receiver infrastructure is not in place on the destination system, the trap is unprocessed. The intended use is oriented towards an infrastructure capable of receiving SNMPv1 traps and performing further actions.

  - The DECT Messenger suite features a module eSNMP that is optimized for this functionality. DECT Messenger can receive the SNMPv1 trap and reliably process it, through several notification channels, including DECT Messaging, SMS messaging, Interactive Voice Response, discrete contacts, e-mail, and so on. In addition, the software allows detecting if message reaches its destination and optionally needs end-user confirmation. DECT Messenger can retry and escalate to alternative destinations.

- The SMS destinations refer to notification based upon SMS messaging to mobile phones.

  - SMS destinations requires at least the SMS_service engine and a supported GSM box with a SIM card. You can add SMS destinations with such an infrastructure to run in stand-alone mode, or as part of an existing DECT Messenger infrastructure.

- NET destinations refer to the notification based upon named pipes transport towards the messenger service that is part of the Windows operating system.

  - The functionality of NET destinations is similar to the NET SEND command found in the Windows operating system. NET destinations allows you to send a Windows pop-up message to a Windows PC. The depending messenger service must be enabled

and active on the destination system, and all prerequisite conditions for named pipes messaging applies.

Install module eFR

Follow the steps in the next procedure to install the software.

Installing module eFR

1. Double-click the EXE files delivered by Avaya.
2. Accept all defaults to install the software.

   After installation, a number of items ARE installed.

```
C:\Program Files\Messenger@Net\Exe\eFR.exe
C:\Program Files\Messenger@Net\Exe\eFR.xml
C:\Program Files\Messenger@Net\Mdb\eFR.mdb
C:\Program Files\Messenger@Net\Exe\MSWINSCK.OCX
C:\Program Files\Messenger@Net\Exe\MSWINSCK.OCX
C:\Program Files\Messenger@Net\Exe\TABCTL32.OCX
C:\Program Files\Messenger@Net\Exe\MSCOMCTL.OCX
C:\Program Files\Messenger@Net\Pdf\Module_eFR.pdf
C:\Program Files\Messenger@Net\Reg\readme.txt
C:\Program Files\Messenger@Net\Reg\eFR.reg
```

Launch module eFR

Follow the steps in the next procedure to launch module eFR.

The eFR module is implemented as a application and not as a service.

Launching module eFR

1. Logon to your PC.

   Implementation as an application means you launch the EXE file after logon on the PC.
2. Create a shortcut to the file eFR.exe, for example

   C:\Program Files\Messenger@Net\Exe\eFR.exe
3. Drag the shortcut to the Startup menu of the user that logs on to the computer.
4. Configure an automatic logon (recommended).

   Your ability to configure an automatic logon depends on your security policy. When you have automatic logon, after a power failure the PC restarts and logs on automatically.

Do not logoff the desktop as this ends the monitoring application.

License module eFR

The module eFR is a licensed software module. The module eFR cannot be copied without valid agreement from Avaya. A software key is required to launch the program. If you do not have a software license key, the following window appears.

Figure 90: Software license key required

Contact Avaya to obtain an evaluation or a permanent key for using the software.

Configure module eFR

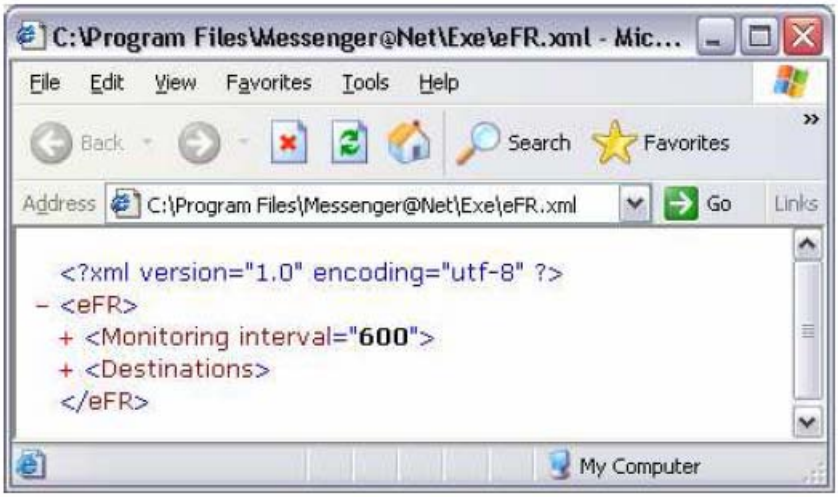
The module eFR is configured using an XML file.

Follow the steps in the next procedure to configure module eFR.

Configuring module eFR

1. Browse to the directory C:\Program Files\Messenger@Net\Exe.
2. Double-click on the file eFR.xml.

You can use Microsoft Internet Explorer to access files with XML extensions. These XML files must have the correct syntax or XML cannot be opened by Internet Explorer and results in a failure of module eFR.



Destinations

Destinations planning

Follow the steps in the next procedure for destinations planning.

Planning destinations

1. Determine what notification methods to use.

   Notification methods include technologies such as the following.

   - e-mail
   - windows popup
   - SMS message to mobile GSM phone
   - SNMPv1 trap

2. Check what prerequisite actions and infrastructure are needed to make the destinations operational.

   Consultant services are available to assist in this process.

   For some transport mechanisms you can depend on other parties and people, for example the administrator of the e-mail infrastructure.

Other transport mechanisms can require you to perform actions in the destination site, for example for NET SEND, the messenger service needs to be activated in the target PC.

3. Gather a list of destination users or destination peripherals with specific information to uniquely address the destination device.

   For example, for e-mail users you need the e-mail address of the target user and for SMS message, you need the extension number of the mobile GSM phone.

Destinations configuration

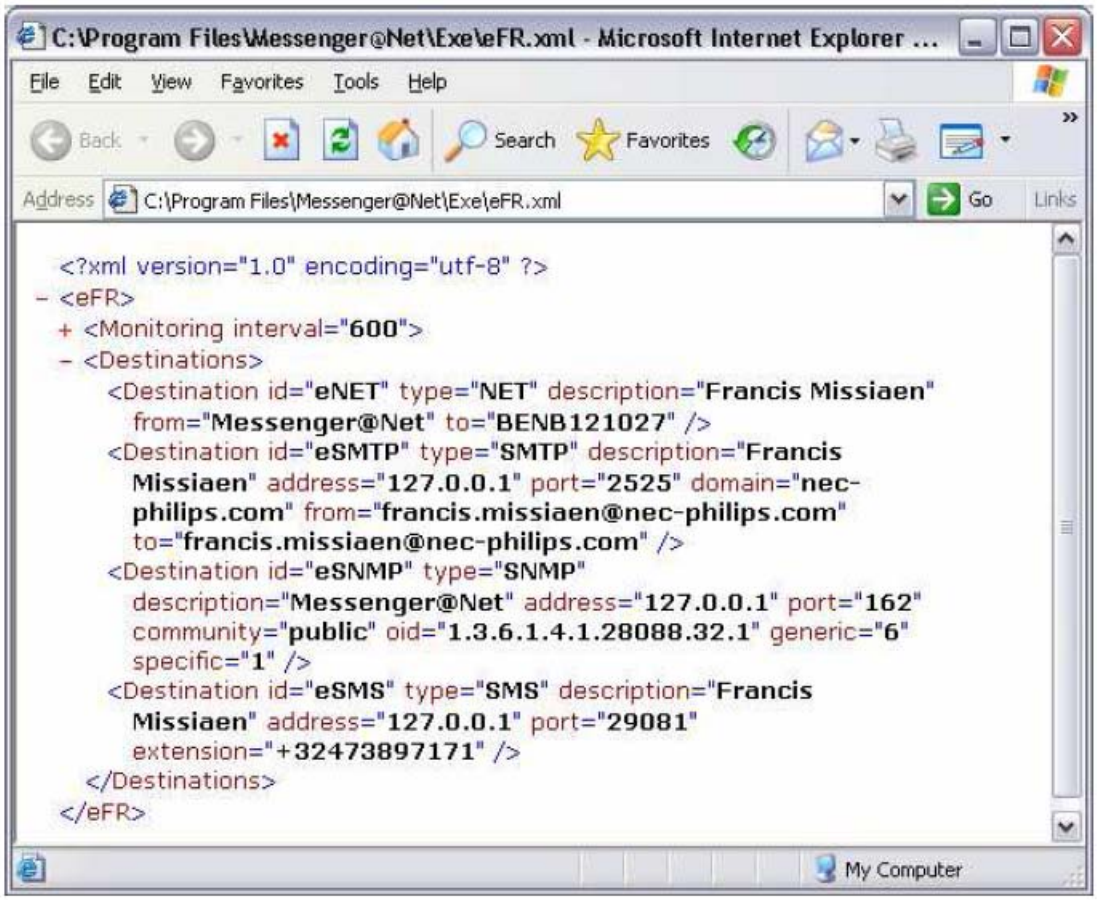
Follow the steps in the next procedure to configure destinations.

Configuring destinations

1. Define the destinations in the Destinations section of the eFR.xml file.

   The destinations section starts with the tag <Destinations> and ends with the closing tag </Destinations>.

   The following figure shows sample destinations.



You can use the windows delivered with Notepad or Wordpad accessories maintain the XML file.

On the internet, a number of free XML editors are available. These tools typically feature syntax checking, colored editing, and advanced editing features including copy and pasting of nodes. You can download the file XmlNotepad.msi (XML Notepad 2007) on the Microsoft web site.

2. Ensure that every destination has a unique value for the parameter id.

   In the sample file in the previous figure, the id values are eNET, eSMTP, eSNMP, eSMS, and so on.

   The parameter id must be unique.

   You can use identifiers that are more meaningful to you, such as your name, for example, Francis Missiaen – e-mail or Francis Missiaen – sms.

3. Ensure that each destination has a unique value for the parameter type.

   The current release supports the following values for parameter types,

   - NET NET is used with NET SEND technology when you generate a Windows pop-up message on a destination PC with a Windows operating system.

- SMS SMS is used with SMS technology to send a SMS message to a destination mobile GSM phone.
- SMTP SMTP is used when sending an e-mail message by means of an SMTP enabled mail server infrastructure, resulting in delivery of an e-mail message to the inbox of the destination addressee.
- SNMP SNMP is used when sending SNMPv1 trap to a destination SNMP trap receiver.

Depending of the selected notification type, define additional parameters as needed. Typically, additional parameters define the remaining values that are specific to the selected transport mechanism, and contain information to identity the destination user or device, as well as parameters that define the intermediate infrastructure.

The following Destination types are described in this chapter.

- Destinations type NET
- Destinations type SMS
- Destinations type SMTP
- Destinations type SNMP

Destinations type NET

The following figure illustrates Destinations type NET.

```
<Destination id="eNET" type="NET" description="Francis Missiaen"
from="Messenger@Net" to="BENB121027" />
```

Figure 91: Destinations type NET

The following parameters are specific to NET.

- The parameter description refers to the name of the destination user, and allows you to assign a user name. For example, you can enter the first name and last name.
- The parameter "from" is part of the resulting Windows pop-up message and is shown to the user. The parameter "from" allows the destination user to identify what user or application has sent the pop-up message. For example, you can use DECT Messenger.
- The parameter "to" is the most important parameter as this parameter defines the destination system used by the embedded NET SEND functionality. The parameter "to" must be the correct PC name of the destination user.

  As a suggestion, open the command window on the target system and use the echo %computername% command to find out the value that needs to be specified in the field "to". For example, In the figure, the resulting value is BENB121027.

Figure 92: Parameter to

Destinations type SMS

The following figure illustrates Destinations type SMS.

```
<Destination id="eSMS" type="SMS" description="Francis Missiaen"
address="127.0.0.1" port="29081" extension="+32473897171" />
```

Figure 93: Destinations type SMS

The following parameters are specific to SMS.

- The parameter "description" refers to the name of the destination user, and allows you to assign a user name. For example, you can enter the first name and last name of the user.
- The parameter "address" refers to the IP address of the system that runs the SMS_service process. When the service runs on the same system, use the loopback value 127.0.0.1.
- The parameter "port" refers to the port number that is configured in the SMS_service process for inbound SMS request.

  The default value 29081 matches most installations. Refer to the SMS_service system administrator.
- The parameter "extension" is the most important parameter, as "extension" defines the destination peripheral of the SMS notification. Format "extension" starting with + and country code, which results in a valid mobile GSM phone. For example, in Belgium a valid mobile extension number is formed starts with +32 followed by the remaining numbers of the mobile GSM phone.

Destinations type SMTP

The following figure illustrates Destinations type SMTP.

```
<Destination id="eSMTP" type="SMTP" description="Francis Missiaen"
address="127.0.0.1" port="25" domain="nec-philips.com"
from="francis.missiaen@nec-philips.com" to="francis.missiaen@necphilips.
com" />
```

Figure 94: Destinations type SMTP

The following parameters are specific to SMTP.

- The parameter "description" refers to the name of the destination user, and allows you to assign a user name. For example, enter the first and last name of the user.
- The parameter "address" refers to the e-mail system IP address that runs the SMTP Server process. When the e-mail server supporting SMTP runs on the same system, use the loopback value 127.0.0.1. Contact your e-mail server administrator to obtain this value.
- The parameter "port" refers to the port number configured in the SMTP server for inbound SMTP requests. The default value 25 matches most installations. Contact the e-mail server system administrator to obtain this value.
- The parameter "domain" refers to the e-mail domain name configured on the SMTP server. In most cases this matches the characters after the @ of the company local e-mail addresses. Contact the e-mail server administrator to obtain this value.
- The parameter "from" refers to an e-mail address that acts as originator of the e-mail address. In many cases, the syntax matches user@domain.com. In some environments, the domain must match the existing domain or be a registered domain. Contact the e-mail server administrator to obtain this value. Avaya recommends that you use an existing e-mail address so people can respond to the e-mail if necessary.
- The parameter "to" is the most important parameter and refers to the e-mail address of the destination e-mail user. Depending on relay settings of the e-mail server infrastructure, this can be limited to an internal e-mail user only, which limits you to send e-mail only to users of the same domain. The parameter "to" can also be an external e-mail user, which allows you to send e-mail to users of any domain. Contact the e-mail infrastructure administrator to discuss the parameters and to determine if an adjustment of the e-mail server is necessary.

Destinations type SNMP

The following figure illustrates Destinations type SNMP.

```
<Destination id="eSNMP" type="SNMP" description="Messenger@Net"
address="127.0.0.1" port="162" community="public"
oid="1.3.6.1.4.1.28088.32.1" generic="6" specific="1" />
```

Figure 95: Destinations type SNMP

The following parameters are specific to SNMP.

- The parameter "description" refers to the name of the destination party, and allows you to assign a destination system name. For example, you can enter the name of the SNMP receiving infrastructure.
- The parameter "address" refers to the SNMP server IP address. Contact the SNMP receiving infrastructure administrator to obtain this value.
- The parameter "port" refers to the SNMP server port number. The default value usually used is the well-known port number 162. Contact the SNMP receiving infrastructure administrator to obtain this value.
- The parameter "community" identifies the community, for example public.
- The parameter "oid" identifies the enterprise OID, for example 1.3.6.1.4.1.28088.32.1.
- The parameter "generic" identifies the generic trap identifier, for example 6.
- The parameter "specific" identifies the specific trap identifier, for example 1 TIP.

Contact the administrator of the SNMP trap receiver infrastructure. In many cases this is the DECT Messenger administrator when the eSNMP module is used to process received SNMPv1 traps.

Monitoring

The monitoring section is defined in the eFR.xml file.

The monitoring section starts with the tag </Monitoring>. Within the tag are the following subsections.

- OM section is not applicable for traditional and SIP DECT systems
- DISK is available starting with <DISK> and ending with </DISK>
- PING is available starting with <PING> and ending with </PING>
- NETSTAT is available starting with <NETSTAT> and ending with </NETSTAT>

The following figure illustrates the monitoring section and subsections.

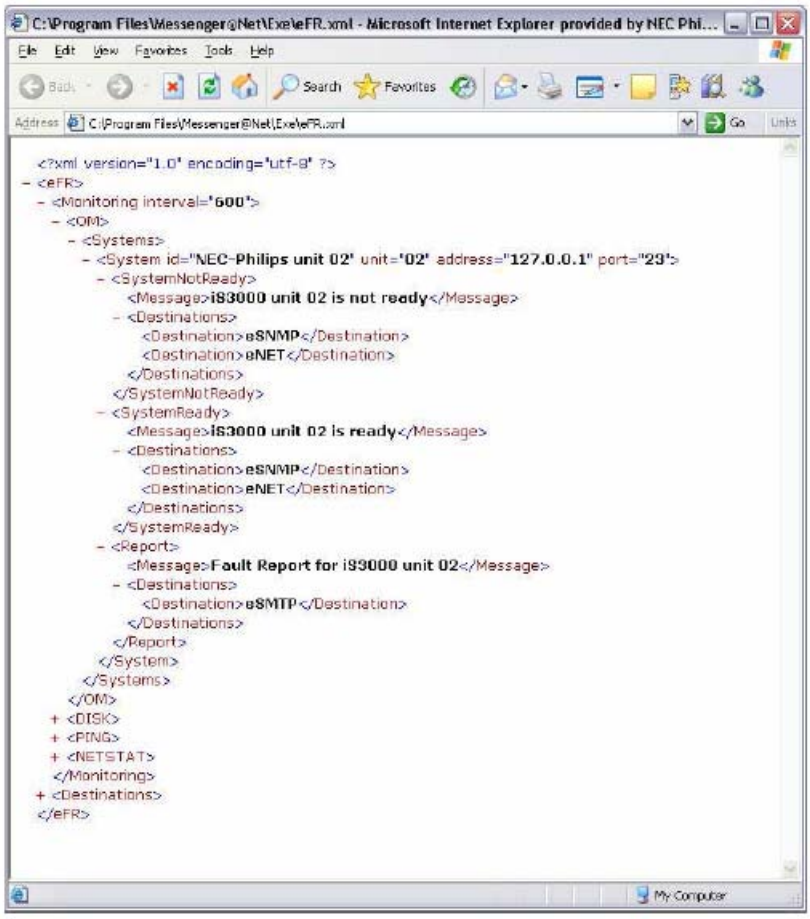


Figure 96: Example of monitoring section and subsections

The parameter interval, with a default value of 600, specifies the frequency at which the monitoring process takes place. When a value of 600 is specified, a delay of 600 seconds takes place between the last monitoring action and the next monitoring action. The eFR module captures information about the configured items roughly every 10 minutes.

- The default value is 600 seconds
- The minimum value is 60 seconds. Every minute a verification cycle takes place. When a value smaller than 60 is configured, the system adjusts it to 60 seconds
- The maximum value is 86400 seconds, so once a day a verification cycle takes place. When a larger value than 86400 is configured, the system will adjust to 86400 seconds.

Note:

The processing cycle takes some time, for example PING to a large number of systems that are not responding increases processing time. Since the interval effects the time between ending the previous verification and starting a new verification, the exact frequency is somewhat larger than the specified interval, depending on the processing time.

Monitoring type DISK

The monitoring of disk drives is configured through the section in the XML file shown in the next figure.

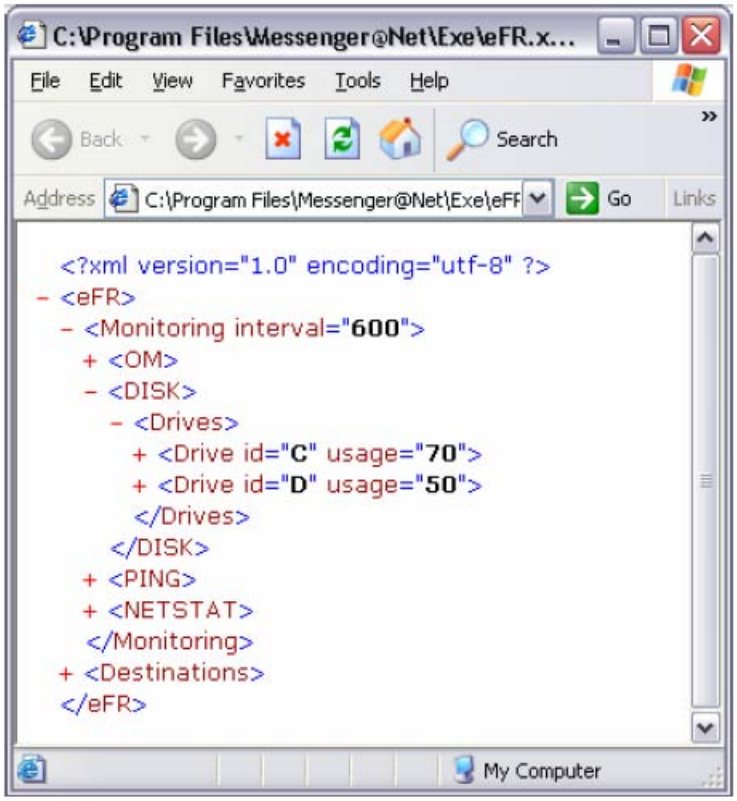


Figure 97: Monitoring type DISK

The opening tag <DISK> and the closing tag </DISK> provide space for configuring the drives that must be monitored. You can define each drive that needs to be monitored in the section between </Drives>.

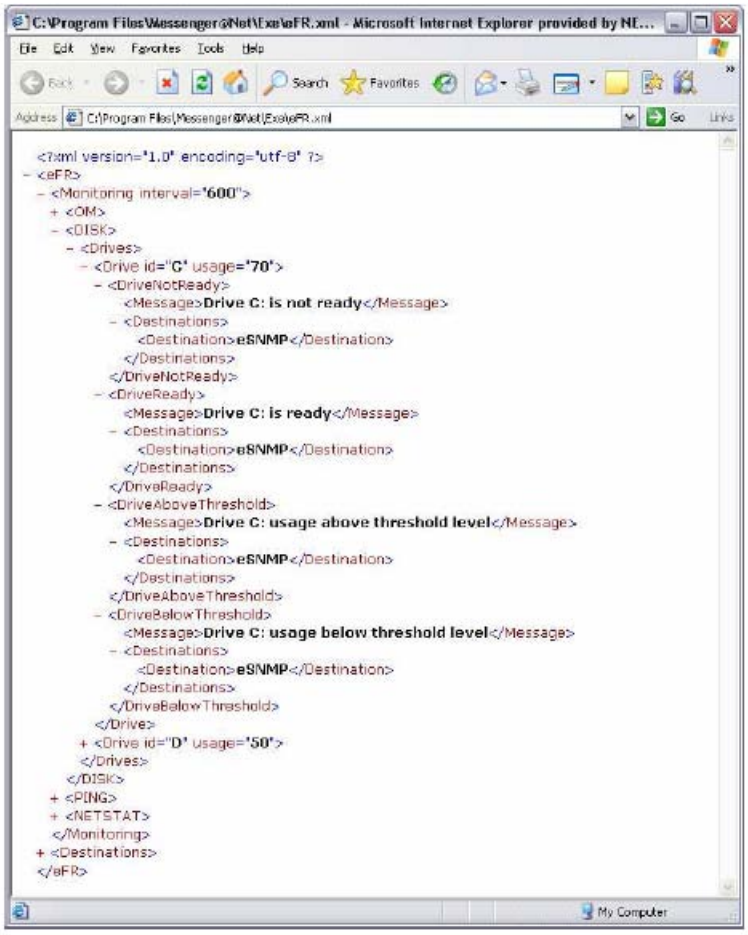


Figure 98: Define drives for monitor type DISK

The line shown in the following figure is the opening tag for a drive definition.

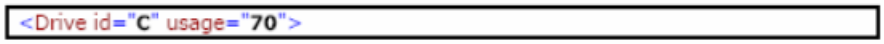


Figure 99: Opening tag for a drive definition

You must define two parameters.

- Create the parameter "id" as a single upper case character that represents the drive name to be monitored. For example, entering C as the "id" indicates that the C: drive is monitored. This character is typically the drive letter you assign to the hard disk drive.
- Create the parameter "usage" as a numeric value that specifies a percentage of allowed disk usage. For example, enter 70 to indicate that up to 70% of the hard disk can be used. When less than 30% free capacity is available, you must notify users.

The window in the following figure shows a green bullet next to the C drive to indicate that the required conditions are met. The window shows a red bullet next to the D drive to indicate that for that drive, the conditions are not met. D drive is not ready or has reached the defined usage.

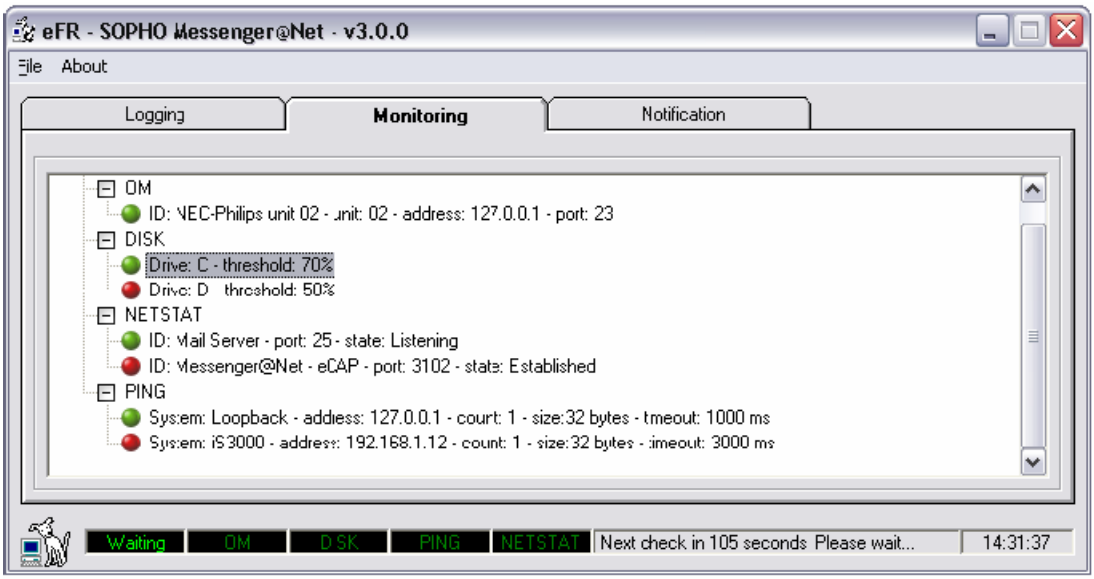


Figure 100: Required conditions met for monitor type DISK

You define the message and the destinations to be informed in the following circumstances.

- When the drive is not ready

```
<DriveNotReady>
```

Figure 101: Drive not ready

- When the drive is ready again

```
<DriveReady>
```

Figure 102: Drive ready

- When the drive is above threshold usage level

```
<DriveAboveThreshold>
```

Figure 103: Drive above threshold usage level

- When the drive is below threshold usage level

```
<DriveBelowThreshold>
```

Figure 104: Drive below threshold usage level

- When the D drive is not ready to be sent as SNMPv1 trap to an SNMP server

```
<Destination id="eSNMP" type="SNMP" description="Messenger@Net"
address="127.0.0.1" port="162" community="public"
oid="1.3.6.1.4.1.28088.32.1" generic="6" specific="1" />
```

Figure 105: Drive not ready to be sent as SNMPv1 trap

The following figure shows the message received by the destination device, implemented in DECT Messenger module eSNMP. The window shows that an SNMPv1 trap is received. The varbind 1 parameter contains the message.

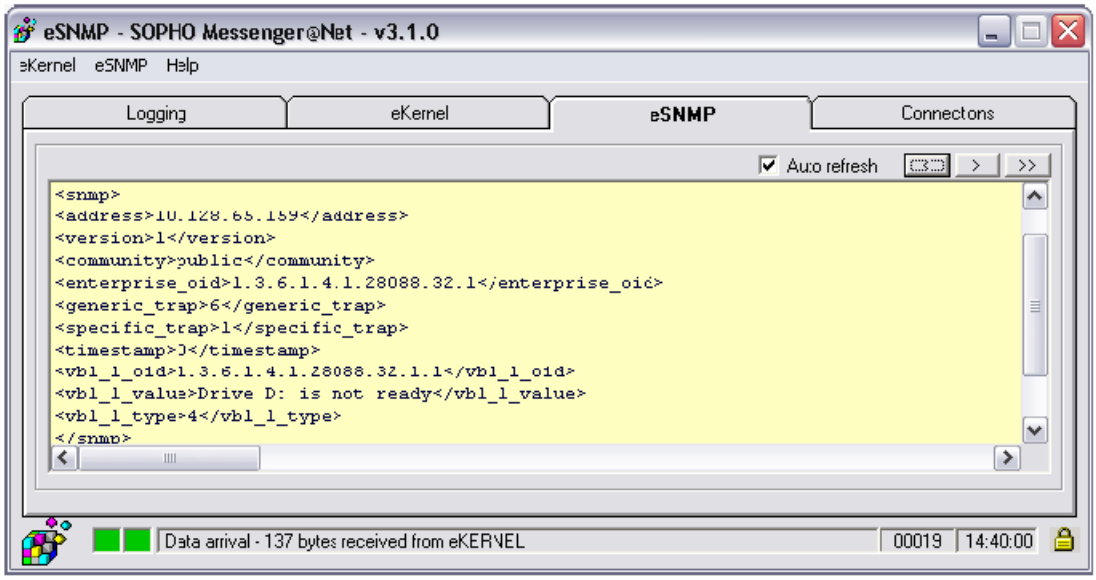


Figure 106: Message received by destination device

Monitoring type PING

You configure the monitoring of PING through the section starting with tag <PING> and ending with closing tag </PING>. This area contains a starting tag <Systems> and ending tag </Systems>.

Adding a <System> definition for each system that you want to monitor. The window in the following figure shows two definitions, one for address 127.0.0.1 and another for address 192.168.1.12.

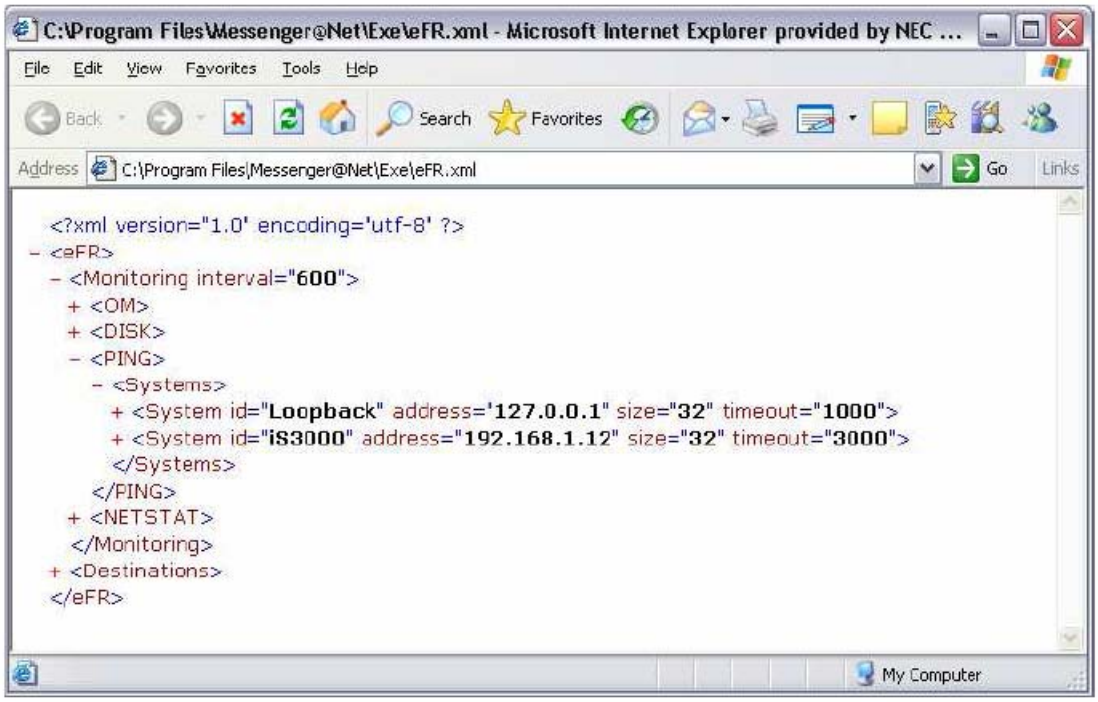


Figure 107: Monitoring type PING configuration

The definition of a system that needs to be monitored with PING is done using a <System> line.

```
<System id="Loopback" address="127.0.0.1" size="32" timeout="1000">
```

Figure 108: Monitoring type PING definition

The following parameters are available.

- Define the parameter id with a unique identifier of the monitored item. All items must have a unique identifier. For example, Lookback is an identifier. Avaya recommends that you specify a name meaningful to you to make future maintenance easier. For example, specify names such as Mail Server, Firewall, Router, IP DECT, and so on.
- Define the parameter "address" to specify the IP address used in the PING test using ICMP protocol.
- Define the parameter "size" to indicate the size of the ICMP packet during the PING test using ICMP protocol. A default value of 32 bytes meets most requirements.
- Define the parameter "timeout" to indicate the number of milliseconds to wait for a response. The default value is 1000. This means you can expect feedback on the PING command within 1 second. For some environments, 1000 is too small a number, and depending on your network topology and system resources, a larger value can be appropriate.

Avaya recommends limiting the definition of system to systems that must be verified and are expected to respond. Do not define an excessively long timeout. The larger the number of the

system and the larger the possible timeout, the less responsive the system. This means that the processing time increases, and the number of verifications decreases.

The following figure shows an example of a definition of a system for PING.

The message to be sent when a system is no longer responding to an ICMP check and the destinations informed are defined in the "NotResponding" section.

The message to be sent when a system resumes responding to an ICMP check and the destinations informed are defined in the "Responding" section.

```
:
:
<System id="Loopback" address="127.0.0.1" size="32" timeout="1000">
<NotResponding>
   <Message>Loopback adapter is not responding</Message>
   <Destinations>
      <Destination>eSNMP</Destination>
   </Destinations>
</NotResponding>
<Responding>
   <Message>Loopback adapter is responding</Message>
   <Destinations>
      <Destination>eSNMP</Destination>
   </Destinations>
</Responding>
</System>
:
:
```

Figure 109: Message sent after ICMP check

Note:

You can use the PING command to verify the ability to perform an ICMP check prior to defining a system in the eFR.xml configuration.

Figure 110: PING command verifying ability to perform ICMP check

Monitoring type NETSTAT

You can verify the configured list of connections using the NETSTAT monitoring capability. You can verify the TCP server connections in status "LISTENING", and you can verify the TCP client connections in status "ESTABLISHED".

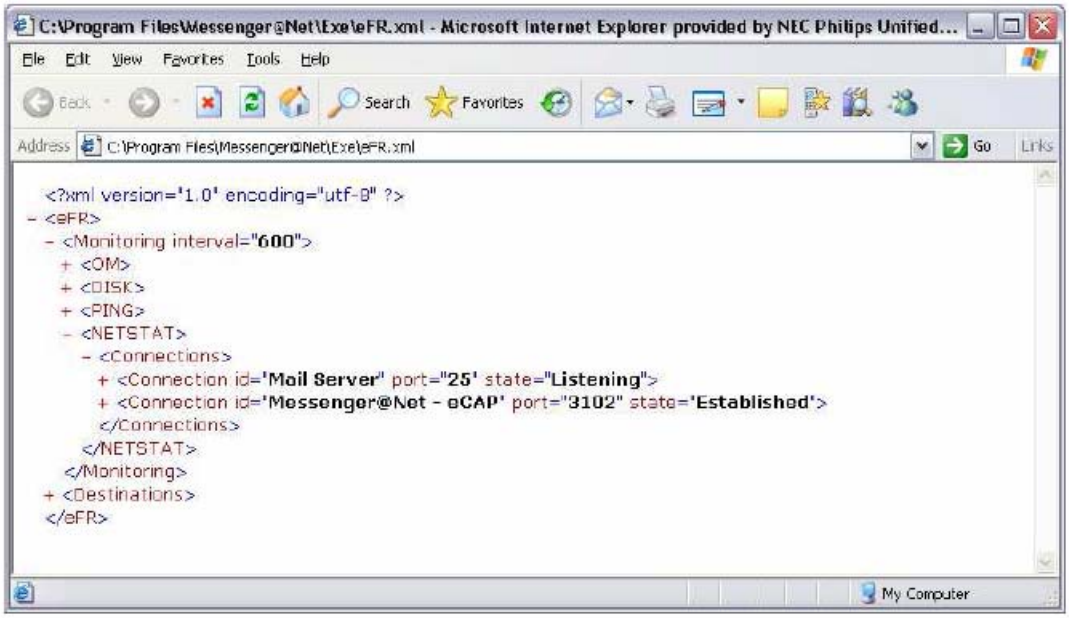


Figure 111: Monitoring type NETSTAT

The configuration for NETSTAT is in the monitoring section starts with the opening tag <NETSTAT> and the closing tag </NETSTAT>. This area contains a subsection starting with opening tag <Connections> and ending with closing tag </Connections>.

```
<NETSTAT>
<Connections>
</Connections>
</NETSTAT>
```

Figure 112: NETSTAT connections

Definition of a TCP server

A sample definition of a connection is shown in the following figure.

```
<Connection id="Mail Server" port="25" state="Listening">
```

Figure 113: TCP server definition

The definition of a connection contains the following parameters.

- Define the parameter id (name) with a unique identifier of the monitored item. For example, value Mail Server defines the connection. Avaya recommends that you choose meaningful names name for the parameters.
- Define the parameter "port" to identify the port number. For example, use the value 25 specified the port used for SMTP servers.
- Define the parameter "state".
- Define the parameter "Listening" for monitoring TCP Servers.

  A TCP Server is always up and running and in "state" of "listening" on port 25 on a local system that runs the eFR monitoring process.

Note:

You can use the command window to verify the current servers. The command "netstat –a –n" shows a screen similar to the one shown in the following figure. In the figure, a TCP server, in the first column with the value TCP, listening on port 25, in the second column ending with :25, is in the "state" of "listening".

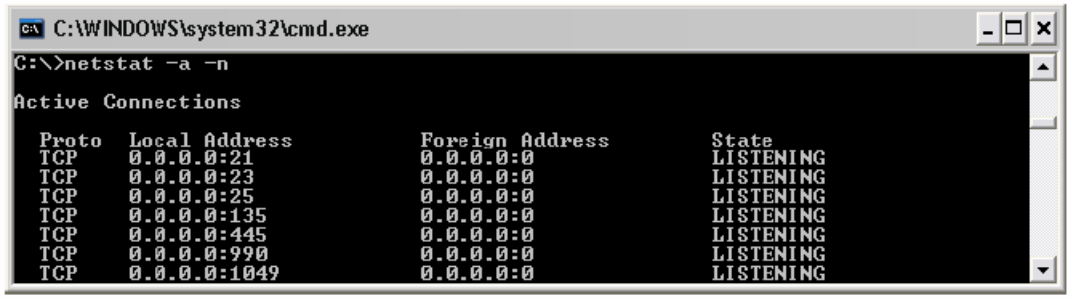


Figure 114: Verify current servers

Note:

Important: NETSTAT assumes that TCP servers remain listening.

Important:

Only apply NETSTAT monitoring to TCP Servers that are multiple access socket servers. When a TCP server is a single connection server, the TCP server is connected while it is monitored by NETSTAT.

The following figure shows a more complete definition of a configuration section monitoring the state of TCP Server that is expected to be always in state "listening". The figures illustrates the example of an environment running a local Internet Information Server with SMTP Server component "listening" to default port 25.

```
<Connection id="Mail Server" port="25" state="Listening">
  <ConnectionDown>
  <Message>Mail Server is down</Message>
  <Destinations>
    <Destination>eSNMP</Destination>
  </Destinations>
</ConnectionDown>
<ConnectionUp>
  <Message>Mail Server is up</Message>
  <Destinations>
    <Destination>eSNMP</Destination>
  </Destinations>
  </ConnectionUp>
</Connection>
```

Figure 115: Monitoring the state of the TCP server

If the SMTP Server is stopped by an administrator, the socket no longer is "listening" on port 25 and the message "Mail Server is down" is sent as an SNMPv1 trap to the configured SNMP destination.

Definition of a TCP client

A sample definition of a connection is shown in the following figure. The definition is very similar to the definition of a TCP Server, but the state is "established" instead of "listening".

```
<Connection id="Messenger@Net - eCAP" port="3102" state="Established">
```

Figure 116: Definition of a TCP client

The definition of a connection contains the following parameters.

- Define the parameter id which defines the unique identifier of the monitored item. The value Messenger@Net - eCAP in the example defines the connection. You can specify a value of your choice. Avaya recommends that you use a name meaningful to you.
- Define the parameter "port" which defines the port number. For example, the value 3102 specifies a specific port used by DECT Messenger.
- Define the parameter "state". For monitoring TCP Clients, define the value "established".

  It is expected that a TCP Server is always up and running and is in state "connected" on port 3102 with a TCP Client in the local system that runs the eFR monitoring process.

Note:

The specified port is the port on the TCP Server.

In a TCP client/server connection, a TCP Server listens on a specific port. A client can reside on the same system or on another system, and make a socket connection by specifying the

address and the port the server listens to. The following figure illustrates that the server listens on port 3102.

Figure 117: TCP client/server connection

If connection is successful, a connection is established between the client and the server. In a NETSTAT command window, a line appears illustrating that a TCP connection is found on port 3102 with the state ESTABLISHED.

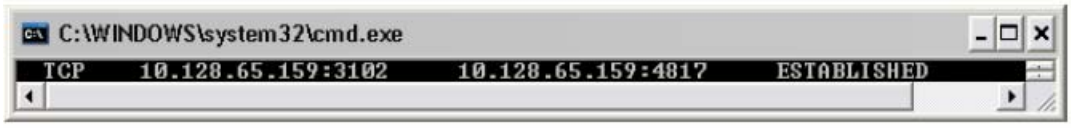


Figure 118: NETSTAT command window

The previous figure shows the client address and port in the third column, 10.1289.65.159:4817. In many environments the client is not bound to a fixed port, and receives a random assigned port number, for example 4817.

The capability of eFR monitoring client connections of NETSTAT is oriented towards application environments, with a persistent socket connection. For example, DECT Messenger is an example of a software suite based upon client/server architecture, where the majority of client modules must be connected permanently with predefined ports on the server. Typically you must monitor the TCP clients (eDMSAPI, eCSTA, eSMS, eCAP…) modules connected to eKERNEL on the configured ports.

Sample e-mail

A sample e-mail is shown in the following figure.

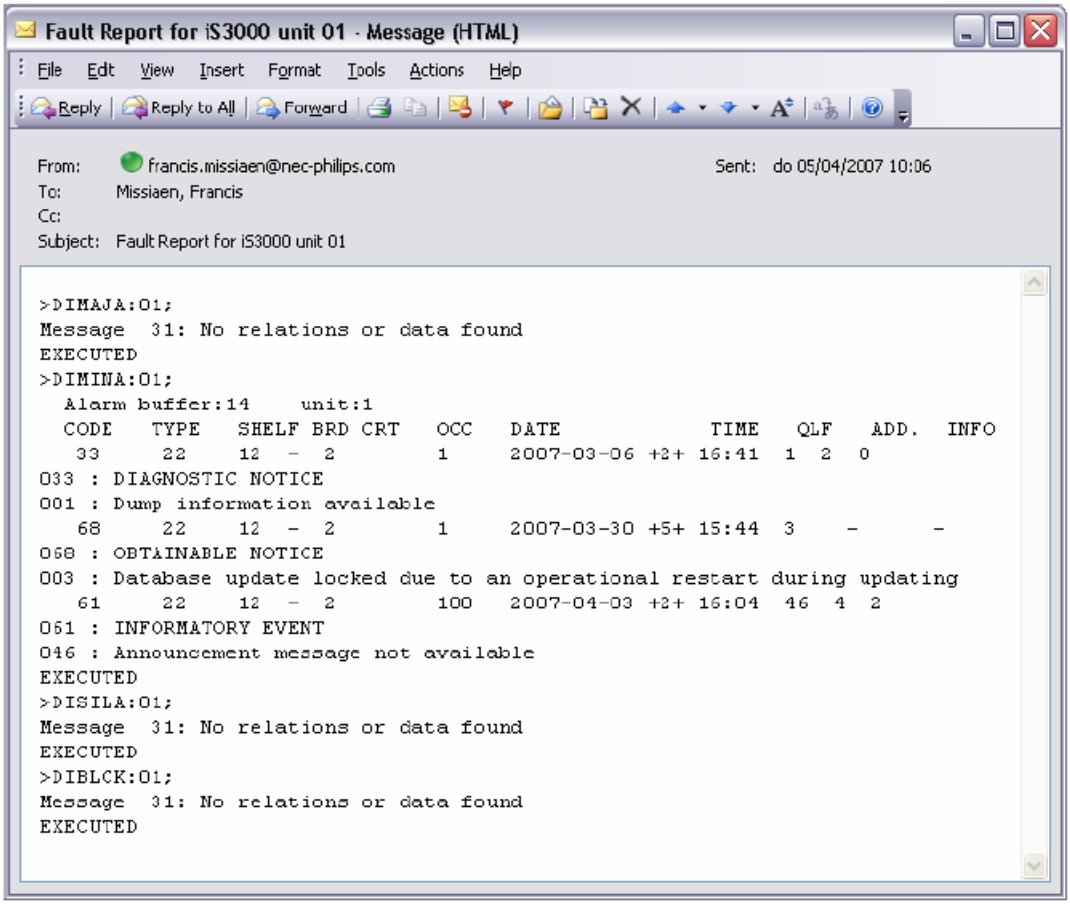


Figure 119: Sample e-mail

Chapter 19: Module - eGRID

The eGRID application gives you a view of the different tables in the databases.

You can start the eGRID.exe application without command line parameters. At startup, the window in Figure 120: eGRID startup window on page 171 is shown:

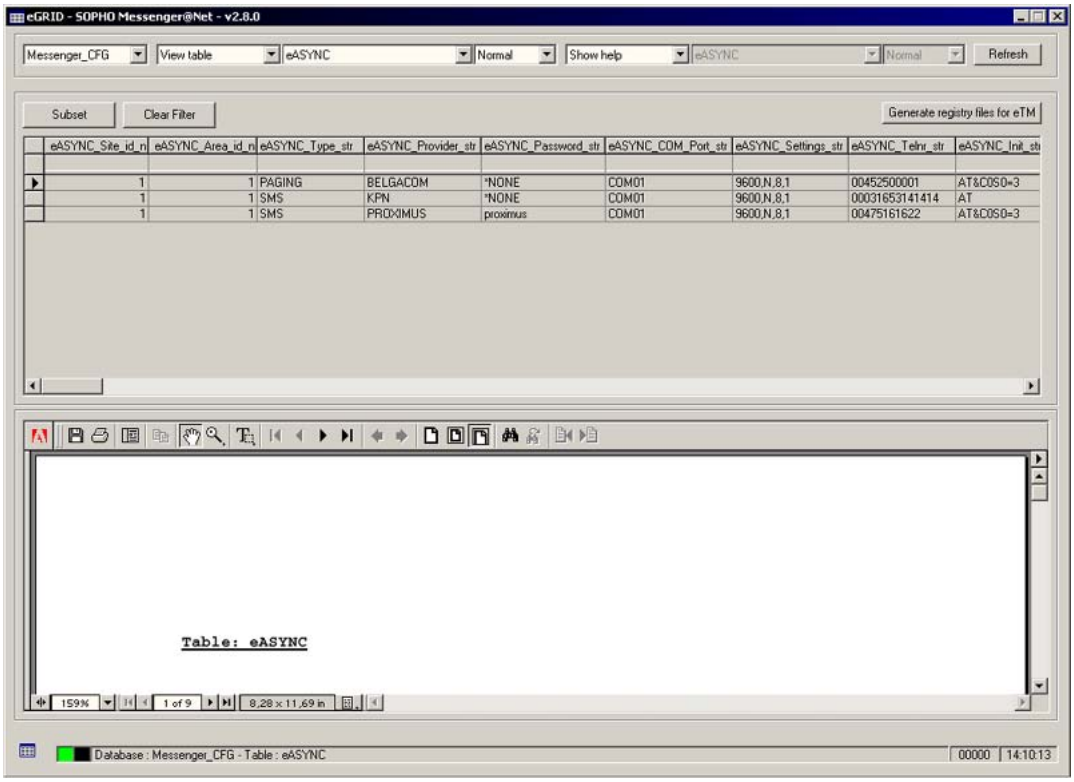


Figure 120: eGRID startup window

Seven drop-down lists are available at the top of the window. From left to right, the functions of these drop-down lists are as follows.

- Use the first drop-down list, on the far left, to select the Messenger_CFG database or the Messenger_DATA database
- Use the second drop-down list to select one of the following:
  - Perform inquiry functions using the View table
  - Perform maintenance using the Edit table
  - Export a table to a comma separated file using Export to CSV
  - Export to an HTML file using Export to HTML

- Use the third drop-down list to select a table. The available tables are retrieved automatically from the database object
- Use the fourth drop-down list to control the GRID view as follows:
  - Normal uses default view
  - Inverted uses a rotated view
  - Drag and drop to group records in Group
- The fifth drop-down list offers the following choices:
  - None uses a full-screen interface for one table
  - Show help splits the window interface in two halves: the top half is used to access the table, the bottom half is used to show the related PDF-file help information
  - You can select a second table with View another table, which splits the window in two; the upper half is used to access the first table, the lower half is used to access the second table
- The sixth drop-down list is available only if a View another table is specified. Use this list to select the second table.
- Use the seventh drop-down list, on the far right, to modify the view of the second table
  - Normal uses default view
  - Inverted uses a rotated view
  - Drag and drop to group records using Group

The example in Figure 121: eGRID with Show help mode on page 173 shows the Show help mode:

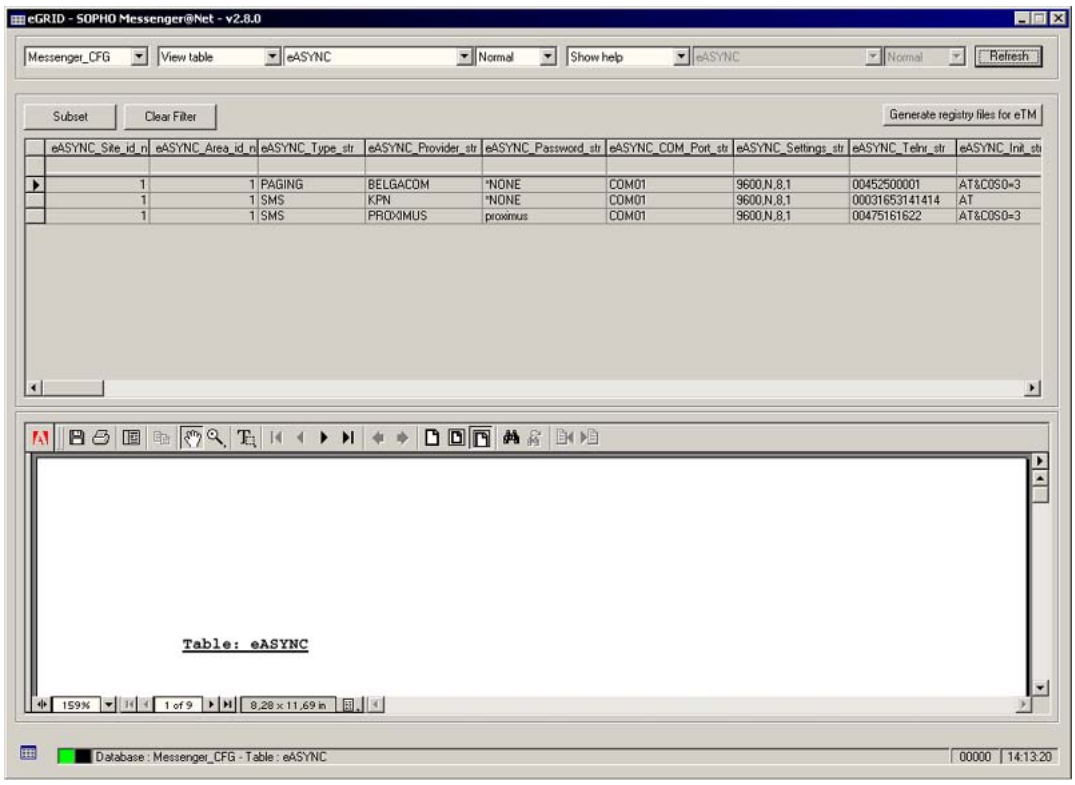


Figure 121: eGRID with Show help mode

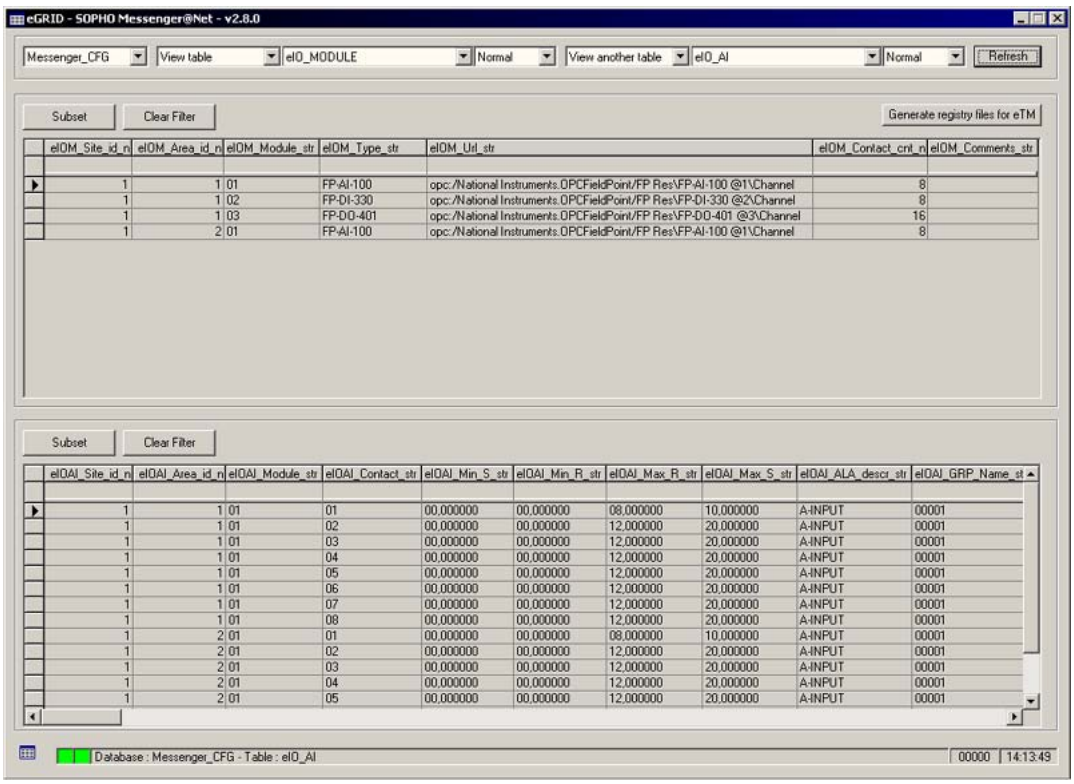


Figure 122: eGRID with View another table mode

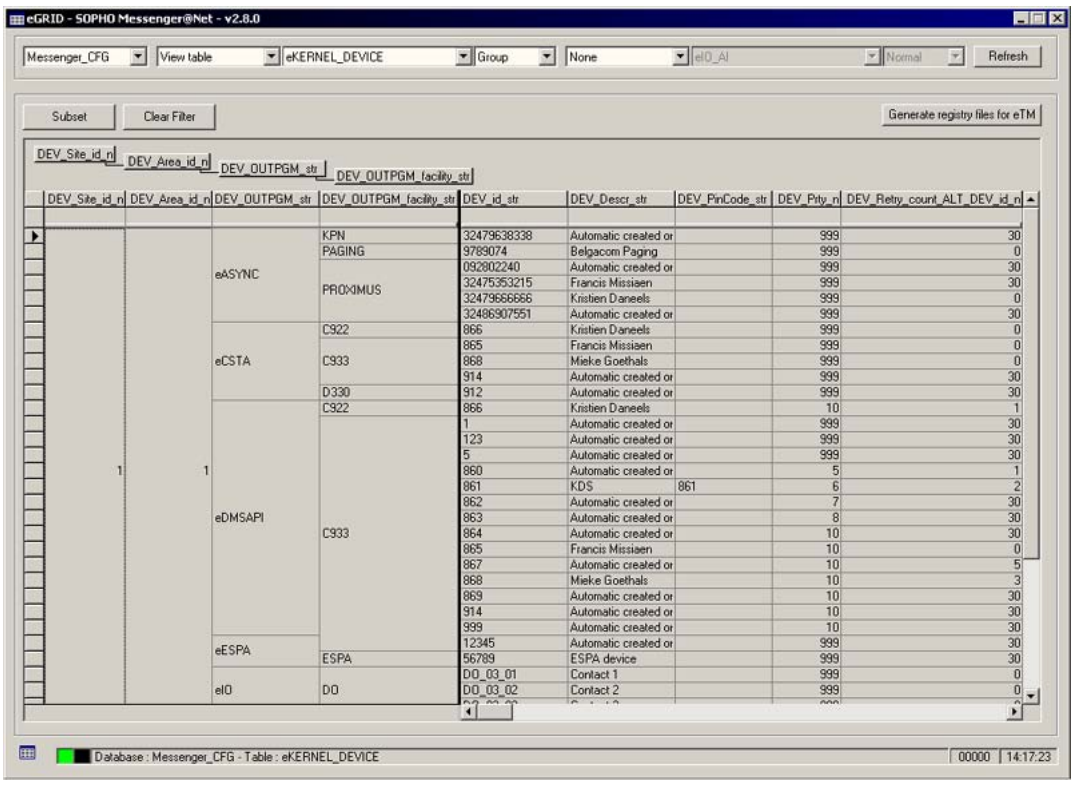


Figure 123: eGRID grouping functions

Because eGRID is the preferred access method for maintenance, an extra functionality is implemented to optimize flexibility. This functionality is referred to as Data Filtering and is handled through the command buttons Subset, Clear filter, and an entry field between the column heading and the first row.

Figure 124: eGRID Data Filtering on page 176 illustrates the usage of Data Filtering. This example shows a subset of the devices of site 3, area 1a, and output program eDMSAPI in the table eKERNEL_DEVICE. You can clear the subset criteria with the Clear Filter button, by selecting another table, or by selecting Refresh.

Note:

Incomplete information is displayed when you use Data Filtering, because only the records with matching criteria are shown.

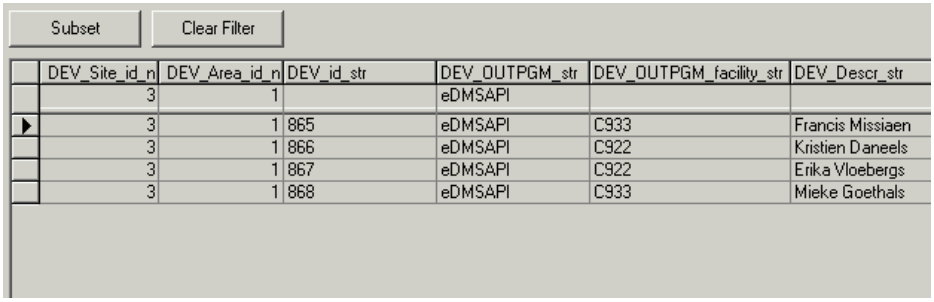


Figure 124: eGRID Data Filtering

Look at the column header to find out what data type the field has. Because the filtering function is based upon SQL instructions, you must specify subset data that results in valid SQL grammar:

- Selecting partial data (omitting training characters) is valid only for string fields with the extension _str. For example, DEV_OUTPGM_str can be part of a subset with e, eD, eDM, and so on. Boolean fields with extension _b and numeric fields with extension _n cannot be part of a subset with partial values and must be fully qualified.
- You must not specify special characters.
- String values can also be subset with syntax %EN, which select *SEND. Specify % to accept generic leading characters.

Specifying invalid filter criteria can result in errors such as the one shown in Figure 125: Invalid filter criteria error on page 176.

Figure 125: Invalid filter criteria error

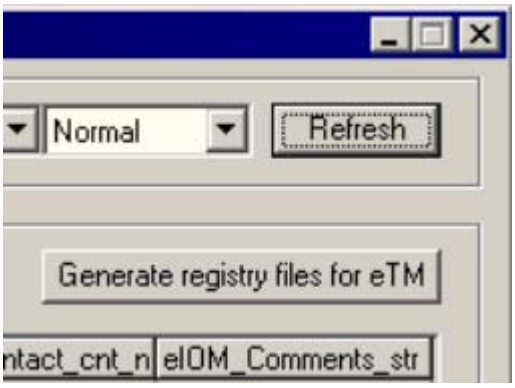


Figure 126: Accessing Generate registry files for eTM

Click **Generate registry files for eTM** to export the configuration for the module eTM, also referred to as Task Manager.

Important:

Ensure the eGRID module is not made available for unauthorized access. Remove the shortcut where applicable. The eGRID module provides direct access to the tables in the database. There is no password protection on this module.

Chapter 20: Module - eIO

Overview

The eIO module is a stand-alone application that communicates with eKERNEL. The module is capable of controlling and measuring Distributed I/O peripherals of National Instruments. eIO offers support for analogue input, digital input and digital output.

See Install PC - Step 3 - National Instruments for a detailed explanation of installation and configurations issues of the modules, and the supporting Measurement Studio components (FieldPoint Explorer and OPC Server).

All these documents contain important information that is required to understand and configure the eIO module. To avoid duplicate information, the concepts are not repeated in this chapter.

Startup

You must start the eIO application by means of a shortcut that uses the syntax described in Figure 127: Parameters in the shortcut for eIO on page 179.

```
"C:\SOPHO Messenger@Net\Exe\eIO.exe"
/Site:3
/eKernel address:*LOCAL
/eKernel port:3108
/Log drive:C
```

Figure 127: Parameters in the shortcut for eIO

The following parameters are required:

- Site

  Defines the site identifier and is used by eKERNEL to verify the identifier with the eKERNEL_TCPCLIENT and eKERNEL_INPGM settings. This identifier is also required so that eKERNEL can respond with the appropriate configuration settings.

- eKernel address

  Defines the IP address of eKERNEL. Use the special value *LOCAL to refer to the same address as the system where eIO resides. In a single-computer environment the *LOCAL

value is usually specified, because eKERNEL and eIO both share the same network adapter. When eIO is running on a different computer, the IP address of the eKERNEL must be specified.

- eKernel port

  Refers to the port number eKERNEL listens to for that specific eIO instance. This port is defined in the eKERNEL_TCPCLIENT table.

- Log drive

  Specifies the drive letter where logging files must be stored.

When all parameters are correctly specified, the eIO contacts the eKERNEL application, producing the log information shown in Figure 128: Logging information for eIO on page 180.

```
25/10/2001 10:41:35 -
S:INF:Application eIO - SOPHO Messenger@Net - v2.0.6 started with param-
eters /Site:3 /eKernel address:*LOCAL /eKernel port:3108 /Log drive:C

25/10/2001 10:41:36 -
S:INF:TCP local port 01183 connected with remote port 03108 (eKERNEL)
25/10/2001 10:41:36 -
```

Figure 128: Logging information for eIO

Once connected to eKERNEL, the eIO module requests its configuration. This is performed through a configuration request. The eKERNEL fetches the configuration from the IO_MODULE, eIO_AI, eIO_DI and eIO_DO tables and responds with all relevant parameters that are needed for eIO to continue processing. Figure 129: eIO configuration request and response on page 181 shows the configuration request and the response eKERNEL sends back.

```
<xml>
<cfgrqs>
<appl>eIO</appl>
<site>3</site>
</cfgrqs>
</xml>

<xml>
<cfgrpy>
<manufacturer>NATIONAL INSTRUMENTS</manufacturer>
<model>*BASE</model>
<mod_cnt>3</mod_cnt>
<mod_01>FP-AI-100</mod_01>
<url_01>opc:/National Instruments.OPCFieldPoint/FP Res\FP-AI-100
@1\Channel</url_01>
<cnt_01>8</cnt_01>
<mod_02>FP-DI-300</mod_02>
<url_02>opc:/National Instruments.OPCFieldPoint/FP Res\FP-DI-330
@2\Channel</url_02>
<cnt_02>8</cnt_02>
<mod_03>FP-DO-401</mod_03>
<url_03>opc:/National Instruments.OPCFieldPoint/FP Res\FP-DO-401
@3\Channel</url_03>
<cnt_03>16</cnt_03>
<ai_01_01_min_s>00,000000</ai_01_01_min_s>
<ai_01_01_min_r>00,000000</ai_01_01_min_r>
<ai_01_01_max_r>12,000000</ai_01_01_max_r>
<ai_01_01_max_s>20,000000</ai_01_01_max_s>
:
:
<ai_01_08_min_s>00,000000</ai_01_08_min_s>
<ai_01_08_min_r>00,000000</ai_01_08_min_r>
<ai_01_08_max_r>12,000000</ai_01_08_max_r>
<ai_01_08_max_s>20,000000</ai_01_08_max_s>
<log_path>C:\SOPHO Messenger@net</log_path>
<log_days>1</log_days>
</cfgrpy>
</xml>
```

Figure 129: eIO configuration request and response

When the configuration is received, the eIO updates the configuration information on the **Connections** tab, as shown in Figure 130: eIO Connections update on page 182. During this time, the eIO is temporarily less responsive to user input. This is due to the large number of OPC Server connections that take place at startup time.

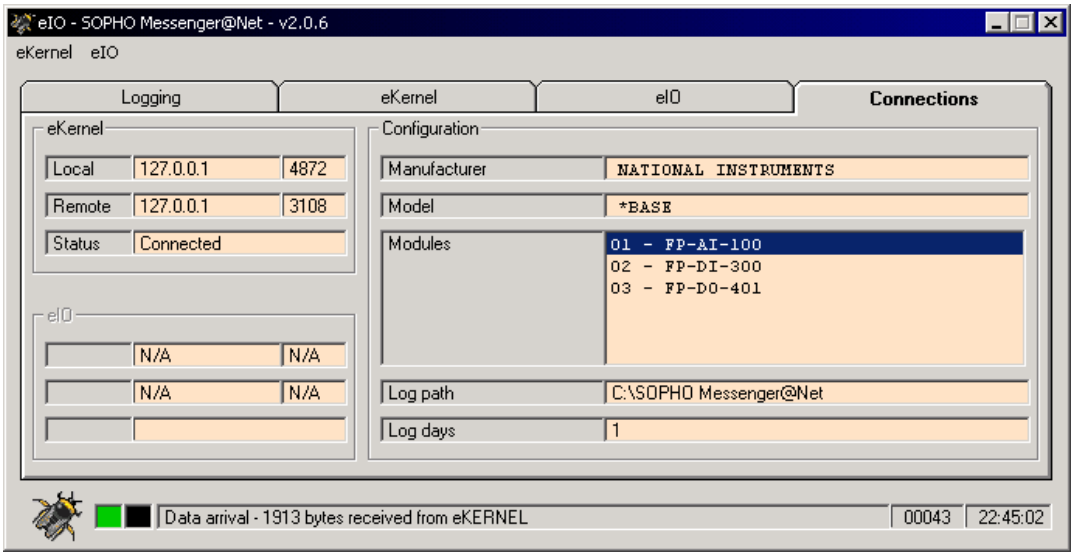


Figure 130: eIO Connections update

eIO Modules

Select the eIO tab to see a panel with details of the available modules and contacts. Use the drop-down list at the right-hand side of the window to select the module to view.

Analogue input

When you select an analogue input module (FP-AI-100), a window similar to the one in Figure 131: eIO analogue input modules on page 183 is shown. For each module a graphical display shows the available contacts. The analogue input also shows the analogue levels.

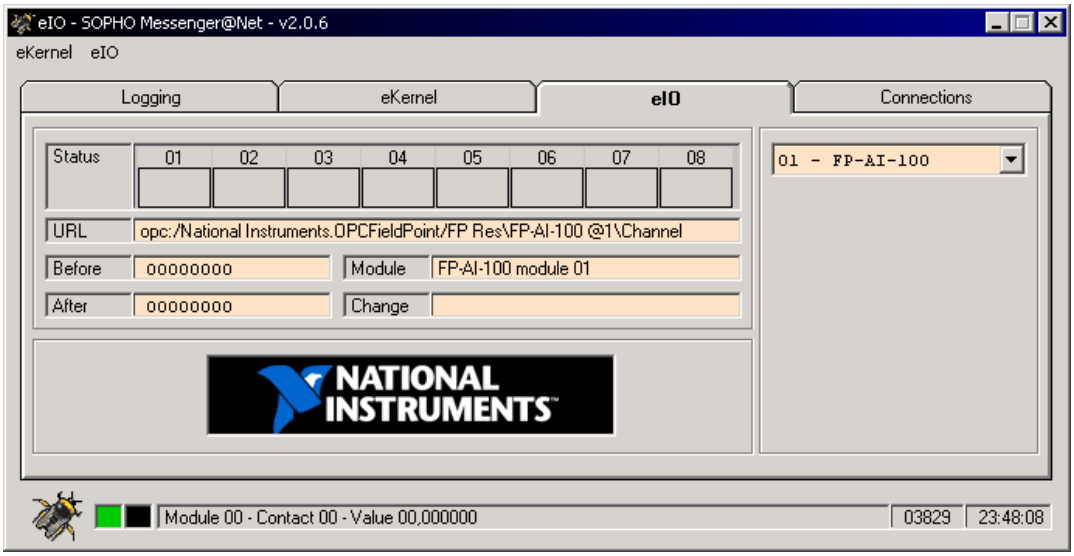


Figure 131: eIO analogue input modules

For each module, the URL is shown, and the module identifier. When you hold the mouse pointer over the status area of a contact, detailed information is shown. An example of the information provided is as follows:

```
Min (set: 02,000000 - reset: 08,000000)
Max (reset: 14,000000 - set: 20,000000)
```

The chart shown in Figure 132: Analogue input ranges on page 184 explains the behavior of these settings. The chart shows the voltage levels between 0 and 24 V on the Y-axis, and the time between 12:00 and 12:45 on the X-axis. There are four different configuration values, which are indicated in yellow. These values are retrieved from the eIO_AI table and match the environment in FieldPoint Explorer.

On the chart, the analogue measured values are shown in black. The green area is the idle zone, the red areas are alarm zones, and the grey areas are transition zones.

- When the measured value reaches 20,00000 V, a MAX ALARM condition is set. This is shown in the chart on 12:07.
- When the measured value drops to 14,00000 B, the MAX ALARM condition is reset. This is shown in the chart on 12:27.
- When the measured value drops to 02,00000 B, the MIN ALARM condition is reset. This is shown in the chart on 12:37.
- When the measured value reaches 08,00000 V, a MIN ALARM condition is set. This is shown in the chart on 12:45.

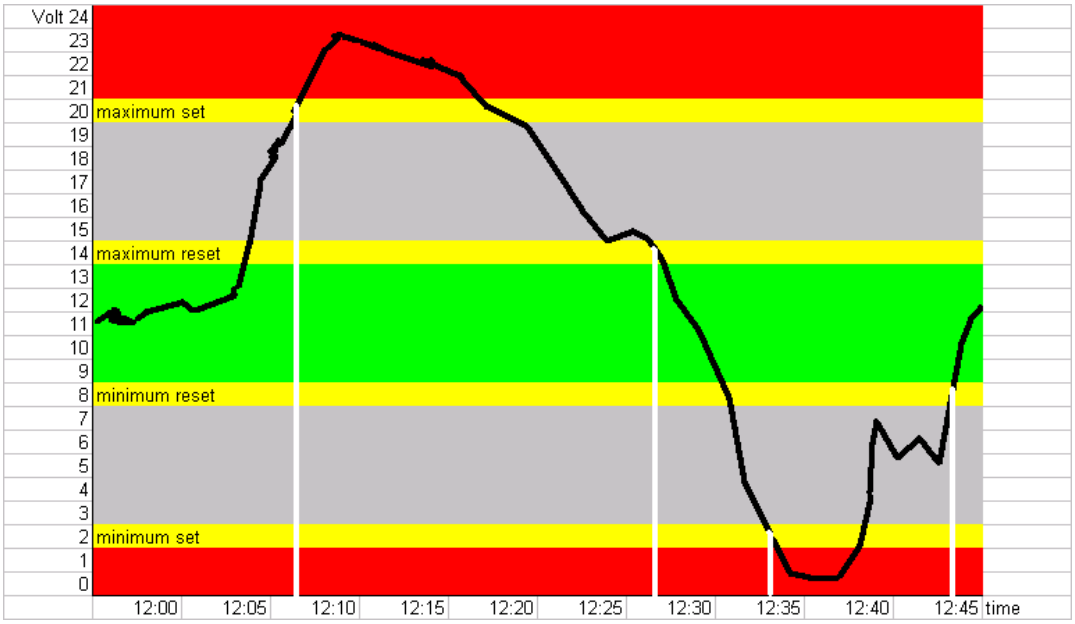


Figure 132: Analogue input ranges

Note:

Alarm values are given in pairs. Both maximum and minimum alarms are set and reset with different values. This was implemented to prevent continuous switching between set and reset when measured values are in the neighborhood of alarm values.

- Left-click in the status zone of an analogue contact to display the currently measured value.
- If a measured value generates a maximum (+) or minimum (-) alarm boundary, the Change area of the interface is updated.

Digital input (discrete input)

When you select a digital input module (FP-DI-300, FP-DI-301 or FP-DI-330), a window appears similar to the one shown in Figure 133: Digital Input module information on page 185. For each module a graphical display shows the available contacts. A grey rectangle indicates a discrete input value is Off, a green rectangle indicates a discrete input value is On.

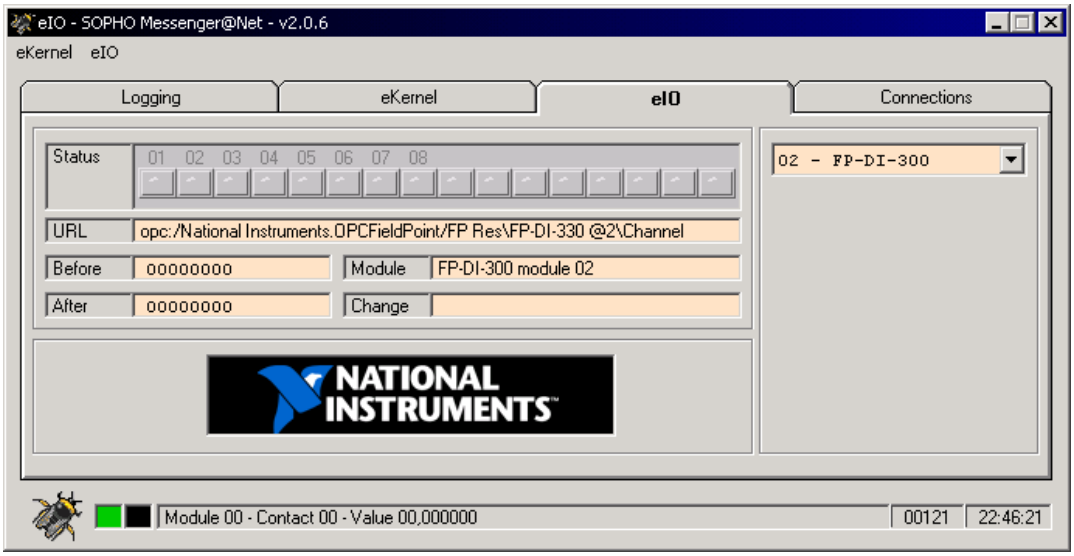


Figure 133: Digital Input module information

Note:

When the value of a contact changes from Off to On or from On to Off, the Before and After fields are updated with the status of the contact before the change occurred and the status of the contact after the change occurred. The Change field is also updated with the new value.

Digital output (discrete output)

When you select a digital output module (FP-DO-401), a window appears similar to the one shown in Figure 134: Digital Output module information on page 186. For each module a graphical display shows the available output contacts. A grey switch directed to the bottom indicates a discrete output value is Off; a grey switch directed to the upwards position indicates a discrete output value is On. In Figure 134: Digital Output module information on page 186, contacts 01 and 04 and 07 are On; all others are Off.

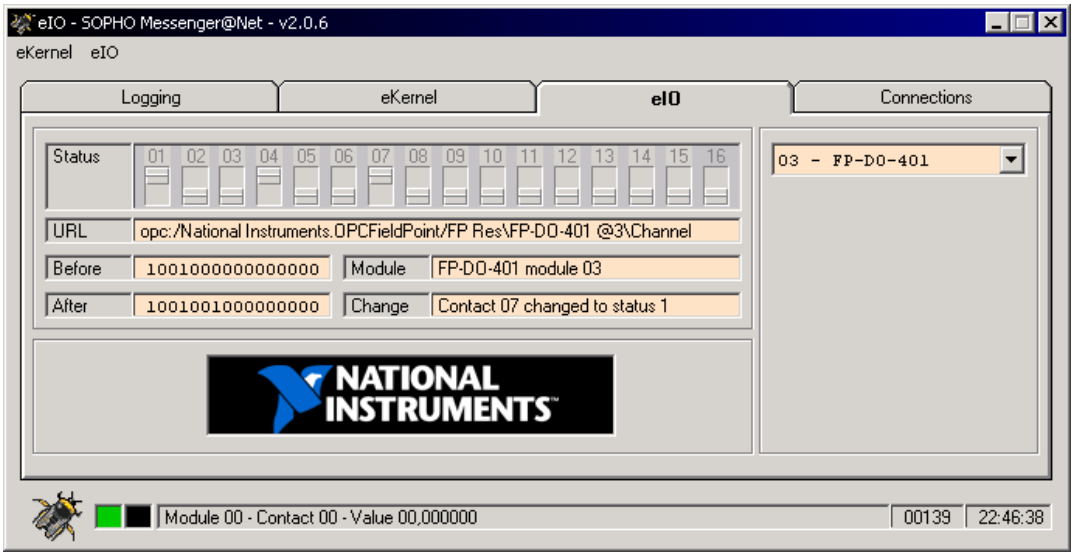


Figure 134: Digital Output module information

Note:

When the value of a contact changes from Off to On or from On to Off, the Before and After fields are updated with the status of the contact before the change occurred and the status of the contact after the change occurred. The Change field is also updated with the new value.

Note:

You can also change the status of the contacts to On or Off by using the mouse to drag the switch to the On or Off position. This must be carried out only while installing or testing. In most environments, the eKERNEL application is responsible to activate or deactivate the alarm condition of the discrete outputs. You can however manually reset a status of a contact, for example, if manual intervention is required.

Logging

The eIO module provides logging facilities, both on-screen and on log files on disk.

The on-screen logs are visible through the Logging tab, as shown in Figure 135: eIO logging on page 187.

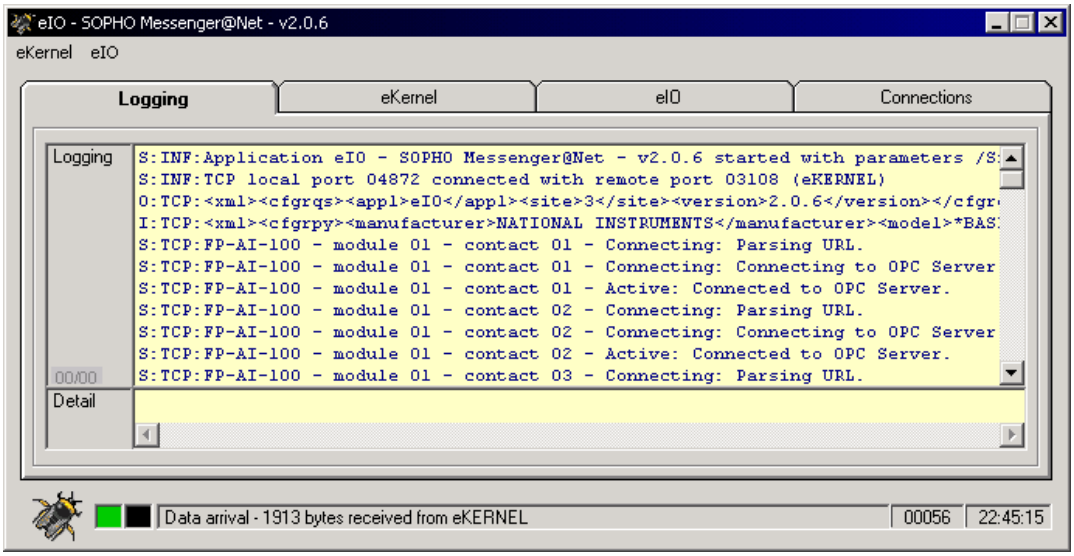


Figure 135: eIO logging

The log files on disk contain the same information as shown on-screen. On the next few pages, Figure 135: eIO logging on page 187 through Figure 140: Log example: Termination on page 189 show examples of log information saved on disk during different steps of eIO setup and use.

```
25/10/2001 10:41:35 -
S:INF:Application eIO - SOPHO Messenger@Net - v2.0.6 started with param-
eters /Site:3 /eKernel address:*LOCAL /eKernel port:3108 /Log drive:C

25/10/2001 10:41:36 -
S:INF:TCP local port 01183 connected with remote port 03108 (eKERNEL)
25/10/2001 10:41:36 -
```

Figure 136: Log example: initialization procedure

```
O:TCP:<xml><cfgrqs><appl>eIO</appl><site>3</site><version>2.0.6</ver-
sion></cfgrqs></xml>

25/10/2001 10:41:37 -
I:TCP:<xml><cfgrpy><manufacturer>NATIONAL INSTRUMENTS</manufacturer>
<model>*BASE</model><mod_cnt>3</mod_cnt><mod_01>FP-AI-100</mod_01>
<url_01>opc:/National Instruments.OPCFieldPoint/FP Res\FP-AI-100
@1\Channel</url_01><cnt_01>8</cnt_01><mod_02>FP-DI-300</mod_02>
<url_02>opc:/National Instruments.OPCFieldPoint/FP Res\FP-DI-330
@2\Channel</url_02><cnt_02>8</cnt_02><mod_03>FP-DO-401</mod_03>
<url_03>opc:/National Instruments.OPCFieldPoint/FP Res\FP-DO-401
@3\Channel</url_03><cnt_03>16</cnt_03><ai_01_01_min_s>00,000000</
ai_01_01_min_s><ai_01_01_min_r>00,000000</
ai_01_01_min_r><ai_01_01_max_r>12,000000</
ai_01_01_max_r><ai_01_01_max_s>20,000000</
ai_01_01_max_s><ai_01_02_min_s>00,000000</
ai_01_02_min_s><ai_01_02_min_r>00,000000</
ai_01_02_min_r><ai_01_02_max_r>12,000000</
ai_01_02_max_r><ai_01_02_max_s>20,000000</
ai_01_02_max_s><ai_01_03_min_s>00,000000</
ai_01_03_min_s><ai_01_03_min_r>00,000000</
ai_01_03_min_r><ai_01_03_max_r>12,000000</
ai_01_03_max_r><ai_01_03_max_s>20,000000</
ai_01_03_max_s><ai_01_04_min_s>00,000000</
ai_01_04_min_s><ai_01_04_min_r>00,000000</
ai_01_04_min_r><ai_01_04_max_r>12,000000</
ai_01_04_max_r><ai_01_04_max_s>20,000000</
ai_01_04_max_s><ai_01_05_min_s>00,000000</
ai_01_05_min_s><ai_01_05_min_r>00,000000</
ai_01_05_min_r><ai_01_05_max_r>12,000000</
ai_01_05_max_r><ai_01_05_max_s>20,000000</
ai_01_05_max_s><ai_01_06_min_s>00,000000</
ai_01_06_min_s><ai_01_06_min_r>00,000000</
ai_01_06_min_r><ai_01_06_max_r>12,000000</
ai_01_06_max_r><ai_01_06_max_s>20,000000</
ai_01_06_max_s><ai_01_07_min_s>00,000000</
ai_01_07_min_s><ai_01_07_min_r>00,000000</
ai_01_07_min_r><ai_01_07_max_r>12,000000</
ai_01_07_max_r><ai_01_07_max_s>20,000000</

ai_01_07_max_s><ai_01_08_min_s>00,000000</
ai_01_08_min_s><ai_01_08_min_r>00,000000</
ai_01_08_min_r><ai_01_08_max_r>12,000000</
ai_01_08_max_r><ai_01_08_max_s>20,000000</
ai_01_08_max_s><log_path>C:\SOPHO Messenger@net
</log_path><log_days>1</log_days></cfgrpy></xml>
```

Figure 137: Log example: Configuration procedure

```
25/10/2001 10:41:37 -
S:TCP:FP-AI-100 - module 01 - contact 01 - Connecting: Parsing URL.

25/10/2001 10:41:37 -
S:TCP:FP-AI-100 - module 01 - contact 01 - Connecting: Connecting to OPC
Server.

25/10/2001 10:41:38 -
S:TCP:FP-AI-100 - module 01 - contact 01 - Active: Connected to OPC Serv-
er.

:
:
:

25/10/2001 10:41:39 -
S:TCP:FP-DI-300 - module 02 - contact 01 - Connecting: Parsing URL.

25/10/2001 10:41:39 -
S:TCP:FP-DI-300 - module 02 - contact 01 - Connecting: Connecting to OPC
Server.

25/10/2001 10:41:39 -
S:TCP:FP-DI-300 - module 02 - contact 01 - Active: Connected to OPC Serv-
er.
:
:
:
25/10/2001 10:41:39 -
S:TCP:FP-DO-401 - module 03 - contact 01 - Connecting: Parsing URL.

25/10/2001 10:41:39 -
S:TCP:FP-DO-401 - module 03 - contact 01 - Connecting: Connecting to OPC
Server.

25/10/2001 10:41:39 -
S:TCP:FP-DO-401 - module 03 - contact 01 - Active: Connected to OPC Serv-
er.
```

Figure 138: Log example: Binding to OPC Servers

```
25/10/2001 10:42:47 - O:TCP:<xml><msgrqs><type>DI</type><module>02</mod-
ule><contact>01</contact><sts>1</sts></msgrqs></xml>
```

Figure 139: Log example: Message Request

```
25/10/2001 10:41:59 -
O:TCP:<xml><pgmsts><value>Shutdown</value></pgmsts></xml>

25/10/2001 10:41:59 -
S:INF:Application ended
```

Figure 140: Log example: Termination

Chapter 21: Module - eKERNEL

General

eKERNEL is the core engine of the DECT Messenger, and is in the basic implementation the only module that accesses the database.

The eKERNEL receives information from various input sources, and exchanges information with various output sources.

Communication with eKERNEL is performed through TCP/IP stream sockets, where the eKERNEL acts as a server. The other modules that communicate to the eKERNEL act as clients.

All data streams are formatted in XML format, and are delimited with an <xml> start tag and an </xml> end tag, followed by CHR$(13) and CHR$(10). Within these tags, a number of keywords are embedded with their appropriated values.

In short, the eKERNEL is the central engine of the DECT Messenger, and controls the functioning of all the other modules. Figure 141: eKERNEL interface on page 191 shows the eKERNEL interface.

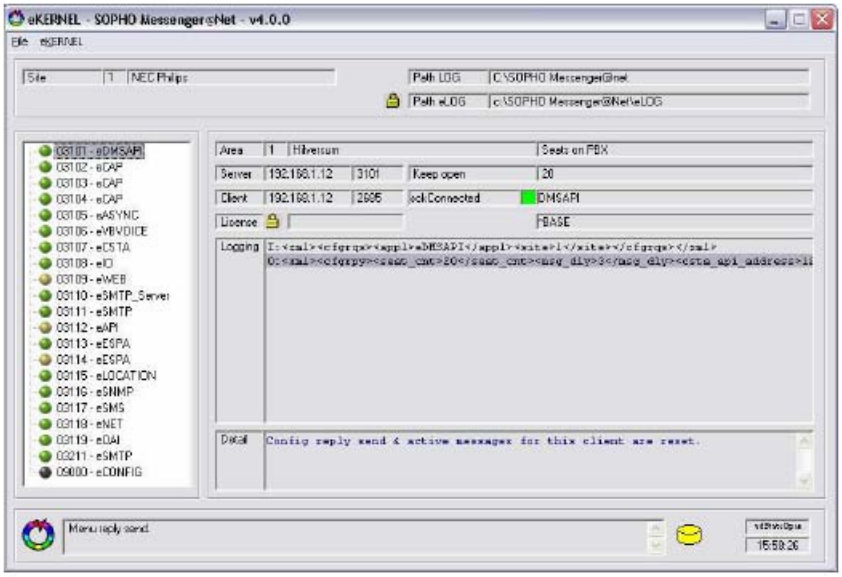


Figure 141: eKERNEL interface

License Manager

The Avaya DECT Messenger package is secured by a licensing system, to prevent unlicensed usage of certain modules and clients.

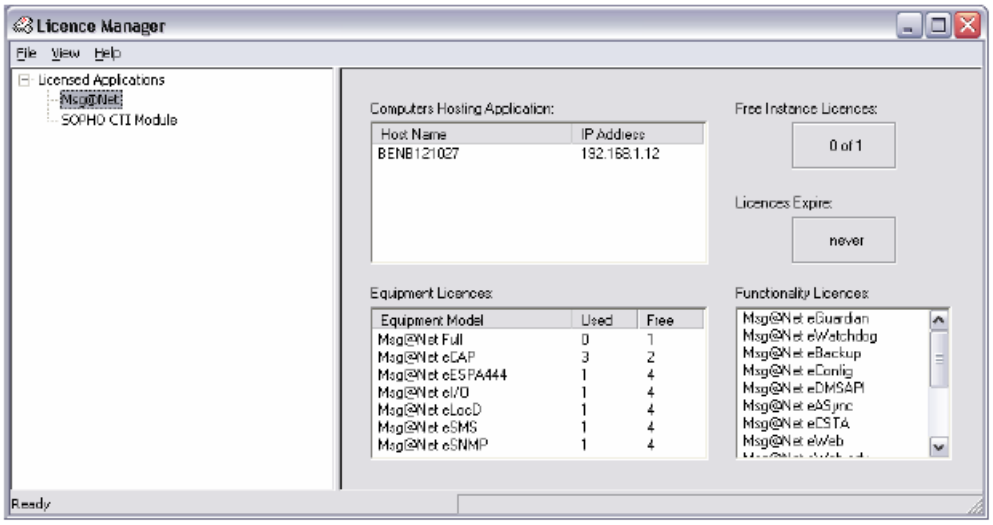


Figure 142: License Manager

At startup, the eKERNEL searches for a valid license. If there is no valid license (because, for instance, the license expired or licensing system is not installed), the eKERNEL program aborts. To determine if the installed licensing system is valid, use the Avaya License Manager. If this component is not yet installed, refer to Install PC - Step 1e - License Manager.

If a valid application is bound, all Tabs of the eKERNEL program show a crossed-through key icon, while the eKERNEL Tab shows a clear key icon.

- Clear key icon: [license bound]

- Crossed-through key icon: [license unbound]

Equipment and Functionality models

The licensing system distinguishes between equipment models and functionality models.

- The following models are assigned as equipment:

  eCAP, eESPA, eIO, eLOCATION, eSMS, and eSNMP.

- The following models are assigned as functionality:

  eWATCHDOG, eBACKUP, eCONFIG, eDMSAPI, eASYNC, eCSTA, eWEB, eSMTP, eSMTP_server, eAPI, and eVBVoice.

A key difference between equipment and functionality models is the count of available licenses, as illustrated in Table 15: License examples on page 193. Equipment models have only a specified number of available licenses, while functionality models have an unlimited number of available licenses.

Table 15: License examples

Module	total licenses	used	free
ECAP [equipment]	3	2	1
EESPA [equipment]	2	2	0
ECSTA [functionality]	unlimited	N/A	N/A
DMSAPI [functionality]	unlimited	N/A	N/A

Whenever a client connects to the eKERNEL through a configuration request, the eKERNEL determines whether the client is an equipment or functionality model.

- Equipment model

  If the client is defined as equipment, the eKERNEL tries to bind this equipment to the license. Success depends on the availability of a free license. To verify how many licenses are available on the system, use the Avaya License Manager. This program gives an overview of bound licenses. If an equipment model disconnects, its license is unbound and the total of free equipment licenses is increased. On the other hand, if an equipment module connects, the total of free equipment licenses is decreased (and the total of used equipment licenses increased). A bound equipment model receives a valid configuration reply. If the equipment model cannot be bound, a status message is sent as follows: **<pgmsts>NO LICENCE AVAILABLE</pgmsts>**.

- Functionality model

  If a client is a functionality model, the eKERNEL checks if the given functionality is available in the license system, and if so, sends a valid configuration reply. If the requested functionality is not available in the license, a status message is sent as follows: <pgmsts>NO LICENCE AVAILABLE</pgmsts>.

When a client (equipment or functionality model) is licensed, eKERNEL also provides a configuration reply, and the specific eKERNEL tab-page is updated with a clear key icon. If the

license cannot be bound or no correct functionality is available, the eKERNEL tab-page is not updated, and the crossed-through key icon remains.

eAPI and eWEB

eAPI and eWEB do not send configuration requests. To ensure that eAPI and eWEB are licensed properly, eKERNEL checks these two functionality models individually. If eAPI and eWEB are found in the License system, the TCP/IP ports for any clients of this kind are opened. If no eAPI or eWeb functionality is available in the license system, the ports are not opened and the eAPI or eWeb clients are not able to connect.

License maintenance

Every 24 hours, at midnight, each client that is connected to the eKERNEL is checked to determine whether the license is still valid. If the Application license (eKERNEL program) is expired, eKERNEL sends a message to all of its clients and closes all of its ports, so no client is able to reconnect. After checking the Application license, equipment and functionality models are checked for validation.

When installing a new license with more available equipment models or adding one of the functionality models eWeb or eAPI, the eKERNEL must explicitly be told about the new license. To tell eKERNEL about a new license, use the menu command: **`eKERNEL > License > Recheck license of all clients (open port)`** as shown in Figure 143: Rechecking licenses on page 194.

Note:

eKERNEL makes this same check automatically at midnight. Note that rechecking all licensing is a time-consuming process.

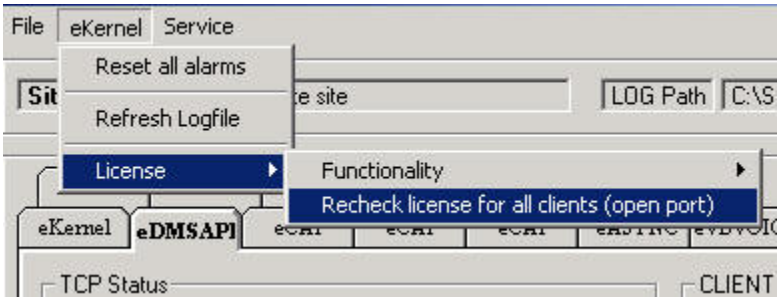


Figure 143: Rechecking licenses

When installing a new license with new available functionality (eGuardian or eWatchdog), the eKERNEL must be told about the new license. To tell eKERNEL about a new license, use the

menu command: **eKERNEL > License > functionality** as shown in Figure 144: Adding license functionality on page 195.

Note:

eKERNEL makes this same check automatically at midnight. Note that rechecking all licensing can take a little time.

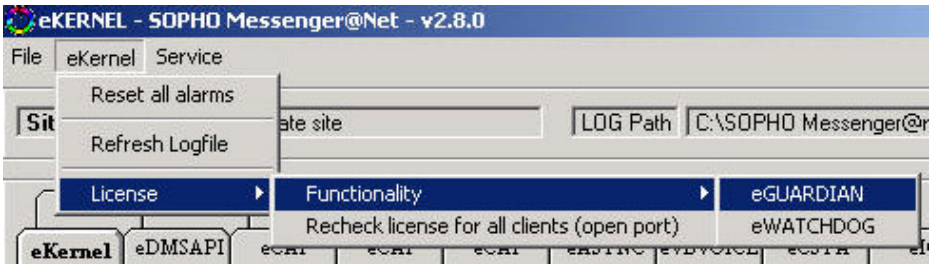


Figure 144: Adding license functionality

If a license is installed with fewer equipment licenses than there are clients that need them, the clients that are no longer licensed continue to function until the license is rechecked. When the license is rechecked, unlicensed clients receive a status message <pgmsts>NO LICENCE AVAILABLE</pgmsts>, and their TCP/IP port is closed. This same principle applies to functionality models such eWEB and eAPI.

External interfaces

There are a number of external interfaces that you can attach to the eKERNEL. These interfaces can act as input source, output source, or play both roles.

An eKERNEL without any external models is unable to perform work. A minimum configuration requires at least one input source (for example, eCAP for capturing a TELEVIC PROTOCOL CONVERTOR signalling system), and one output source (for example, eDMSAPI for sending E2-data messages to cordless DECT handsets).

You can attach additional input sources to the product. For example, eCAP for other signalling systems, eIO for unpowered contact detection, eSMTP for receiving MAIL, and eWEB for receiving messages from the Internet.

You can attach additional output sources to the product. For example, eSMTP for sending electronic mail, eASYNC for sending short messages to GSM and Pagers, eCSTA for sending voice call based user-to-user messages, eVBVOICE for handling outgoing voice calls.

Note:

In release 4.0, the maximum amount of modules has been increased from 21 to 30.

Database

The eKERNEL application is the only application that communicates directly with the databases. Every external application receives its configuration from eKERNEL.

There are two databases. One is named Messenger_CFG.mdb, and another is named Messenger_data.mdb. Both databases are in Microsoft Access 2000 format, and are processed through applications written in Microsoft Visual Studio 6.0 (Visual Basic and C++).

In release 4.0, the Messenger_DATA database can also reside on SQL Server 2005 Express, SQL Server 2000 Desktop Engine (MSDE), or an external SQL 2000 or 2005 server.

The Messenger_CFG.mdb contains several tables, and defines the configuration of the DECT Messenger software. These tables determine the behavior of the product.

The Messenger_data.mdb contains several tables, and are an internal work space of DECT Messenger. Some models, such as eKERNEL, access this database heavily. Avaya recommends that you avoid using the data database, except for problem determination and recovery services.

TCP Connections

The eKERNEL acts as a TCP Server, and typically listens to several ports.

You can configure the TCP clients in the configuration database.

The status of each connection is visible on the screen of the eKERNEL application. In normal operation, an active connection is indicated by the color green. A client that is not connected is indicated by the color yellow. Other colors indicate an intermediate state or an error condition.

Logging

You can log every event to ASCII log files and on the screen. The on-screen buffer is limited to 100 records. To see details, double-clicking on the log records. The log files are commonly stored in the directory specified in the CFG_Log_path_str field of the eKERNEL_site table.

If CFG_Log_path_str = C:\SOPHO Messenger@net, all log files are stored in the C:\SOPHO Messenger@Net\log\eKERNEL directory. Each day a new log file is created at midnight.

Menu options

- **File > Exit**

  This option closes the eKERNEL application.

- **eKERNEL > Reset all alarms**

  This option clears all active alarms in the data database.

- **eKERNEL >Refresh logfiles**

  This option closes and reopens the log file of the eKERNEL application. Perform this action before opening the log file for the current day, so all data that is still in memory is copied to the log file.

- **Service > Delete all data records**

  This function deletes all the records of the selected table. Perform this action to be sure you start with a clean data database at the customer.

Watchdog

When the Watchdog facility is enabled, an icon of a dog is visible at the right top of the window, as shown in Figure 145: Watchdog enabled on page 198. When active, Watchdog sends the command string entered in the CFG_Watchdog_cmd_str field of the eKERNEL_SITE table to the connected com port. The command string is sent every CFG_Watchdog_interval_n seconds.

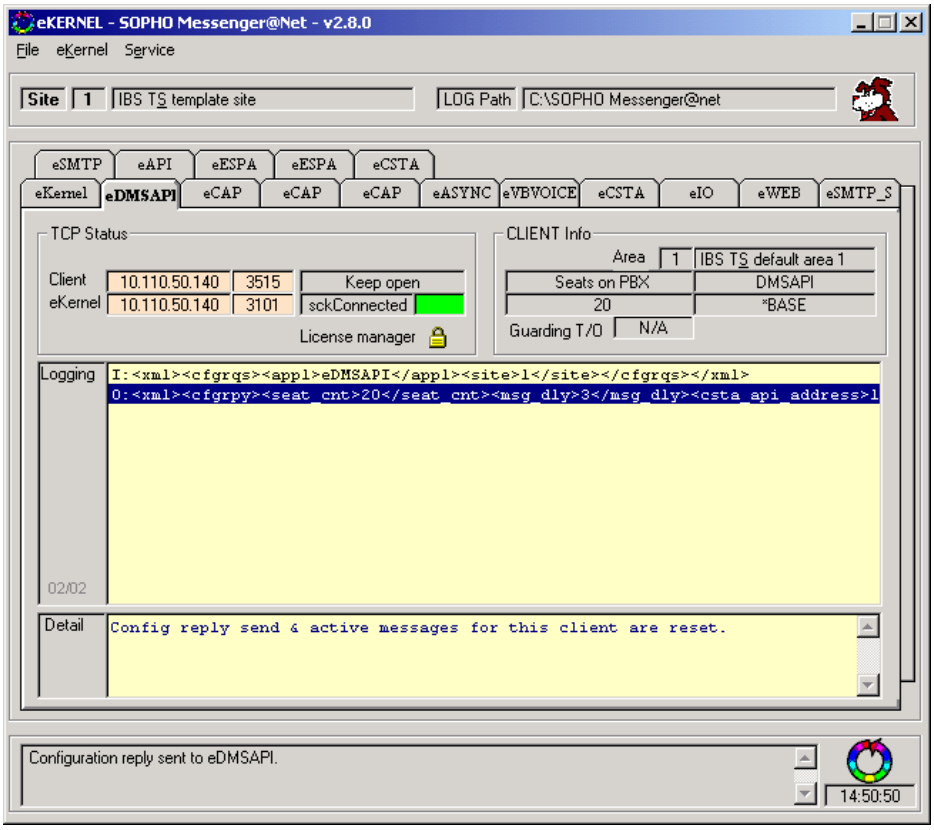


Figure 145: Watchdog enabled

Guarding

For every input program, the administrator can configure a guarding facility, as shown in Figure 146: Guarding information on page 199.

If guarding is activated for a specific input program, an indication is given in the Client information frame for every TCP/IP client.

The Guarding T/O field specifies the timeout that is defined in the eKERNEL_guarding table, and the last event field text box shows the number of seconds that have passed since the last request was received from the TCP/IP client. Once the Last event value exceeds the guarding timeout value, a guarding alarm is generated.

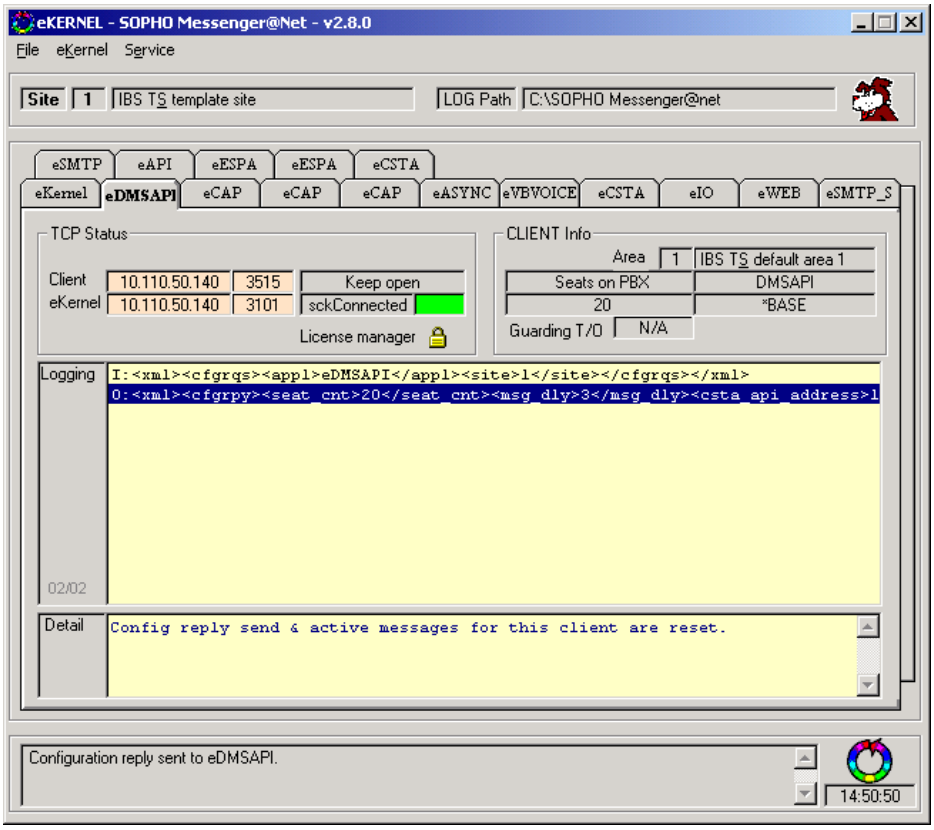


Figure 146: Guarding information

Chapter 22: Module - eLICENSE

Important:

Due to the ongoing development of the DECT Messenger product suite, some modules that provide additional functionality may become available after the initial release of DECT Messenger 4.0.

The following modules are described in this document but are not available at initial General Availability.

- eFR
- eLICENSE
- eLOCATION
- eSMS
- eSNMP
- eVBVOICE

The eFR module is an add-on module and is licensed separately through the eLICENSE module. Some of the modules listed in this attention box are available only on a site-specific basis.

In DECT Messenger release 4.0, portions are packaged and distributed as add-on modules, including eLICENSE.

License mechanism

The license mechanism used is the major difference between regular modules and add-on modules. Regular modules are licensed through the License Manager and protected by a hardware dongle and license string. Add-on modules use a different license mechanism.

Ordering

Add-on modules are currently not supported. Contact Avaya for further information.

Install module eLICENSE

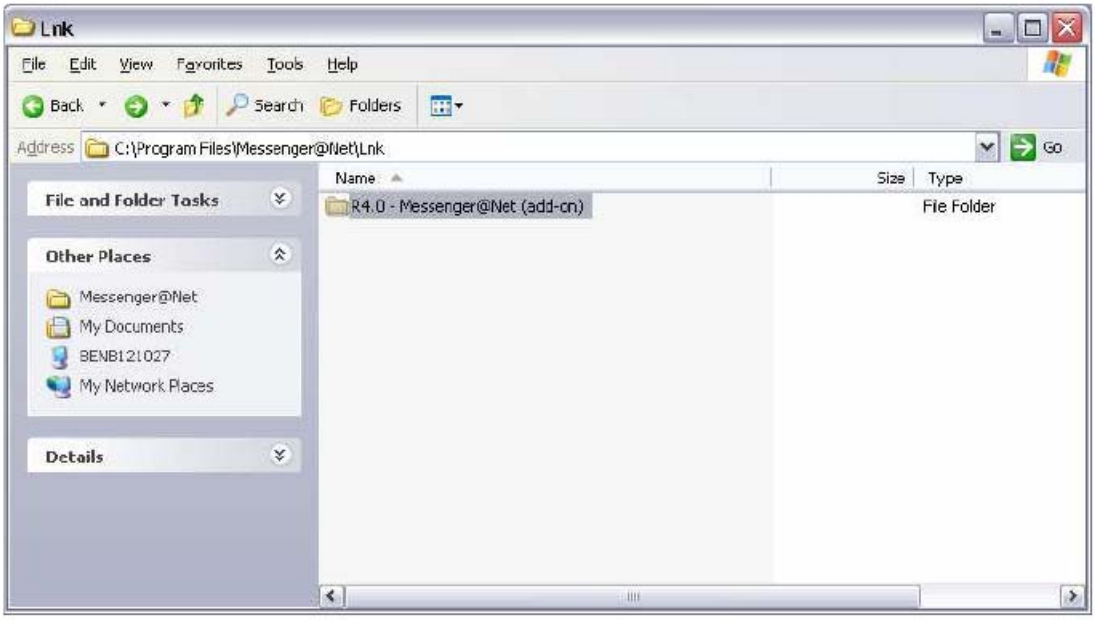
The module eLICENSE provides a windows application used in to retrieve the fingerprint of the system. The eLICENSE software must be installed at the destination system where the

add-on modules are activated. In the majority of the cases, add-on modules, such as eFR, reside on the MESSENGER system running eKERNEL. However, when add-on modules are installed on different systems, install eLICENSE on the systems where add-on modules are installed.

Installing eLICENSE

1. Locate and open the eLICENSE module from the CD-ROM image, found in directory 09 - Add-ons - 2008.04.23 > 2008.04.23 – eLICENSE.
2. Double-click the SETUPEX.EXE file to install.
3. Accept all defaults.

   After installation, the software resides in C:\Program Files\Messenger@Net\Exe.
4. Open Explorer and navigate to C:\Program Files\Messenger@Net\Lnk.
5. Copy the directory R4.0 – Messenger@Net (add-on) and drag it to the Start button (bottom-left of the windows desktop).

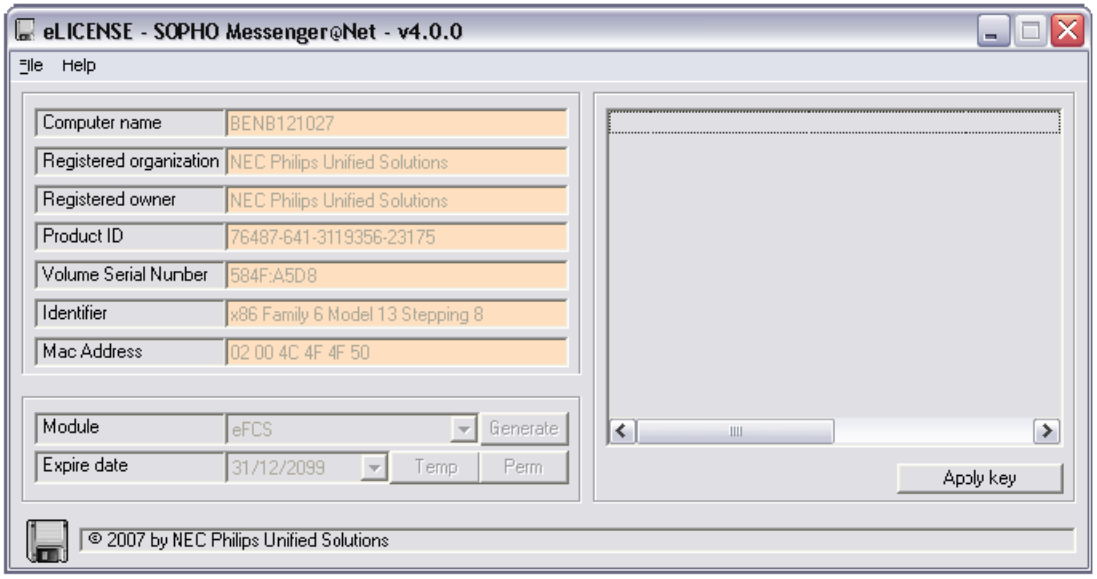


Run module eLICENSE

Running module eLICENSE

1. Start the module eLICENSE by double-clicking file C:\Program Files \Messenger@Net\Exe\eLICENSE.exe

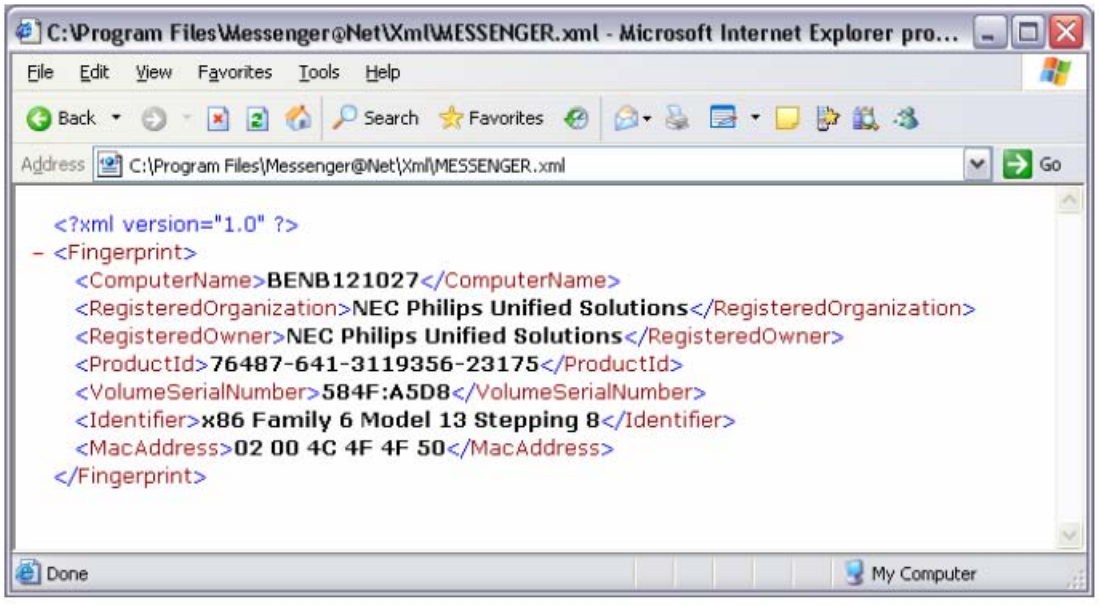
   A window similar to the one in the following figure appears.



2. Save the XML file by selecting File > Save Fingerprint.
3. Accept the default path C:\SOPHO Messenger@Net\Xml .

   The default filename is usually your computer name with extension Xml. For example, on computer MESSENGER the file is MESSENGER.Xml.

   The resulting XML file contains 7 parameters retrieved from your hardware and operating system.
4. Use Internet Explorer to open the XML file. Double-click the file residing in C:\Program Files\Messenger@Net\Xml.



Applying the key

The license file name contains the originating computer name, for example, MESSENGER, as well as the licensed module name, for example, eFR. Therefore, if you request a license for computer MESSENGER for add-on eFR, the filename is MESSENGER_eFR.reg.

A sample license file follows.

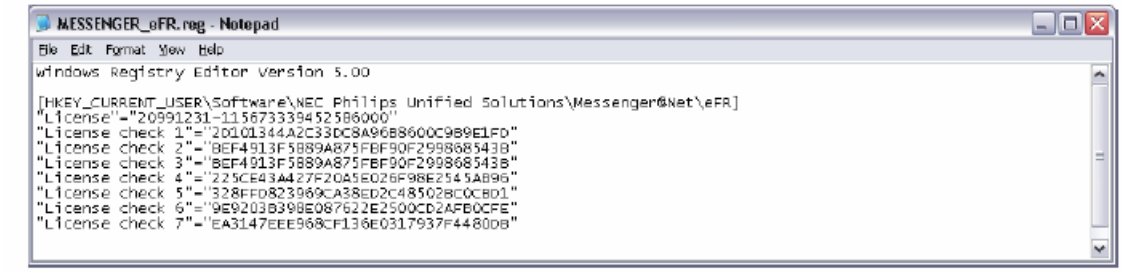


Figure 147: Sample license file

Applying the license file

1. To apply the license file, place the license file in any directory of your destination system, for example, in C:\Program Files\Messenger@Net\Reg.
2. Double-click the license file.

   A message appears.
3. Click Yes when the message appears that asks if you are sure you want to add the information in the license files, for example, C\Program Files\Messenger@Net\Reg \Messenger_eFR.reg, to the registry.

   A Registry Editor message appears that states that the Information in your license file is successfully entered into the registry.
4. Click OK when the Registry Editor message appears.

When you request a license key for other add-on modules, you receive additional REG-files that also must be merged.

Disaster recovery

If your system is replaced by another system, the following message box appears to indicate that the previous license file is no longer valid. For example, if your hard disk and Ethernet adapter are both replaced, this warning appears.

Figure 148: Invalid license key warning

If this occurs, request an updated license file.

Note:

If you cannot wait to restart the system, type the word Evaluation in the text box in the invalid license key warning, shown in Figure 148: Invalid license key warning on page 205. This gives you access to the module, but it does not apply a permanent key. By entering the word Evaluation, you can continue to run the application until you receive a new valid license for the changed system.

Note:

The invalid license key warning appears each time the software starts unless you apply the new license key.

Chapter 23: Module - eLOCATION

Important:

Due to the ongoing development of the DECT Messenger product suite, some modules that provide additional functionality may become available after the initial release of DECT Messenger 4.0.

The following modules are described in this document but are not available at initial General Availability.

- eFR
- eLICENSE
- eLOCATION
- eSMS
- eSNMP
- eVBVOICE

The eFR module is an add-on module and is licensed separately through the eLICENSE module. Some of the modules listed in this attention box are available only on a site-specific basis.

The eLOCATION module is used with SIP DECT systems only. It collaborates with eDMSAPI and eKERNEL to allow location detection of SIP DECT extensions.

The eLOCATION module receives message location detection requests from the eKERNEL module and responds with location reply results to eKERNEL. Then, eKERNEL resumes processing and dispatches the resulting alarm message to the defined group of users.

Each eLOCATION connects to a DAP Controller and communicates with it to find out the last used RPN for the requested SIP DECT extension. This involves a sockets connection between eLOCATION and the DAP Controller.

For this connection, eLOCATION is the TCP client and the DAP Controller is TCP server listening on port 28008.

Initialization

You use a shortcut to start the eLOCATION module. The following figure shows an example of the required keywords.

```
"C:\SOPHO Messenger@Net\Exe\eLOCATION.exe"
/Site:1
/eKernel address:*LOCAL
/eKernel port: 3203
/Log drive:C
```

Figure 149: eLocation module keywords

The following keywords are used to start the eLocation module.

- The "site" keyword indicates the site that is assigned to the eLOCATION module
- The "eKernel address" keyword indicates the IP address that is assigned to the eKERNEL module. The eLOCATION contacts this IP address to connect to the eKERNEL
- The "eKernel port" keyword indicates the port number that is assigned in the configuration for the eLOCATION client instance.

When the "eLOCATION" application starts the application attempts to connect to the "eKERNEL". The attempt to connect is based upon the address and port information obtained from the Shortcut.

At connection, the "eLOCATION" requests that the "eKERNEL" provide additional configuration settings. This request is known as a configuration request. The "eKERNEL" then authenticates the client and responds with a "configuration reply".

The following figure illustrates the initialization procedure.

```
S:INF:Application eLOCATION - SOPHO Messenger@Net - v4.0.0  started
with parameters /Site:1 /eKernel address:*LOCAL /eKernel port:3203 /Log
drive:C

S:INF:TCP local port 02545 connected with remote port 03203 (eKERNEL)
O:TCP:<xml><cfgrqs><appl>eLOCATION</appl><site>1</site></cfgrqs></xml>

I:TCP:<xml><cfgrpy><la_address>192.168.32.10</la_address><la_port>28008<
/la_port><generaltimeout>10</generaltimeout><polling_intv>60</polling_
intv><log_path>C:\SOPHO
Messenger@net</log_path><log_days>14</log_days></cfgrpy></xml>
S:INF:TCP local port 02549 connected with remote port 28008 (DCC)
```

Figure 150: Initiation procedure

When the configuration is received, a screen similar to the one shown in the following figure is shown. View the configuration in the "Connections" tab.

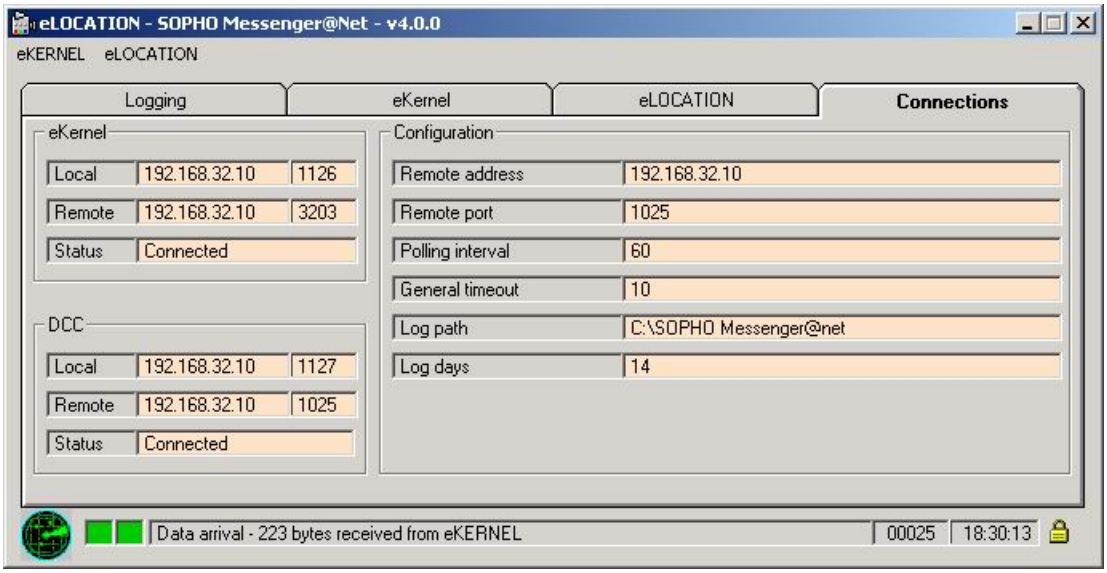


Figure 151: eLocation configuration connections tab

The logging tab shows logging information.

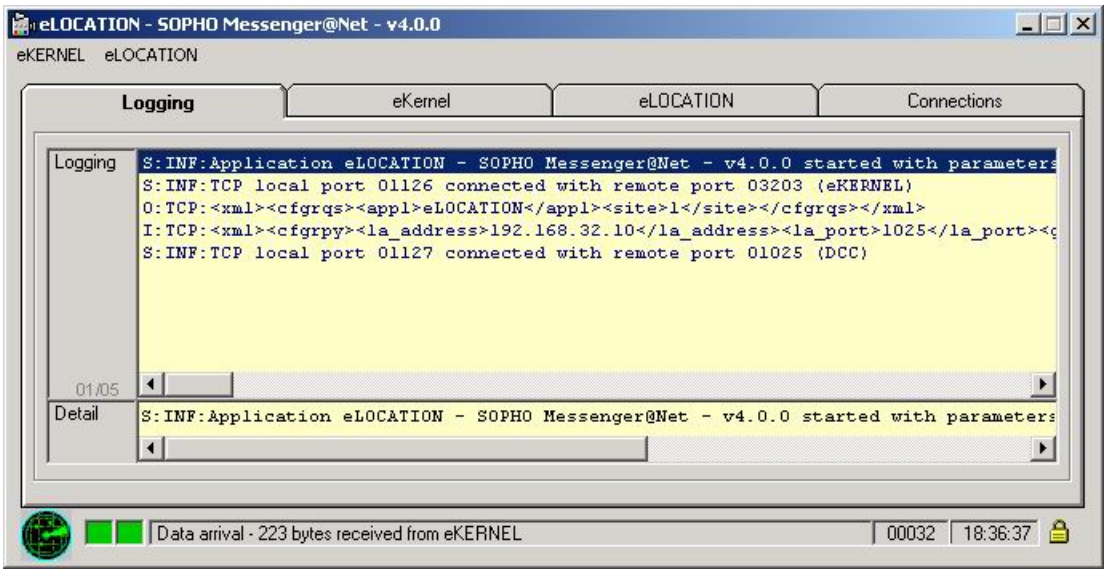


Figure 152: eLocation configuration logging tab

Program activity

When the startup phase is done, eLOCATION has two permanent socket connections.

- One socket connection is established with the eKERNEL and is used to exchange "location requests" and "location reply" data flow.
- The other socket connection is established with the DAP Controller PC (DCC) and follows a proprietary protocol to retrieve location information of SIP DECT extensions.

The link with the DAP Controller exchanges data on regular time interval, so both parties can verify the activity of the remote party. This guarding is typically represented by the informational message "Data arrival – 1 bytes received from DCC". When the connection is broken, eLOCATION attempts to reestablish a session.

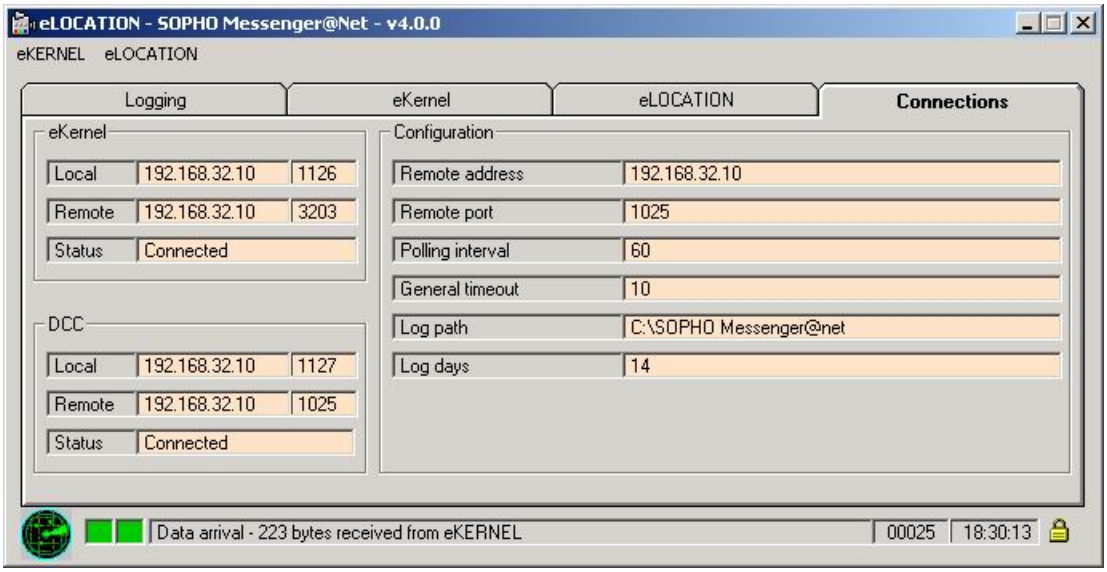


Figure 153: eLocation program activity

Note:

Define The IP address of the PC that runs eLOCATION in the PBX. When the IP address is not defined, the connection request is rejected by the DAP Controller. The two green indicators on the bottom of the eLOCATION panel illustrate the successful connection state between both external parties.

Keep track of the progress of the actual location detection process through the eLOCATION tab. The location detection process takes place when eKERNEL sends a location request to eLOCATION. The following figure shows the idle state at system startup prior to executing the first request.

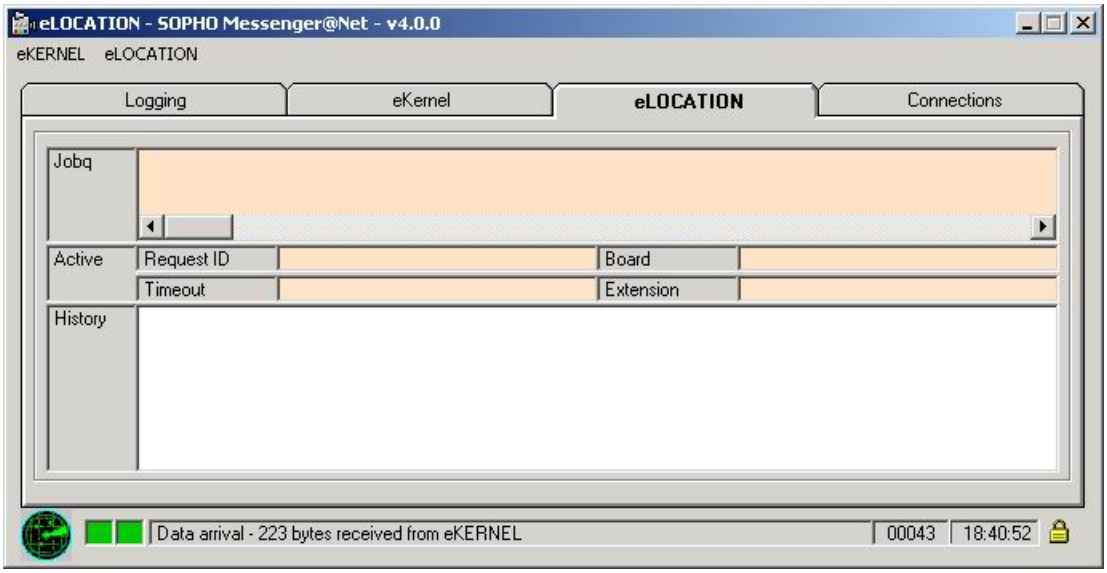


Figure 154: Idle state of system at startup

After the processing of a location request, the panels are updated as shown in the following figure.

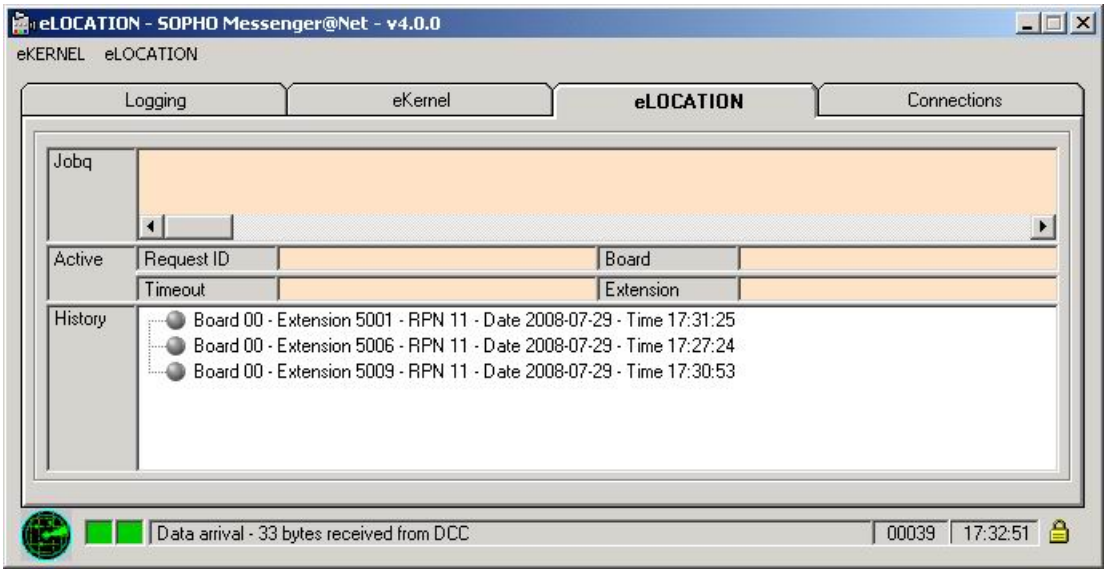


Figure 155: eLocation updated panels

The "history" panel keeps track of all location request results. In the previous figure several requests with positive feedback are presented in a history tab.

The "logging" panel in the following figure illustrates the detailed data exchange with the DAP Controller for the previous requests.

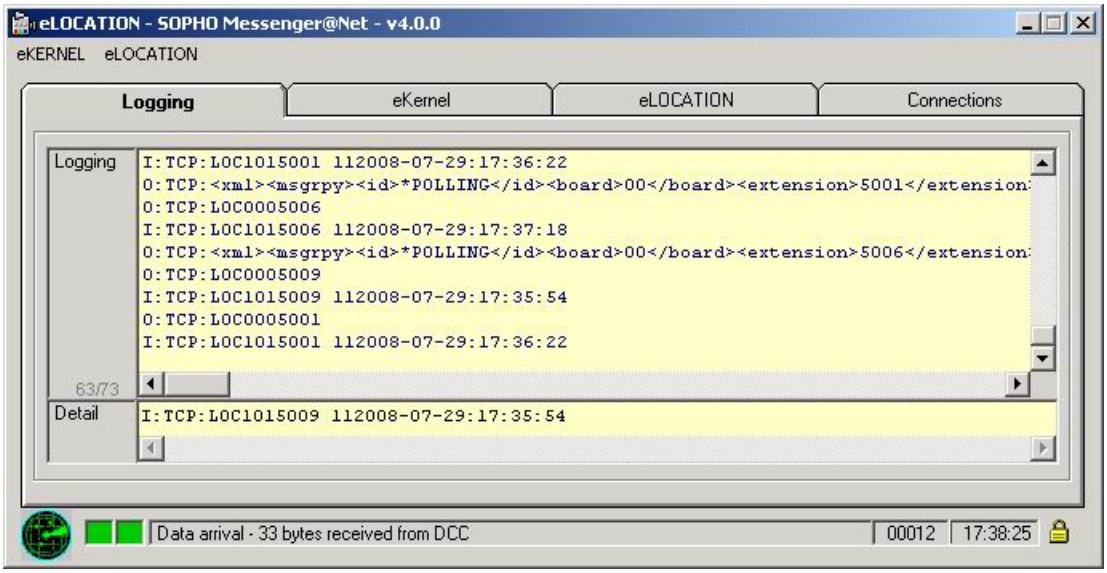


Figure 156: Logging panel

Architecture

There are several steps involved to make eLOCATION operational.

The effected items in the setup of the functionality follow.

- One special number with '*LA' type should be defined in the eDMSAPI module in the eDMSAPI_INBOUND table (i.e. number 112 or 911).

  In the following figure, eCONFIG is used to illustrate the configuration process.

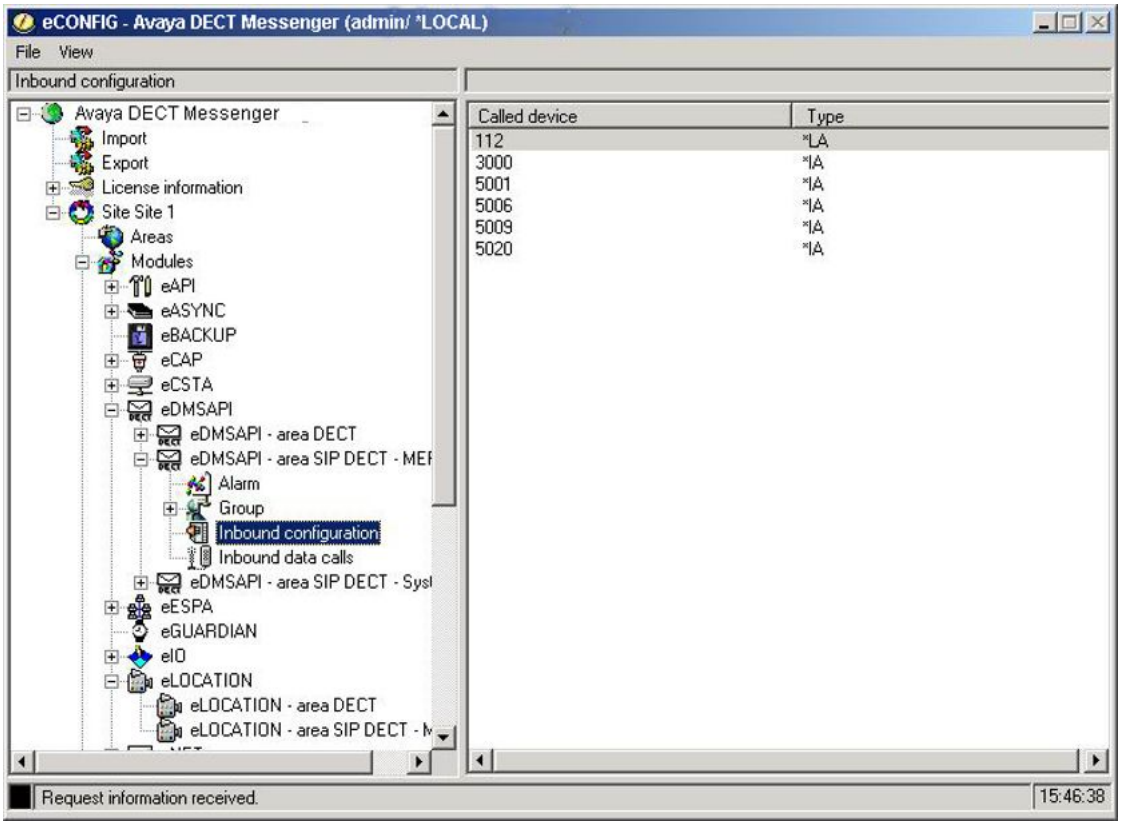


Figure 157: eConfig illustrates the configuration process

In the following two figures, extension 112 with type '*LA' is defined for location detection.

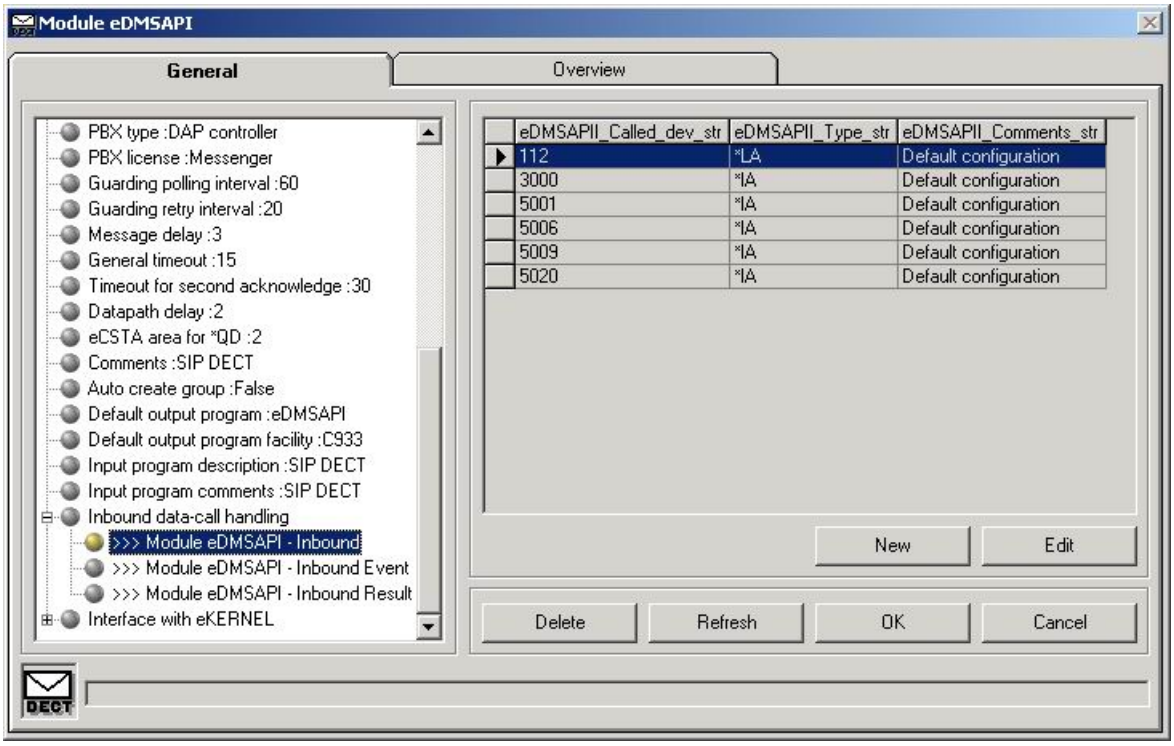


Figure 158: Module eDMSAPI

- The next item that you must define is the correct record in the table eDMSAPI_INBOUND_EVENT for the special number defined above in the same manner as it is done for all other extensions.

  The following figure shows an example of the special number configuration in the eDMSAPI_INBOUND_EVENT table.

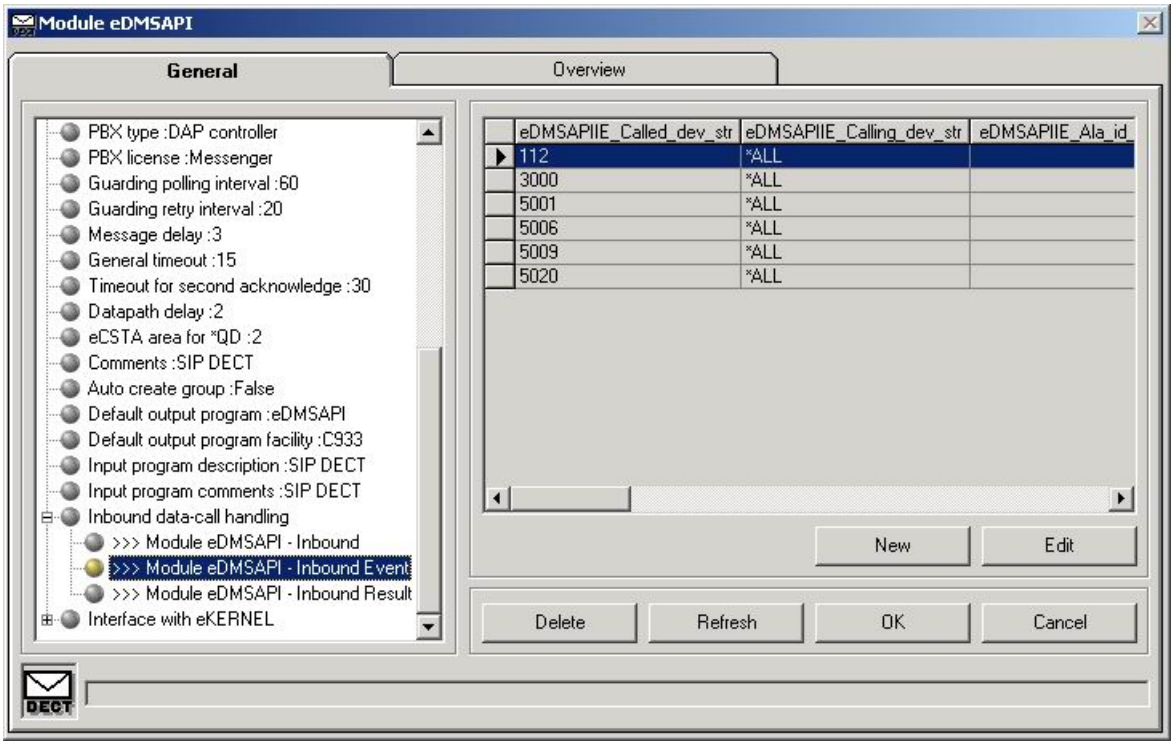


Figure 159: Module eDMSAPI inbound event

The following figure illustrates detailed configuration for a special number in eDMSAPI_INBOUND_EVENT table.

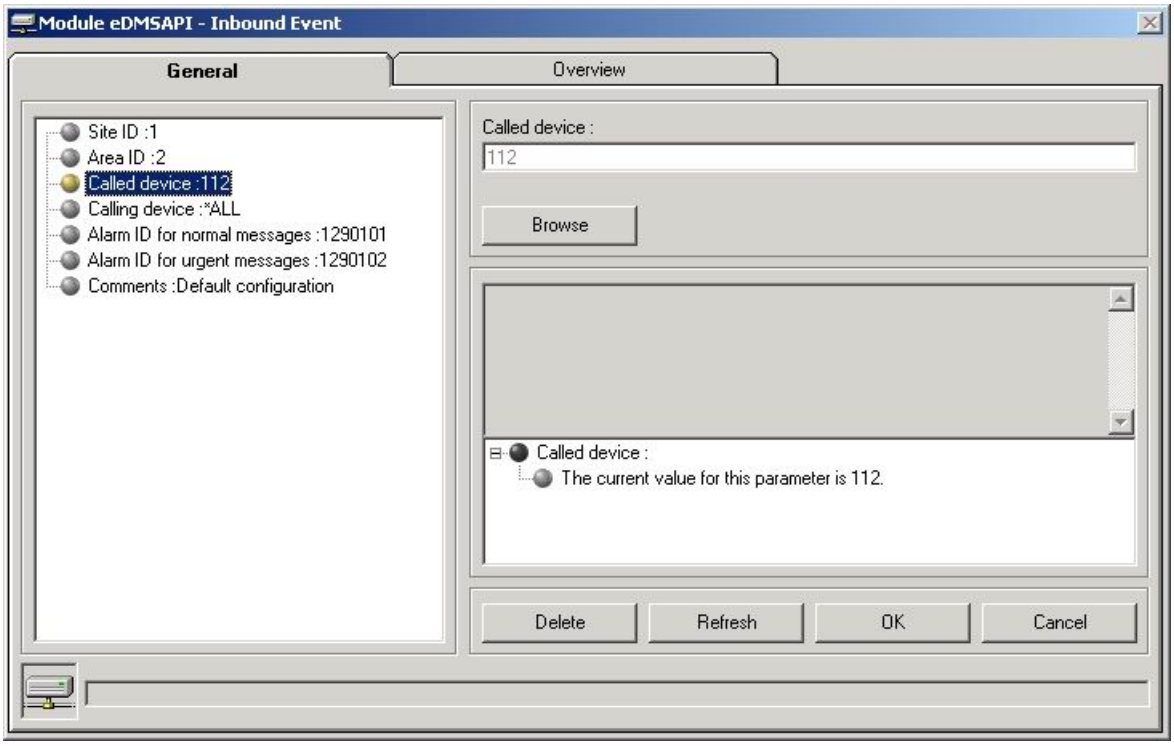


Figure 160: Module eDMSAPI inbound event - detailed configuration

- Next the eKERNEL module contacts the eLOCATION module to find out the physical location of the SIP DECT user.

  In DECT Messenger prior to release 2.9.10, the eKERNEL contacts only the eLOCATION of the same site and the same area as the eDMSAPI module that alerted the alarm. This means if the alarm was detected from eDMSAPI site 1 area 1, eKERNEL only contacts eLOCATION site 1 area 1. This was a limitation to the scalability of the solution, as there is a one-to-one relationship between eDMSAPI numbering and eLOCATION numbering.

  In release 2.9.10 and in subsequent releases, the eKERNEL contacts every instance of eLOCATION of the same site of the originating eDMSAPI module. This means that an alert from eDMSAPI site 1 area 1 is forwarded to all eLOCATION instances of site 1. As a result, in a environment with three DAP Controller PCs and three associated eLOCATION instances, the location request is sent to all three eLOCATION modules. Each DAP Controller PC then has the opportunity to provide feedback on the last known location of the requested extension or calling party, and return this last known location to eKERNEL.

  Depending on the version of eKERNEL, prior to release 2.9.10 or release 2.9.10 and subsequent releases, the eKERNEL interprets the results from the location request. In release 2.9.10 and newer version, the last use timestamp indication is used to determine the location.

  For these steps the eLOCATION instances must be configured, as shown in the following figure. The configuration of eLOCATION involves a record in eLOCATION for each DAP

Controller. This DAP Controller, 192.168.32.10, is handled by eLOCATION site 1 area 2 as shown in the following figure.

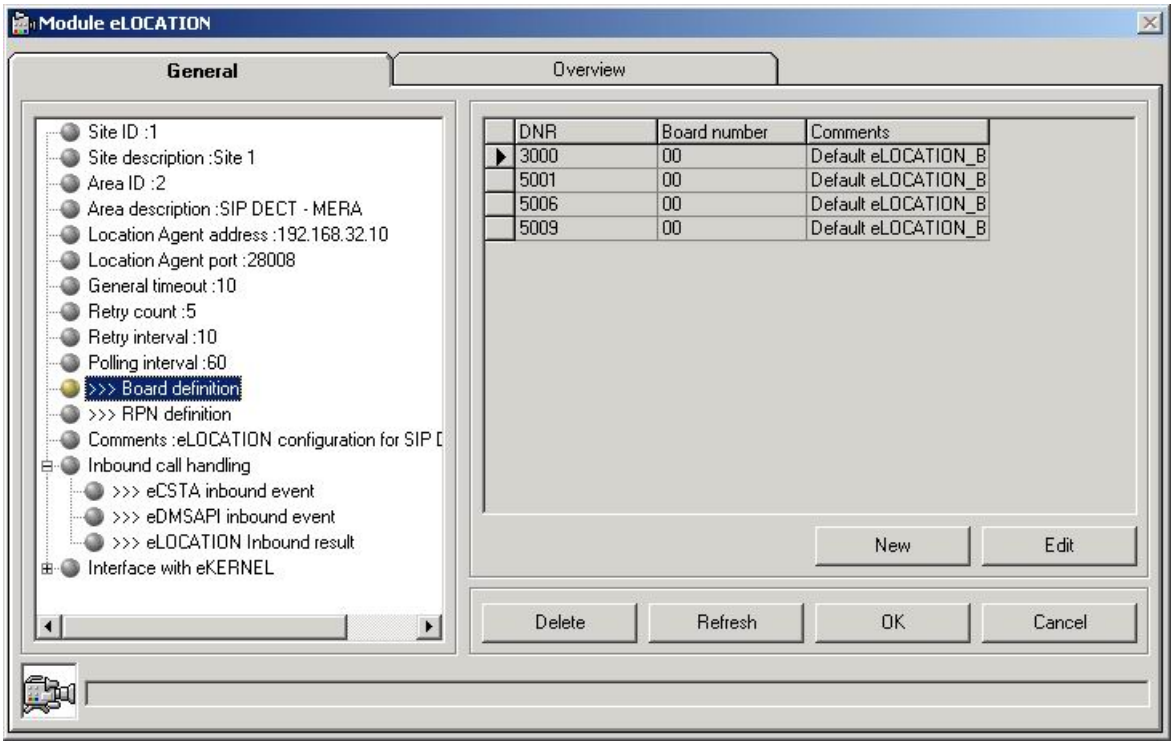


Figure 161: Configuration of eLOCATION

For each DAP Controller and each eLOCATION instance, a number of associated definitions are required in the tables eLOCATION_BOARD, eLOCATION_RPN, and eLOCATION INBOUND_RESULT.

The eLOCATION_BOARD defines the SIP DECT extensions that are registered in the DCC board. See the configuration of the PBX to determine the DECT extensions that are associated with a DAP Controller.

The devices are entered in the system as shown in the following two figures.

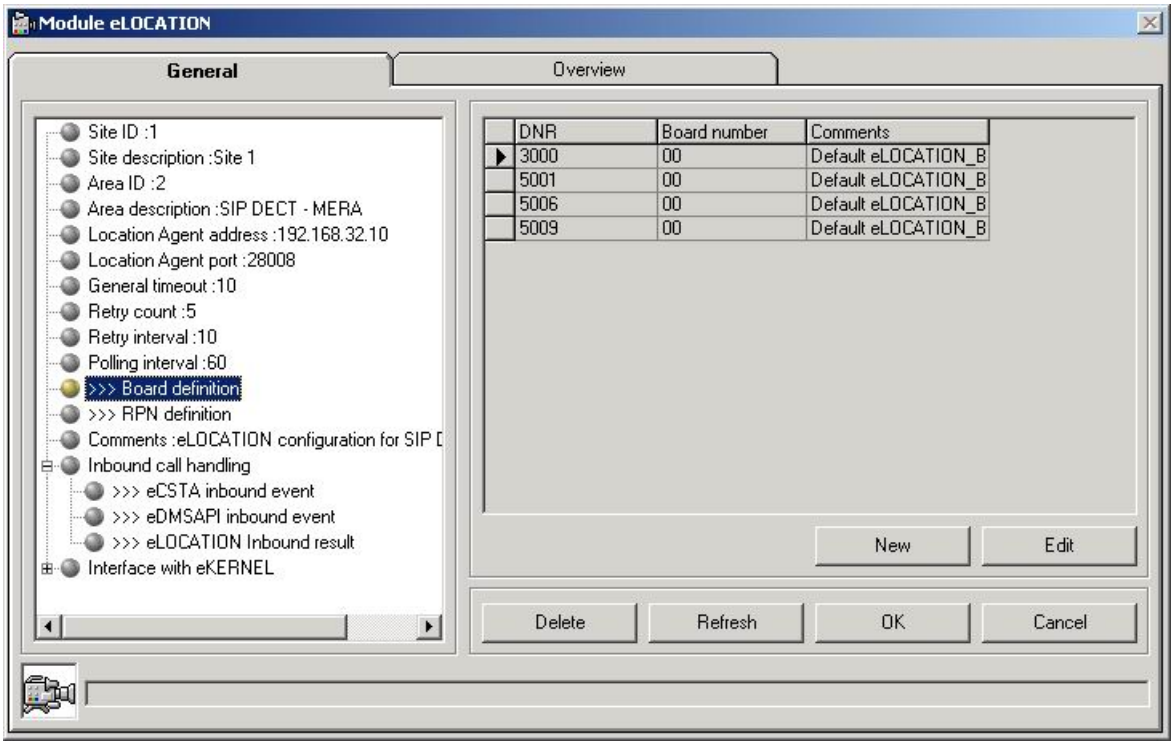


Figure 162: Module eLOCATION board definition

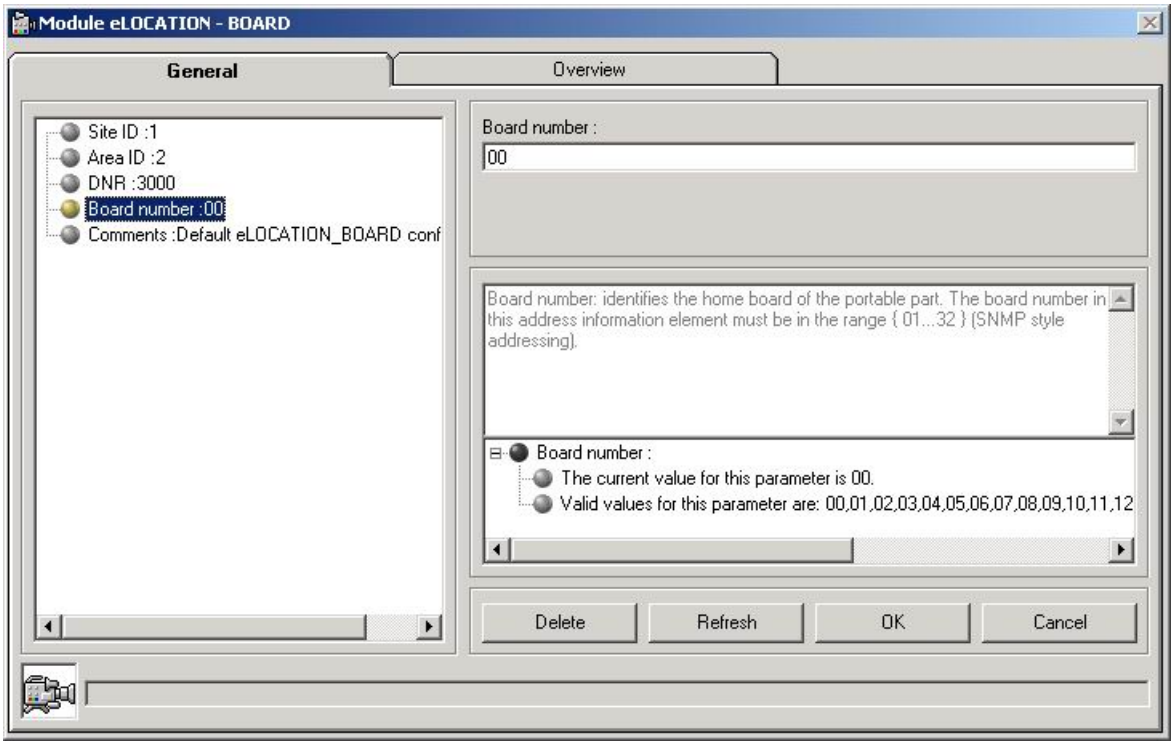


Figure 163: Module eLOCATION board 00 definition

- Next the eLOCATION_RPN table must be populated with records for every RPN known in the DAP Controller configuration. In the following figure, RPN "11" up to RPN "13" is defined. A generic catchall definition is indicated with a "?". For each RPN, we define a description in words that tell the end-user the physical location of the RPN.

  In the sample illustrated in the following figure, the texts are "nearby RPN 011", "nearby RPN 012", but in a real environment the text appears in words rather than technical code, for example Emergency room, Elevator 1, Psychiatric department, and so on.

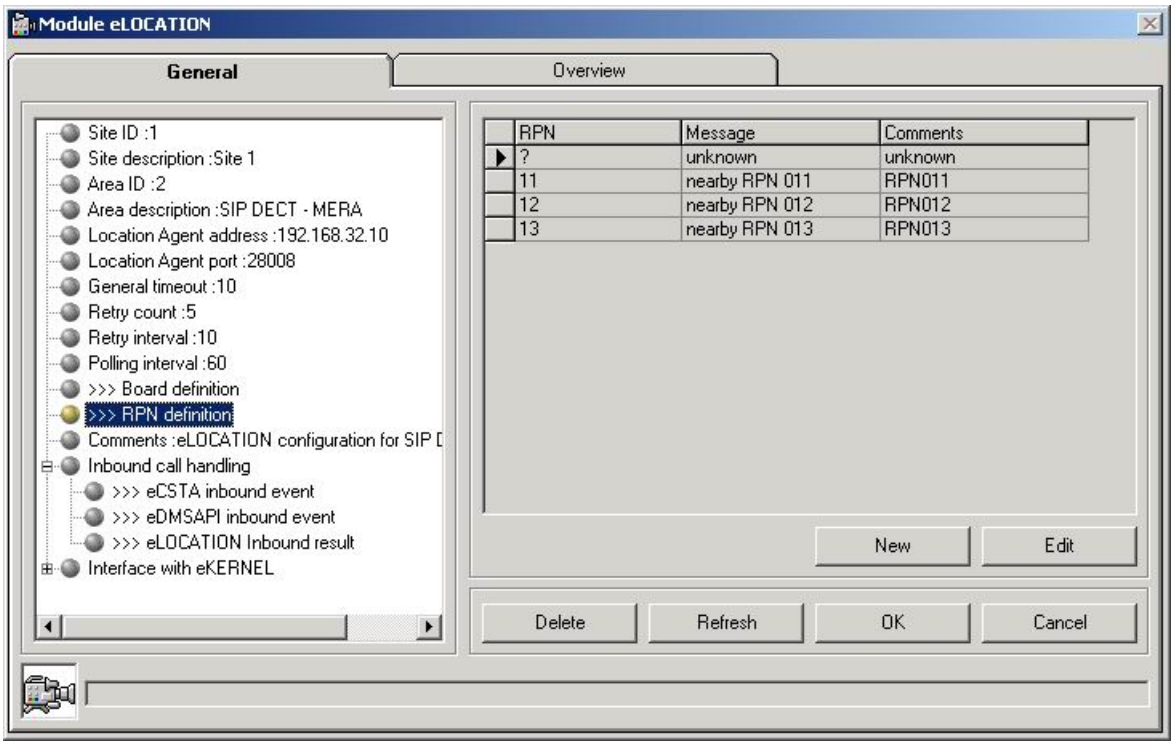


Figure 164: eLOCATION_RPN table

At this stage, you already know the calling number, the called number, and the physical location. With this knowledge, eKERNEL can resume processing. A group of users is alerted with an alarm message in words, not technical code, that contains enough information to let the group of users know the geographical location of the alerting DECT user.

This is done using the eLOCATION_INBOUND_EVENT table, as illustrated in the following figure.

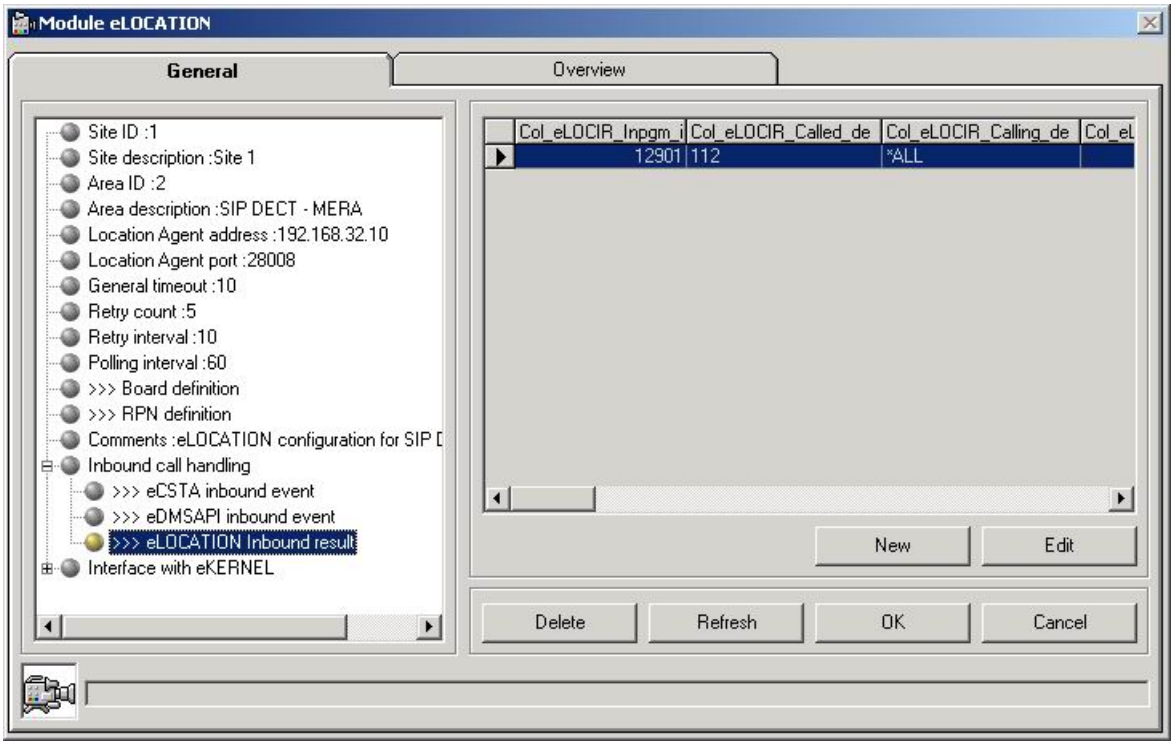


Figure 165: eLOCATION_INBOUND_EVENT table

In the sample in the following figure, alarms from the eDMSAPI input program 12901, site 1, area 2 and all calling devices, that called special extension 112, result in an alarm sent to group 3000.

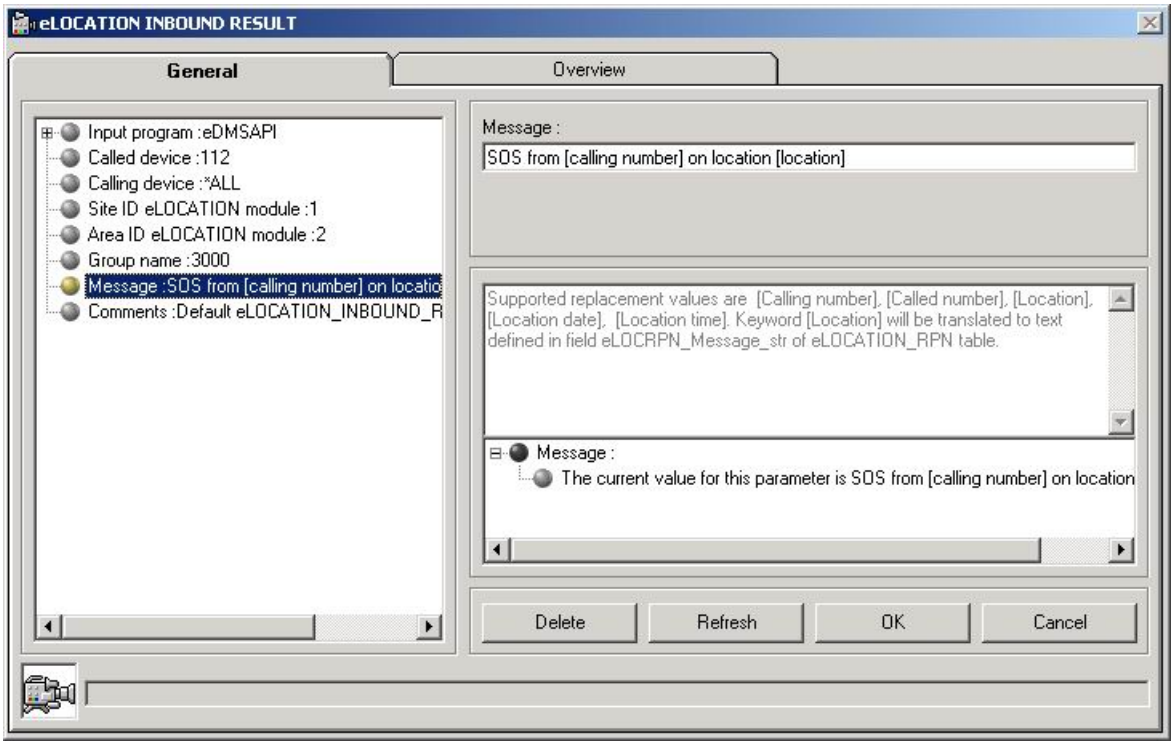


Figure 166: eLOCATION inbound result SOS message from eDMSAPI

The following figure shows an example of a message sent to the group as defined in the Message parameter.

```
SOS from [calling number] on location [location] at [location time]
```

Figure 167: SOS message sent to group

The eKERNEL parses the data stream and performs the appropriate replacements of the replacement values [calling number], [location] and [location time].

Note:

The "Site ID eLOCATION module" equals 1 and the "Area ID eLOCATION module" equals 1 in the example shown in the previous figure. The example indicates that the location request is resolved by the eLOCATION module site 1 and area 1.

In DECT Messenger prior to release 2.9.10, the request is only forwarded to eLOCATION with matching site and area (same for eDMSAPI and eLOCATION). This means when eDMSAPI resides on site 1 area 1, only the configuration records in eLOCATION_INBOUND_RESULT with reference to site 1 and area 1 are used.

In DECT Messenger release 2.9.10 and in subsequent releases, these requests are published to all available eLOCATION modules residing on the same site, and the most recent location is used. In this environment, the matching definitions for the available areas must also be defined.

The possibility of using a "visual DNR" to a device in Messenger (new field DEV_Visual_dnr_str in table eKERNEL_DEVICE for the replacement value [calling

number] is introduced in release 3.0. In release 3.0 and subsequent releases, when the system configuration configures a device with a visual DNR, this DNR is used to format a message when it contains [Calling number]. The end-used is confronted with the visual DNR instead of the device id.

Chapter 24: Module - eSMS

Important:

Due to the ongoing development of the DECT Messenger product suite, some modules that provide additional functionality may become available after the initial release of DECT Messenger 4.0.

The following modules are described in this document but are not available at initial General Availability.

- eFR
- eLICENSE
- eLOCATION
- eSMS
- eSNMP
- eVBVOICE

The eFR module is an add-on module and is licensed separately through the eLICENSE module. Some of the modules listed in this attention box are available only on a site-specific basis.

Architecture

The eSMS module works with the SMS_service module. Together, these modules provide the ability to send outbound SMS messages from DECT Messenger to mobile GSM phones. Incoming SMS messages that originate from mobile GSM phones can also be processed to confirm alarms.

Siemens TC35i module

The SMS_service module currently only supports the Siemens TC35i module. Verification of compatibility of other SMS boxes can be handled on a project-by-project basis.

Some other modules are compatible and other modules show intermediate problems of incompatibility issues.

In addition a Siemens TC35i module, you need the following items.

- a serial cable to connect the TC35i with the PC, included in TC35i
- a power unit and cable for the TC35i, included in TC35i

- an antenna for the TC35i, included in TC35i
- a SIM card of mobile provider of choice with a PIN and a PUK code

Follow the steps in the next procedure to install the Siemens TC35i module.

Installing the Siemens TC35i module

1. Install the SIM card in a regular mobile phone.
2. Power on the mobile phone.
3. Refer to the SIM card information and obtain the valid PIN and PUK codes.

   You usually need the pincode at startup.
4. Use the menu options of your regular mobile phone to disable the PIN code check at power-up.

   To verify that this step is complete, switch the regular mobile phone off and on. The mobile phone no longer asks for a PIN code and immediately becomes operational.

   Important:
   Do not continue unless this step completed.
5. Ensure the location where you plan to install the Siemens TC35i has sufficient coverage of your mobile provider.

   The building environment can prevent good reception of a mobile operator signal. When the reception is not optimal, consider moving the antenna and box to a location with better reception. Another solution to less than optimal reception on your mobile phone is moving DECT Messenger to another place. You can also consider a distributed setup running the eSMS and SMS_service on a different PC.
6. Send and receive an SMS message manually to ensure operator connectivity for SMS messaging runs properly.
7. Remove the SIM from your regular mobile phone, and insert it in the Siemens TC35i module.

   Refer to the documentation that comes with the Siemens TC35i module for installation details.

SMS service

When you install eSMS, the eSMS module, the SMS_service module, SMS_service.ini, and SMS_service.exe are installed in the directory: C:\SOPHO Messenger@Net\Exe.

The SMS_service module is a separate process and communicates with the GSM box.

SMS_service.exe and SMS_service.ini are two resources related to SMS_service functionality.

Configure SMS_service through the .INI file. The default values suit most environments. You can change the parameter COM Port if COM1 is not available. Change the value according to your environment, for example specify 3 if COM3 is available.

In the sample file illustrated in Figure 168: Default configuration section on page 225, there is a single INI file. There can be more than one instance of SMS_service with multiple Siemens TC35i boxes. In most cases, only one Siemens TC35i is used and only a single instance of eSMS and SMS_service is used.

When the service is started with the following shortcut, the default configuration section is used.

```
"C:\SOPHO Messenger@Net\Exe\SMS_service.exe"
```

Figure 168: Default configuration section

The default configuration is configured in INI file with the statements shown in the next figure.

```
[Default]
System = Module 1
```

Figure 169: INI file

The statement in Figure 169: INI file on page 225 indicates that in absence of a /System:xxxx in the shortcut, the default system is used. In the sample configuration in the following figure, the configuration Module 1 is set as a default system. This means only Module 1 is used.

```
[Module 1]
Manufacturer            = SIEMENS
Model                   = TC35i
COM port                = 1
COM settings            = 9600,n,8,1
Mode                    = Text
Log days                = 7
Queue depth             = 10
Local                   = +32473897171
Remote                  = +32473897171
Outbound HTTP port      = 29082
Outbound HTTP sockets   = 10
Outbound XML port       = 29081
Outbound XML sockets    = 10
Inbound interval        = 10
Inbound XML address     = 127.0.0.1
Inbound XML port        = 29080
Inbound signatures      = M, Msg, Messenger, Msg@Net, Messenger@Net
```

Figure 170: Sample configuration

Special circumstances exist where two instances of eSMS and SMS_service and TC35i can be requested.

For example, if you have a need for high-level redundancy, you can use SIM cards from two different mobile providers. If one box of SIM card fails, you can use the other box. You setup a configuration to first send to devices on eSMS area 1, mobile provider 1, and if NACK occurs, you use alternative devices defined on eSMS area 2, mobile provider 2. You alter the shortcuts referring to the desired configuration section shown in the following figure.

```
"C:\SOPHO Messenger@Net\Exe\SMS_service.exe" /System=Module 1
```

```
"C:\SOPHO Messenger@Net\Exe\SMS_service.exe" /System=Module 1
```

Figure 171: Altered shortcuts

The following figure shows the default configuration.

Figure 172: Default configuration

At startup, initialization takes place. The SMS_service contacts the Siemens TC35i module and initializes the system. A message window appears that states "Initialization completed normally". The SMS_service is then operational for sending outbound messages received from eSMS.

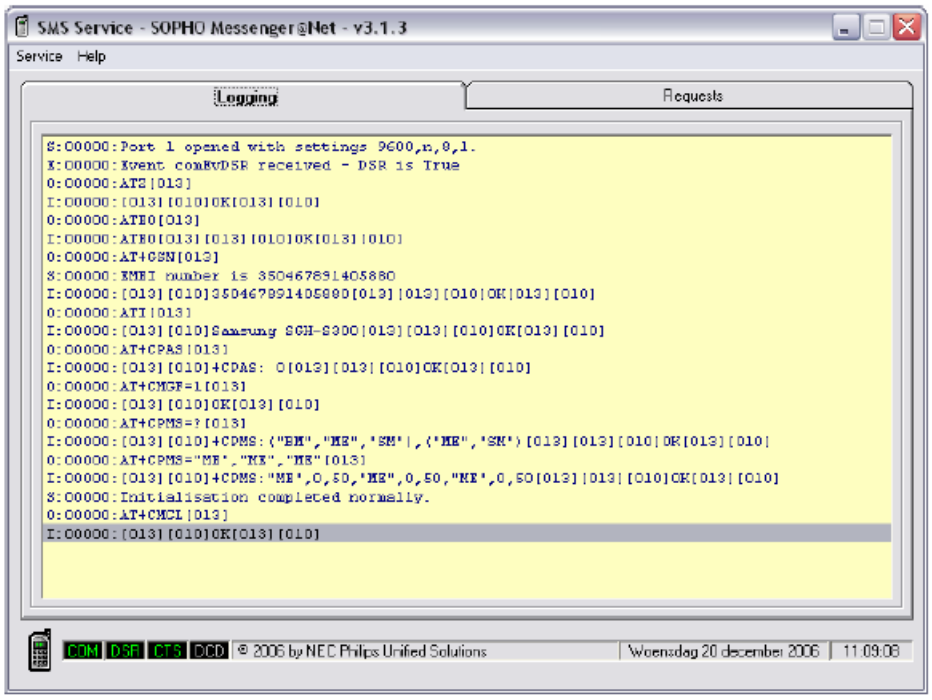


Figure 173: SMS_service operational

The module polls the box to find out if inbound messages are received from external mobile phones through the mobile network.

The logging file can feature the AT+CMGL loop condition during the idle time.

eSMS module

The support for SMS messaging is implemented with different processes.

The eKERNEL process integrates with eSMS process, and eSMS process integrates with SMS_service process.

This design is similar to other modules, such as eKERNEL – eDMSAPI – CSTA_service or eKERNEL – eCSTA – CSTA_service.

At startup, eSMS contacts the eKERNEL to request its configuration. You configure the IP address and port in the shortcut. When parameters are absent, you are prompted for startup values.

Figure 174: Startup values prompt

The command line parameters are shown in the following figure.

```
/Site:1 /eKernel address:*LOCAL /eKernel port:3107 /Log drive:C
```

Figure 175: Command line parameters

The module eKERNEL responds to the <cfgrqs> with <cfgrpy>. This configuration is retrieved from the Messenger_CFG database table.

In native mode, the configuration is retrieved from the eSMS table. This table contains, for example the IP address and the port number used to connect to the SMS_service process. This SMS_service is a separate program that communicates with the SMS box, and can run on a local or distributed PC. The trace of the configuration exchange is shown in the following figure.

```
S:INF:Application eSMS - SOPHO Messenger@Net - v4.0.0 started with
parameters /Site:1 /eKernel address:*LOCAL /eKernel port:3117 /Log
drive:C

S:INF:TCP local port 01158 connected with remote port 03117 (eKERNEL)

O:TCP:<xml><cfgrqs><appl>eSMS</appl><site>1</site></cfgrqs></xml>

I:TCP:<xml><cfgrpy><sms_address>127.0.0.1</sms_address><sms_outbound_por
t>29081</sms_outbound_port><sms_inbound_port>29080</sms_inbound_port><ve
rsion>2.9.10</version><log_path>C:\SOPHO
Messenger@net</log_path><log_days>14</log_days></cfgrpy></xml>
```

Figure 176: Configuration exchange

At startup, the main screen appears, shown in the following figure.

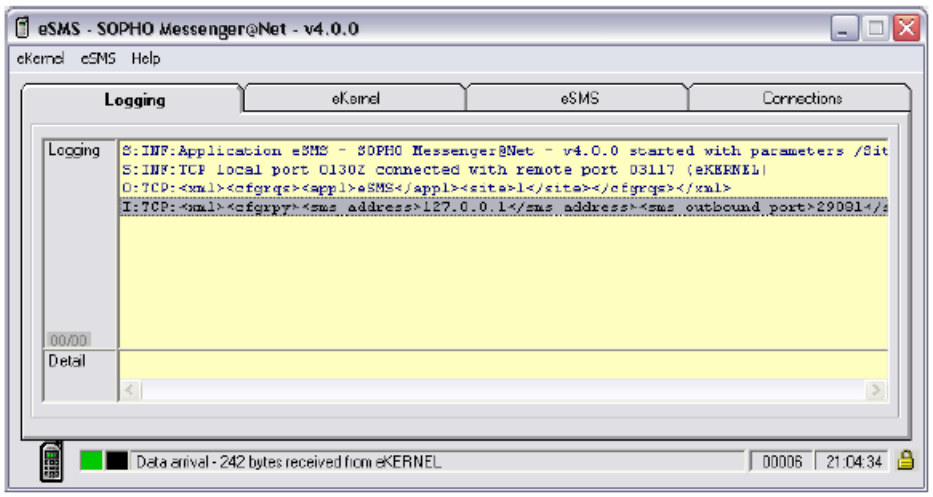


Figure 177: eSMS Main screen

The section on the left features outbound messaging and the section on the right features inbound messaging.

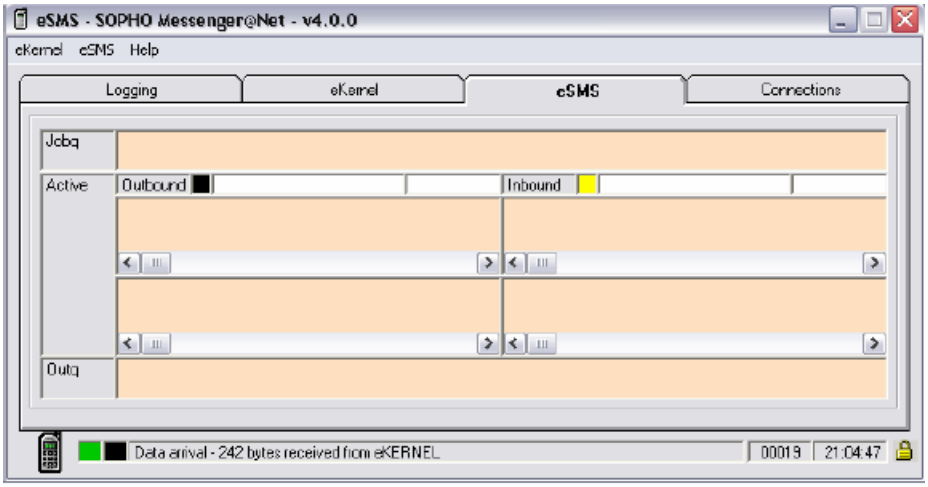


Figure 178: Inbound and outbound messaging

The tab connections shows the details of the configuration received from the eKERNEL.

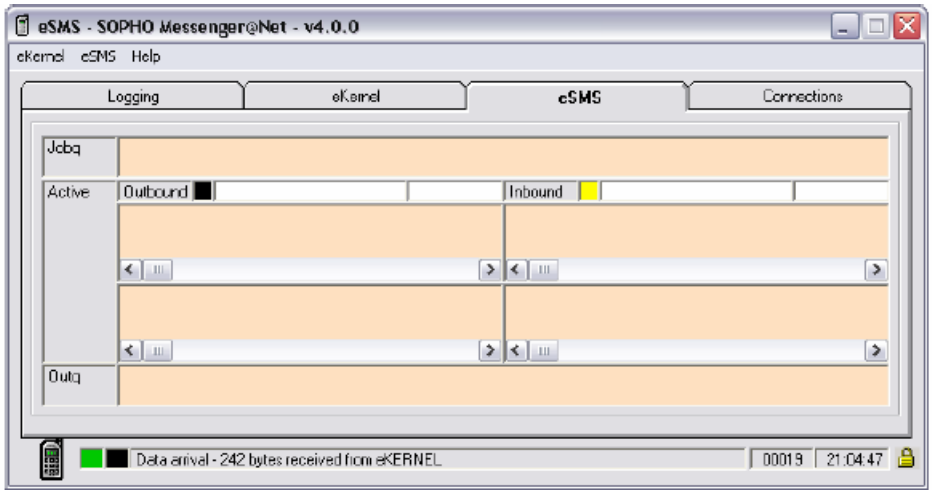


Figure 179: Configuration details in tab connections

Outbound messaging

The eSMS module can send outbound SMS messages to mobile phones. It connects to the mobile world through an SMS box, controlled by the SMS_service process.

The eSMS module contacts the SMS_service process during the messaging delivery process. The message delivery process bidirectional. The eSMS sends a request to the SMS_service and the SMS_service sends a reply to the eSMS module.

The following figure shows a sample message is sent to a mobile phone.

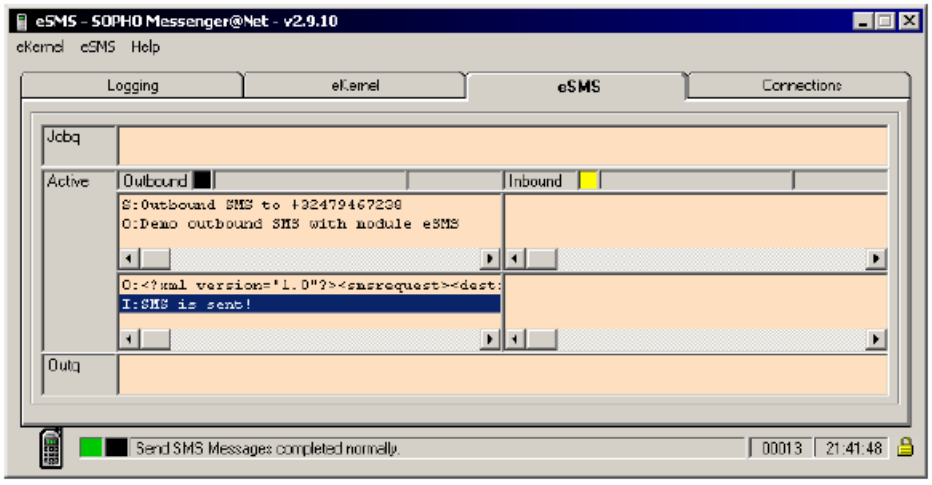


Figure 180: Example of an outbound message to a mobile phone

More details on this dialogue are listed under the Logging tab.

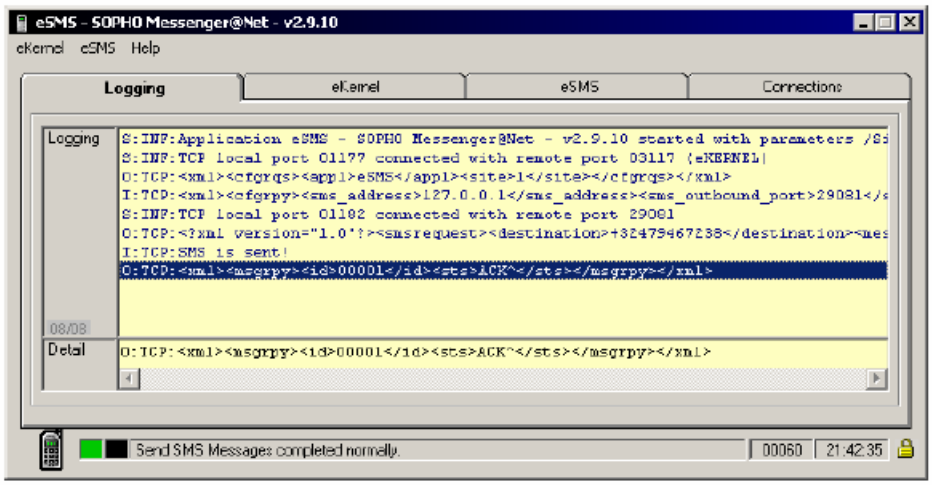


Figure 181: Details under logging tab

The logging files provide more details on the data exchange.

```
S:INF:TCP local port 01182 connected with remote port 29081
O:TCP:<?xml version="1.0"?><smsrequest><destination>+32479467238
</destination><message>Demo outbound SMS with module eSMS
</message></smsrequest>

I:TCP:SMS is sent!

O:TCP:<xml><msgrpy><id>00001</id><sts>ACK^</sts></msgrpy></xml>
```

Figure 182: Logging files

When message delivery is successful, positive feedback is returned to eKERNEL. This XML datastream contains the <ACK> characters.

When message delivery fails, a negative response is sent to eKERNEL, as shown in the following figure. The window in the following figure shows that the eSMS cannot connect to the SMS_service due to a network error.

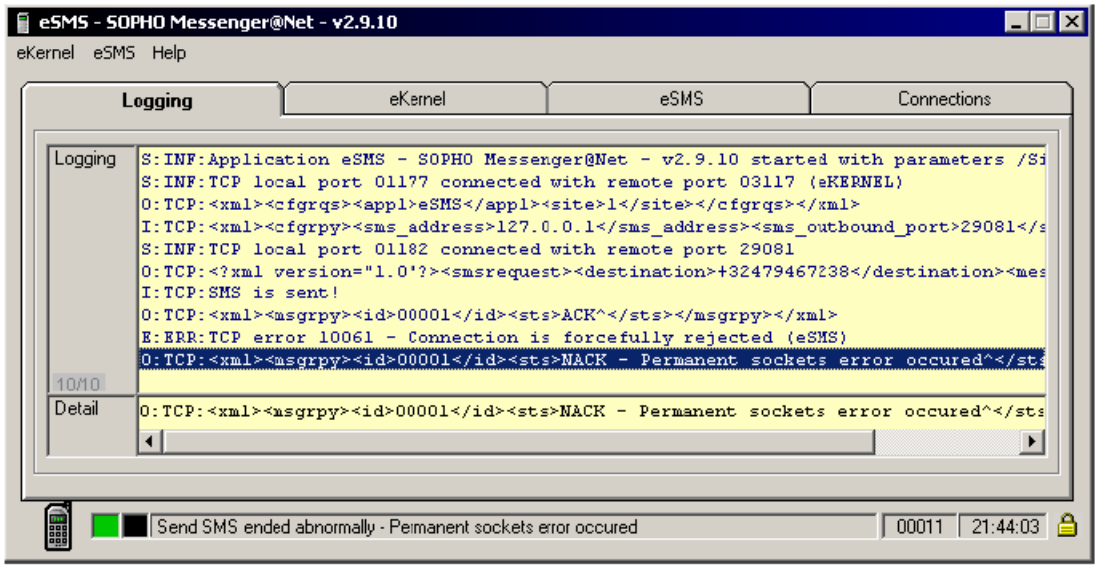


Figure 183: Example of message delivery failure

Inbound messaging

The eSMS module also handles inbound SMS messages. These messages are received on the GSM box, and transferred to the eSMS module to be processed in DECT Messenger by eKERNEL.

For this purpose, additional tables eSMS_INBOUND and eSMS_INBOUND result are added to the Messenger_CFG database.

The eSMS module communicates with the SMS_service through a separate TCP connection. The eSMS module acts as the TCP Server and the SMS_service module acts as the TCP Client. The eSMS is listening to a port, for example 29080, for inbound messages requests from mobile phones.

In the current release of eSMS, the inbound functionality can confirm alarms, either based upon the calling line identifier, or mobile phone extension number, or based upon a pincode, provided in the SMS message.

The following figure illustrates the reception of an inbound SMS message. The message confirms all alarms for the devices that have pincode 238.

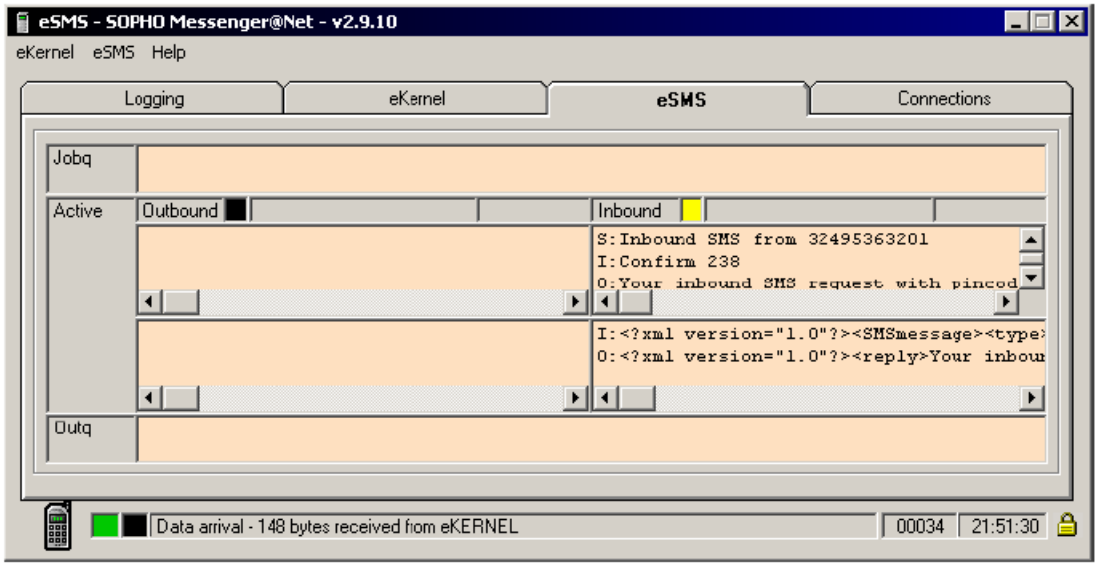


Figure 184: Reception of an inbound SMS message

More details on this data exchange are given under the logging tab, as shown in the following figure.

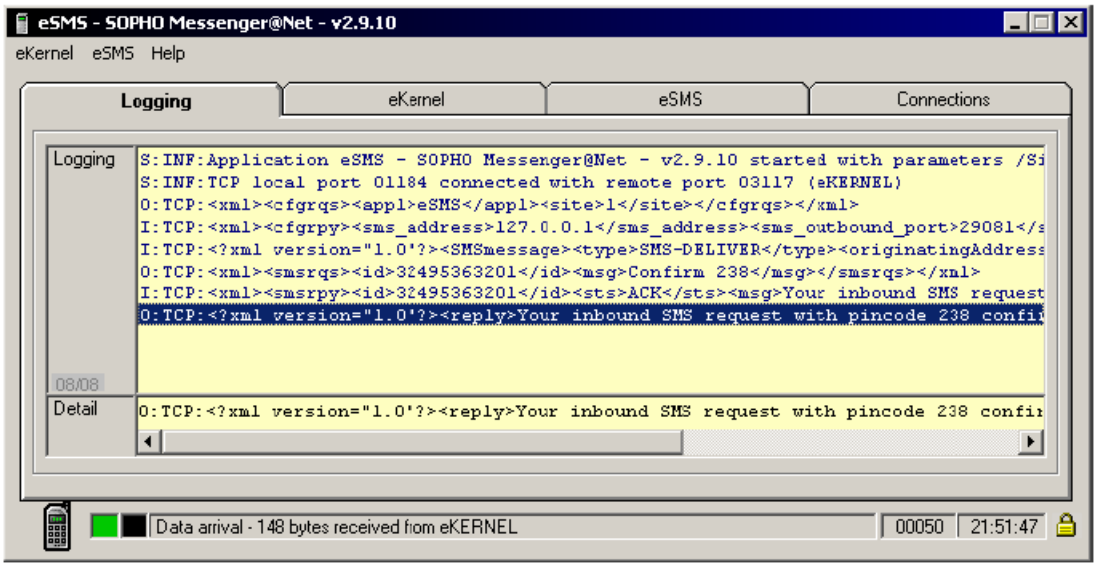


Figure 185: Logging tab

The logging files provide more details on the data exchange, as shown in the following figure.

```
I:TCP:<?xml version="1.0"?><SMSmessage><type>SMS-
DELIVER</type><originatingAddress>32495363201</originatingAddress><userD
ataLength>11</userDataLength><userData>Confirm
238</userData></SMSmessage>

O:TCP:<xml><smsrqs><id>32495363201</id><msg>Confirm
238</msg></smsrqs></xml>

I:TCP:<xml><smsrpy><id>32495363201</id><sts>ACK</sts><msg>Your inbound
SMS request with pincode 238 confirmed 0 alarms on 5
devices</msg></smsrpy></xml>

O:TCP:<?xml version="1.0"?><reply>Your inbound SMS request with pincode
238 confirmed 0 alarms on 5 devices</reply><result>OK</result>
```

Figure 186: Logging files

Configuration

The eSMS module is configured through the eCONFIG module. The eSMS module requires eCONFIG 2.9.10 or above.

The following eight figures illustrate the configuration process for both outbound and inbound messages.

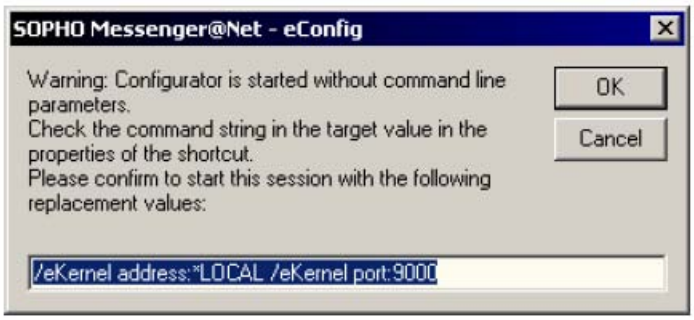


Figure 187: eCONFIG configurator

Figure 188: Sign on to DECT Messenger configurator

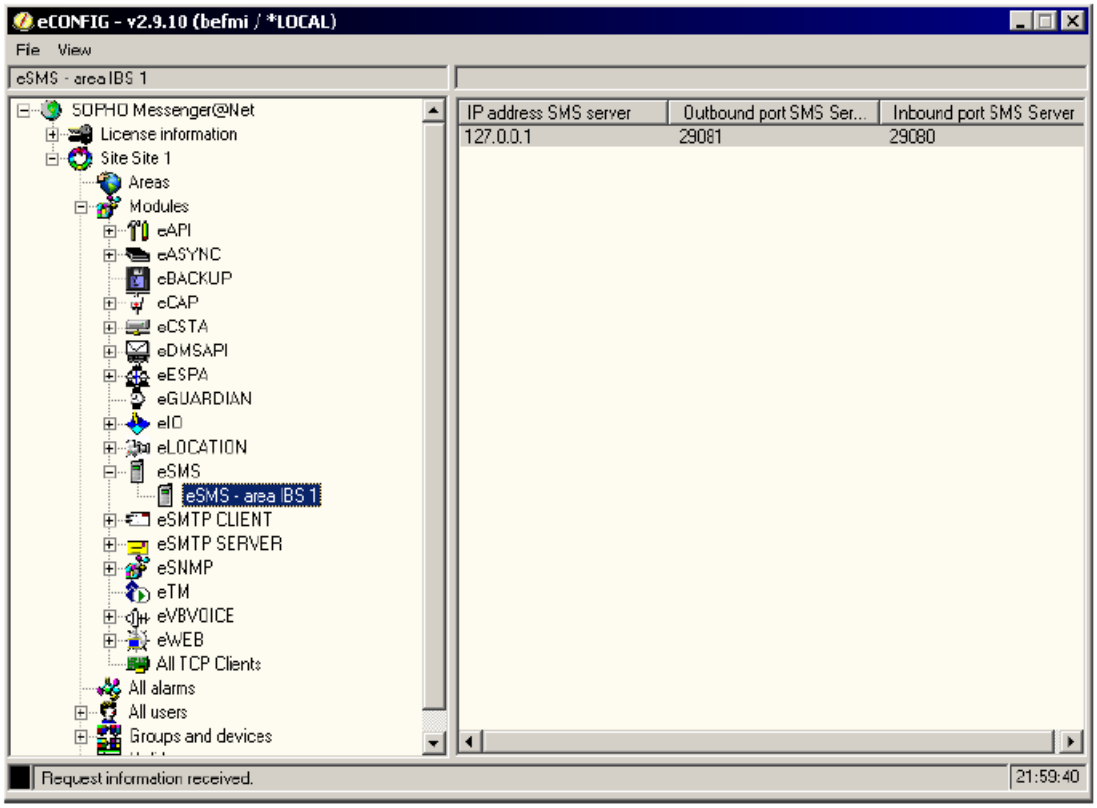


Figure 189: eSMS area IBS 1

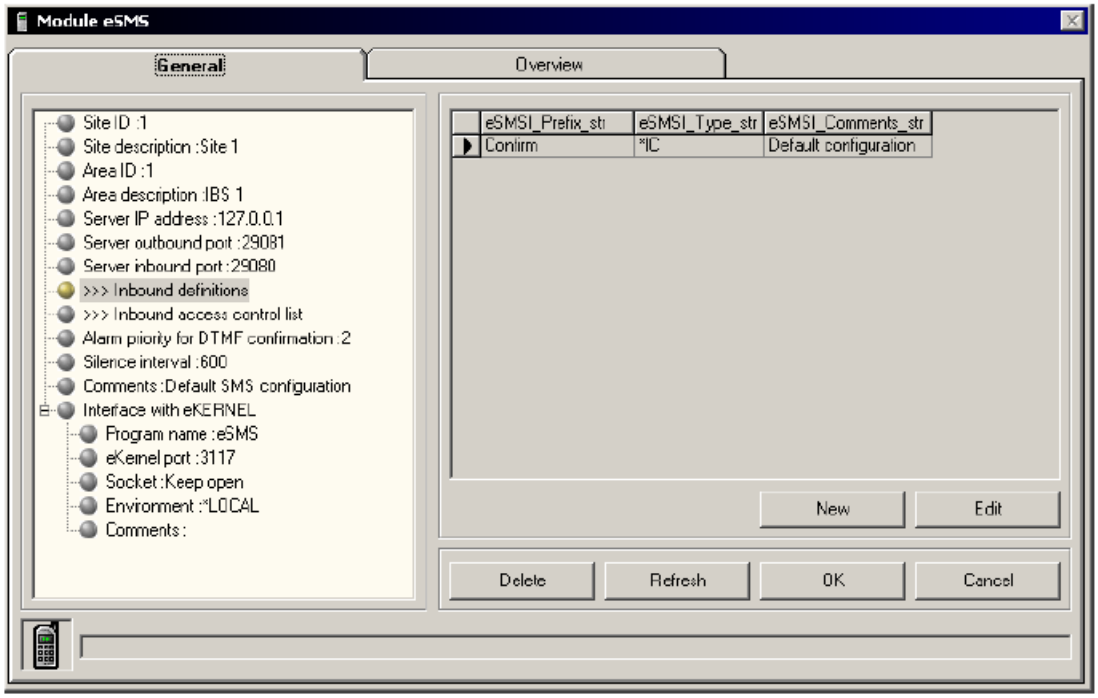


Figure 190: Module eSMS inbound definitions

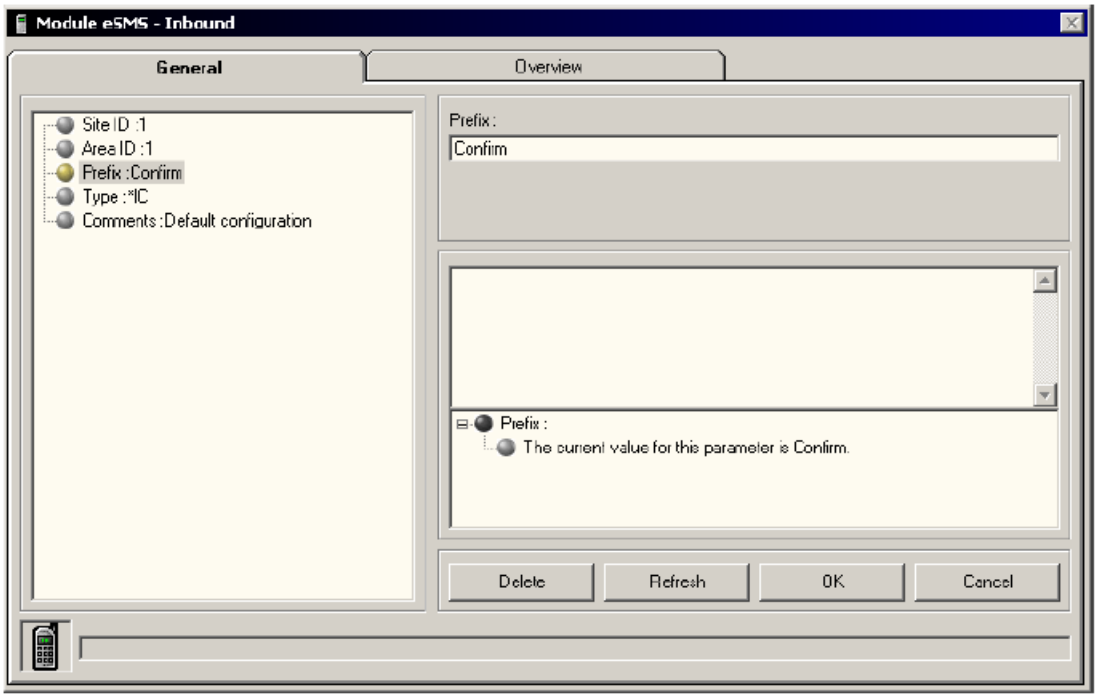


Figure 191: Module eSMS - Inbound

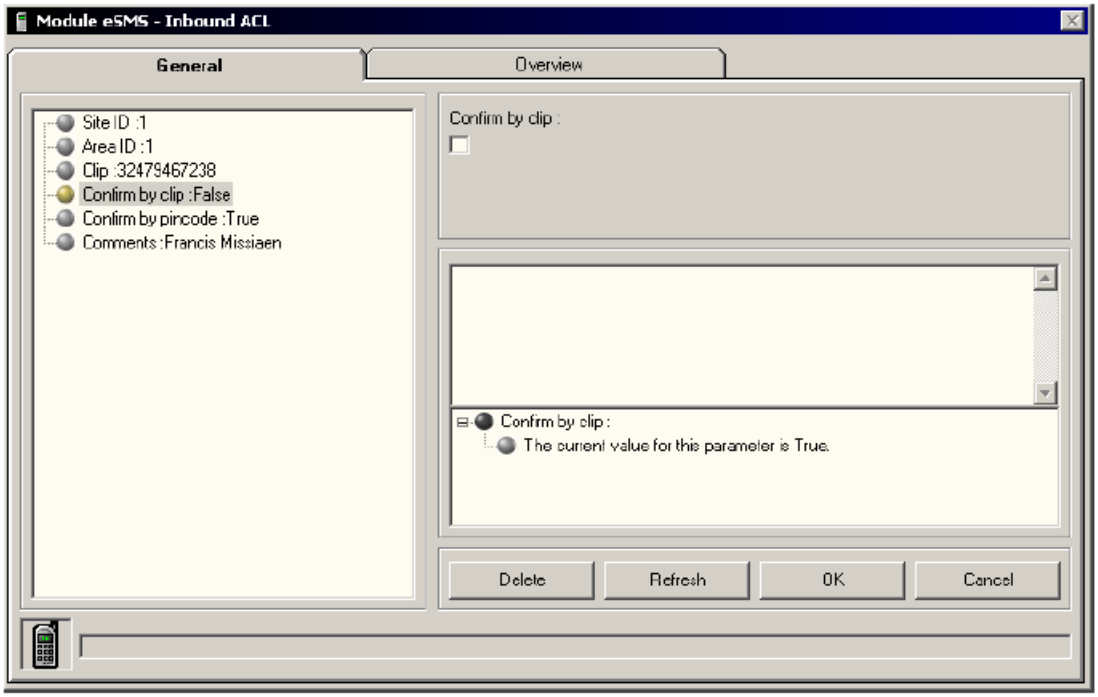


Figure 192: Module eSMS - Inbound ACL

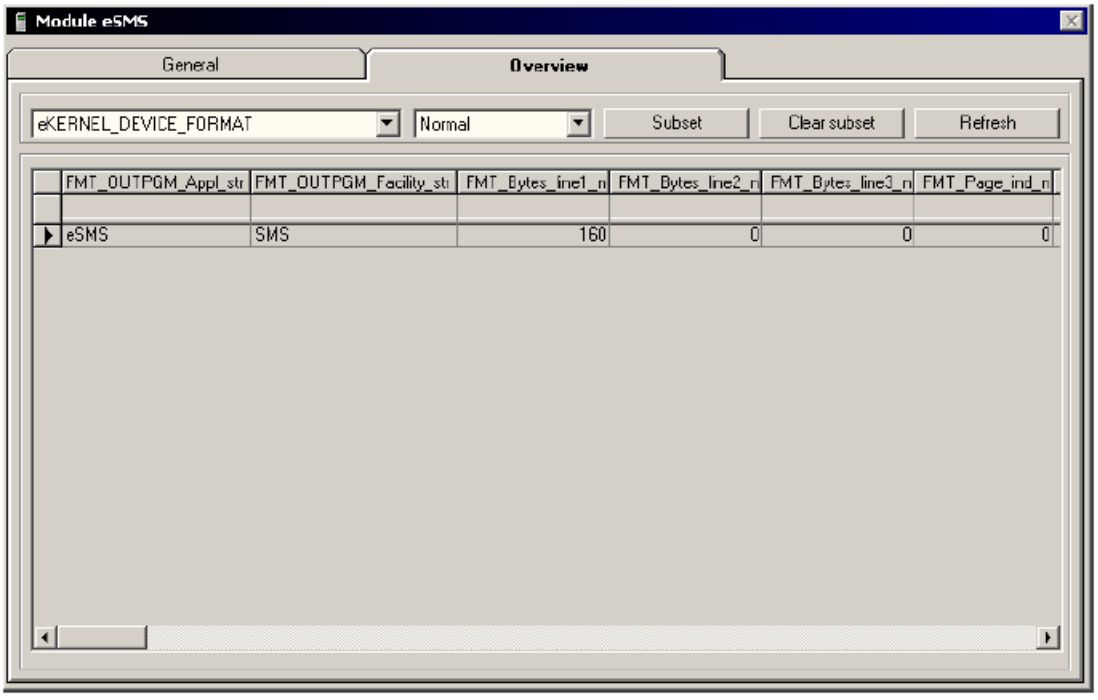


Figure 193: Module eSMS eKERNEL_DEVICE_FORMAT

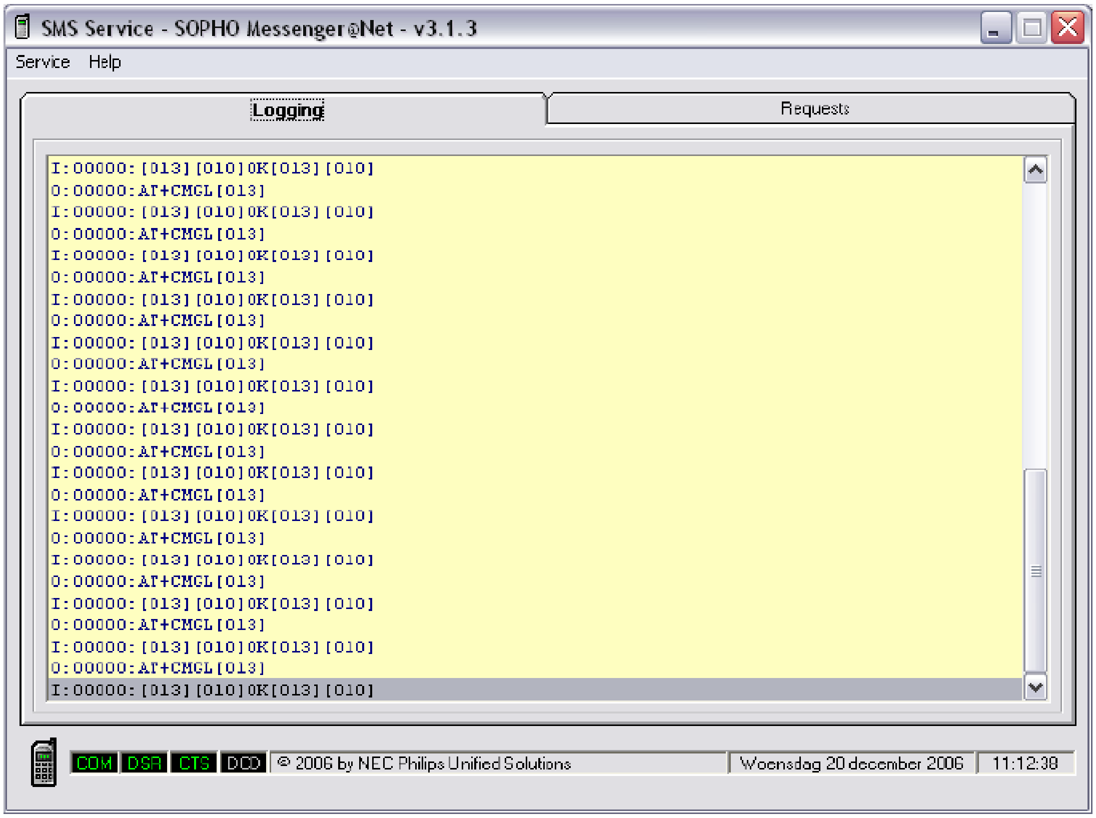


Figure 194: SMS Service logging tab

Web interface

When eSMS is configured, the new Web Administrator interface features a window where you can send an SMS message from the web interface. This window retrieves all defined devices from the eKERNEL_DEVICE table configured for the eSMS output program.

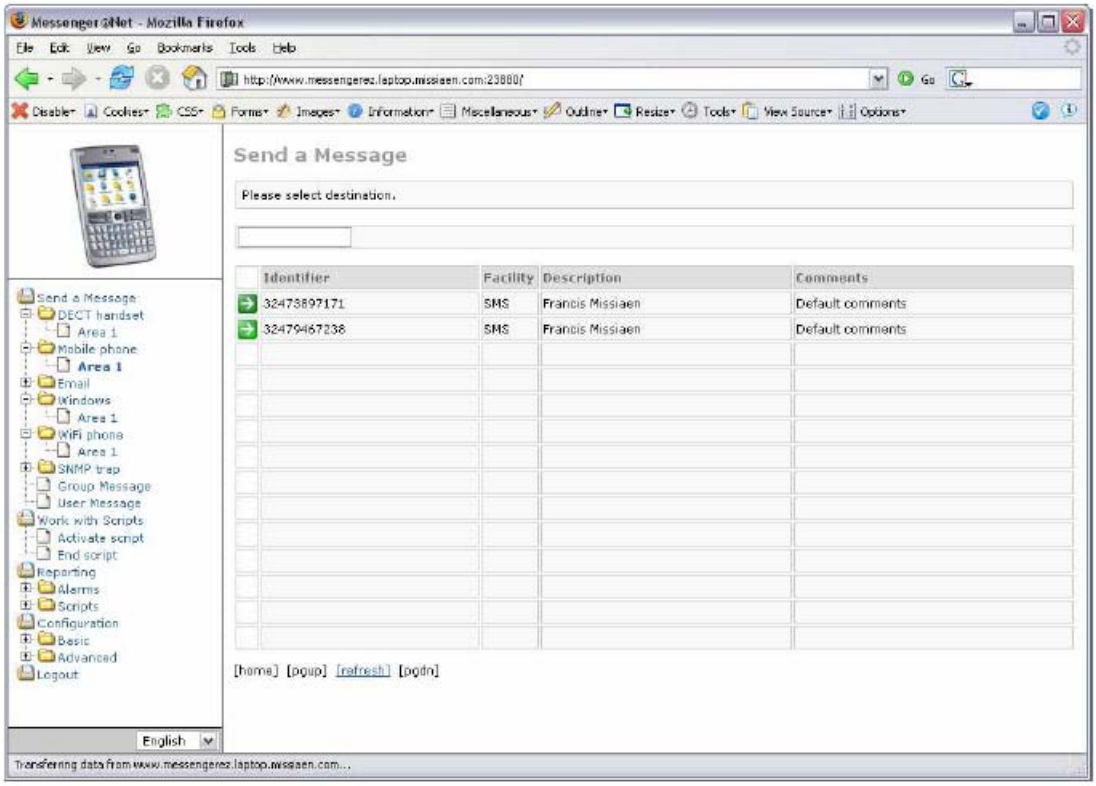


Figure 195: Select destination

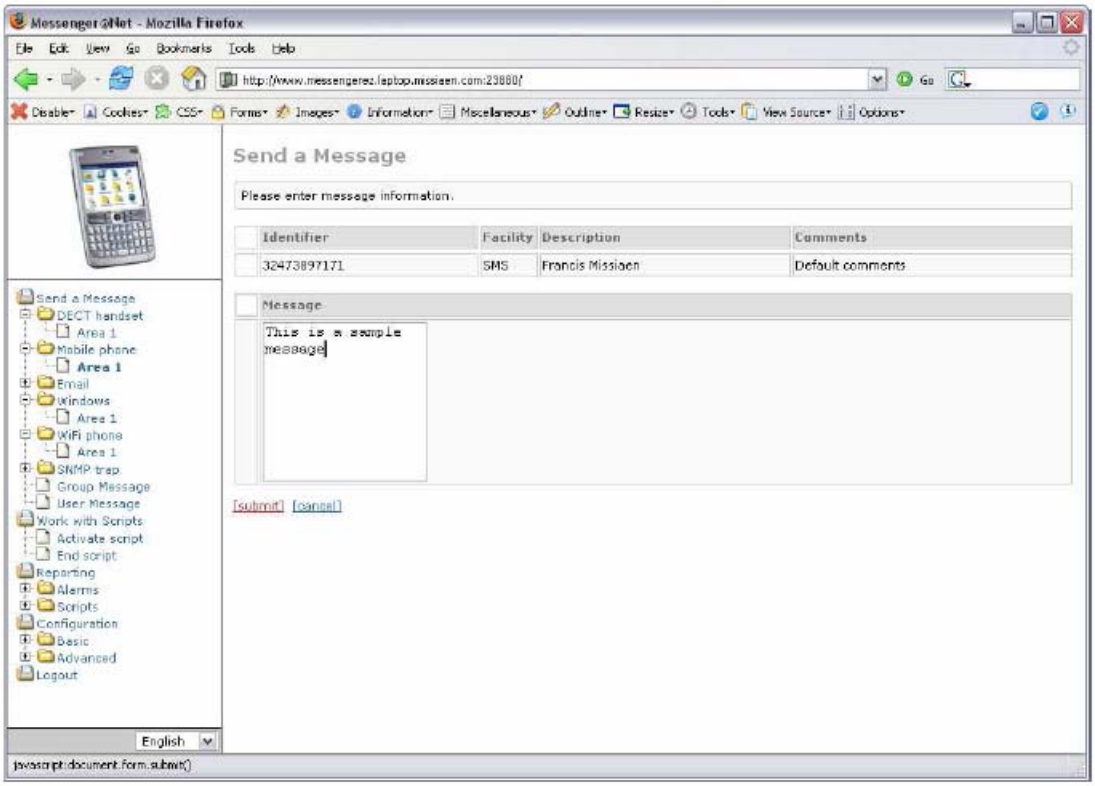


Figure 196: Enter message information

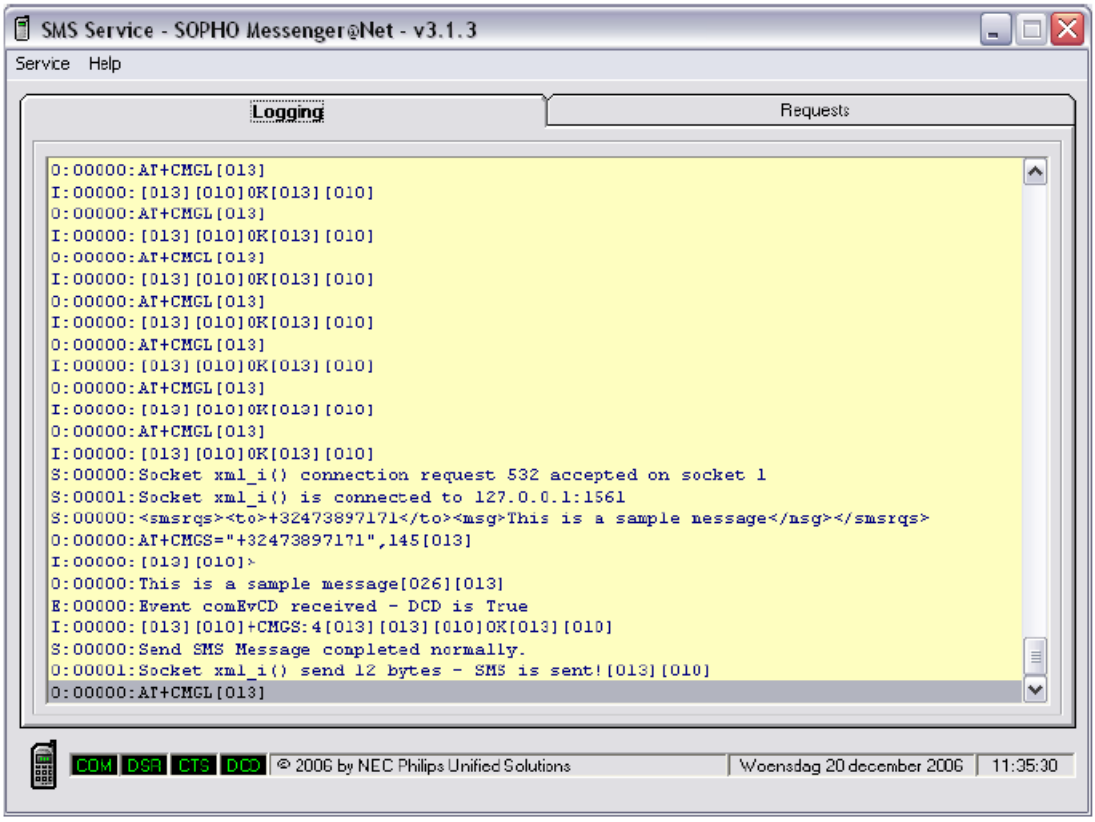


Figure 197: Logging tab

Index

F

G

H

I

L

M

N

O