CMX V6.0B (Solaris)

Operation and Administration

Edition June 2005

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion of this manual. Your feedback helps us optimize our documentation to suit your individual needs.

Fax forms for sending us your comments are included in the back of the manual.

There you will also find the addresses of the relevant User Documentation Department.

Certified documentation according DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Copyright © 2005 Fujitsu Siemens Computers GmbH.

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

This manual is printed on paper treated with chlorine-free bleach.

Continued ▶

SNMP subagent for CMX

Using TLI applications

Reference section and index

Contents

1 Introduction

The Communications Manager for UNIX Systems (CMX) is the basic product for Solaris communication software. Together with the Communication Control Programs (CCPs), CMX implements an open communication system. CMX transmits data between different transport systems and program interfaces, and enables program-to-program communication regardless of the transport systems used. The same applies to your own applications, which you can create using the available program interfaces (ICMX, XTI).

With CMX and the appropriate Communication Control Programs (CCP), you can use all the usual communication services:

Network Access Services offer you access to

– wide area networks (WANs) such as PSDN, CSDN, PSTN, Frame Relay

– Ethernet, Fast Ethernet, Gigabit-Ethernet, Token Ring, and FDDI-LANs

– ISDN via S_0 and S_2 connections

End-to-End Services support

– RFC 1006 via TCP/IP, OSI, and TRANSDATA NEA protocols in the transport system

– full integration of your Solaris system in SNA networks

– X.25 linking of systems and terminals via WAN/ISDN without a transport protocol

The following diagram illustrates the product structure of Solaris communication together with the subnetworks served by the CCP products.

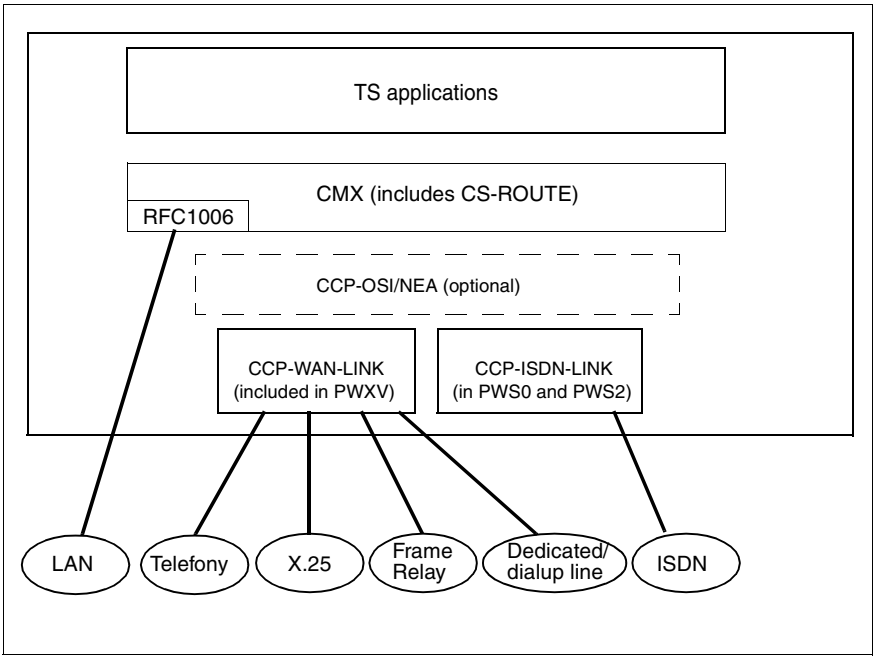


Figure 1: Overview of the product structure for Solaris communication

CMX offers all the necessary functions for configuring and managing your communication resources. All administration, maintenance, and diagnostic tasks can be performed either via the command interface or using the menus. Root authorization is no longer necessary for these operations. All tasks relating to administration, maintenance and diagnostics can be conducted under the user role *cmxadm*.

The transport profile TCP/IP-RFC1006 is a component of the CMX product.

1.1 Target group

This manual is intended for system administrators. In order to work with CMX, you must be familiar with the Solaris operating system. Knowledge of the principles and methods of data communications will also prove helpful, especially with regard to the OSI Reference Model as standardized in ISO7498.

1.2 Summary of contents

Two manuals provide a comprehensive description of the CMX product:

– CMX “Operation and Administration” for system administrators

– CMX “Programming Applications” for programmers of TS applications (TS = Transport Service) who use the CMX program interfaces

Chapter 2 describes the architecture of the Solaris communication software. Here you will find an overview of the available CCP products and their functionality as well as important basic information on CMX system administration.

Chapter 3 describes the user role *cmxadm*.

Chapter 4 contains basic information on addressing using the components TNS and FSS. You will need this knowledge to be able to configure your computer network.

A description of the installation procedure is given in chapter 5.

Chapter 6 describes the configuration procedure and provides instructions for operation and administration by the character-oriented user interface CMXCUI.

Chapter 7 describes the input format and file format of the CMX components TNS and FSS.

Chapter 8 describes the web-based administration of CMX. It describes the installation, configuration and starting of the CMXwca package. This package enables access to CMX administration from the web-based user interface and also provides troubleshooting information and instructions on security issues.

Chapter 9 contains address and command information for the transport service RFC1006 via TCP/IP which is supplied as a component of CMX.

Chapter 10 describes the commands involved in CMX system administration and their syntax.

Chapter 11 describes an additional component of CMX, namely the CMX agent for remote administration of CMX via SNMP. It provides information about the function and operation of the CMX agent.

Chapter 12 describes operation of TLI applications via CMX.

The “Programming Applications” manual describes the program interfaces of CMX, i.e. all program calls you require for developing your own applications.

Man pages

Included in the CMX product you will find the *Online Man Pages*. In addition to the commands described in the manual, these man pages document further expert commands which you may find useful for system administration. You will come across occasional references to these man pages in this manual.

References to the Release Notice

Special features specific to the operating system are not described in this manual but in the Release Notice (product-specific README file).

References to other publications

The text contains references to other publications in the form “see ’title of the manual’ manual [n]”, where n is a number. The “Related publications” section contains a list of the corresponding publications by number, together with a brief overview of the contents.

1.3 Changes to the previous version

The following changes have been made to the previous version this manual (CMX V6.0A (Solaris), Edition June 2003).

CMX is Live Upgrade-capable

CMX can now be installed in an alternative boot environment during normal operation. Details are provided in the chapter “Installation and startup” on page 53.

Extended web-based administration

Web-based administration is now implemented by means of ServerView. This means that the CMX communication servers can be administered either via a LAN console/SMC or via ServerView-based Windows administration clients.

It is now also possible to encrypt communication between administration client and communication server by means of SSL/TLS.

For more information refer to the chapter “Web-based CMX administration” on page 137.

New and extended CMX administration commands

The following commands have been added to the command interface.

– **cmxconf**
Checks the configuration of a CMX application, see page 231.

– **cmxstat**
Outputs TSP-specific status information, see page 283.

– **cmxtrc**
Switches traces on/off for a transport system, see page 287.

The following commands have been modified.

- **cmxprod**
Outputs product information separately for boot environment and root directory, see page 287.
- **comtr**
Outputs trace data directly to *stdout*, see page 291.
- **StartStop commands**
Live Upgrade installation supports starting/stopping of CMX and/or its components during normal operation, see page 302.

1.4 README files

Information on any functional changes and additions to the current product version can be found in product-specific Release Notices. You will find these notices in the readme package which is supplied with the relevant product.

1.5 Notational conventions

As far as possible, the command descriptions are presented within the following framework:

– Description of the command
– Syntax
– Syntax description
– Output format
– Exit status
– Error messages
– Files
– Example
– See also

The above components are explained below:

Description of the command

This first part of each command description tells you the following:

– how the command functions
– the various tasks of the different command formats if more than one command format exists
– the environment in which the command is to be used (e.g. entries in files, access permissions)
– background information

Syntax

cmd [␣**-a**] [␣**-b**] [␣**-c**] [␣**-d**␣arg1] [␣**-f**␣arg2]␣file␣...

You must enter *cmd* and specify for *file* one or more files, each separated from the next by a blank. You can also enter:

– one or more options **-a, -b, -c**. These can be entered separately (**-a**␣**-b**␣**-c**) or together (**-abc**).

– the option *-d*, where *arg1* must be replaced by an argument

– the option *-f*, where *arg2* must be replaced by an argument

Boldface characters
Constants. Characters in bold type must be entered exactly as shown.

Normal characters
: Variables. These characters stand for other characters which you can select and enter.

[ ]
: Options. Arguments between square brackets are optional and therefore need not be specified. The square brackets should not be entered unless otherwise specified.

␣
: Blanks. These must be entered.

..
: Repetition. The preceding expression can be repeated. If blanks which are not included in the expression are to be entered between repetitions, a ␣(blank) will appear before the ... .

{ | }
: Choice. Select one of the expressions separated by the vertical bar.

Underscore
: Default value.

If a command allows you to enter several alternatives for one option, the command syntax is given twice. In the first representation a positional parameter is entered for the relevant option. In the second the positional parameter is replaced by all possible entries for the option. The second representation is intended to provide quick reference for the expert.

Syntax description

This heading is followed by a description of the options and arguments (input files, parameters, variables, etc.) which you can enter when calling the command. In continuous text no distinction is made between constants and variables. All syntax elements as well as file names, path names and commands appear there in *italics*.

Output format

This section describes the output format(s) of the command.

Exit status

An exit status is the value returned to the calling process by a command after it has been executed. The value contains information on whether or not the command was executed successfully. The exit status is a numeric value and is stored in the variable ?. You can query the exit status using the *echo $?* command.

The exit status is only described if it has a value other than one of the two regular values shown below:

0 if the command was executed successfully

≠0 if an error occurred

Errors

This part explains important error messages and provides notes on avoiding and correcting errors.

Error messages are generally output to standard error output *stderr*. Normally, the screen is the standard error output.

Files

Under this heading you will find the files which are accessed or generated by the relevant command.

Example

Examples serve to illustrate the main function of the command, the use of the most important options and sensible combinations of options and arguments. In application examples, system input is displayed in fixed pitch boldface. All these input lines are completed with ↵. This key is therefore not specified at the ends of the lines.

Except in continuous text, system output is displayed in `fixed pitch`. In continuous text the output appears in *italics*.

See also

Here you will find references to other commands which have a similar function or work together with the command involved. You will also be referred to other literature relating to this command.

Notes and warnings

This symbol indicates particularly important information.

Caution!

This symbol indicates danger of data loss or damage to equipment.

2 Architecture of Solaris communication

This chapter contains an overview of the Solaris communication products. It describes the most important features of these products and explains the role played by CMX in Solaris communication and the services it offers. Here you will find explanations of all the most important terms used in subsequent chapters.

The Solaris communication products are used to illustrate the multitude of networking options provided by these products, and the services offered by CMX are listed. You can therefore see at a glance which services are available through which products.

Since using CMX requires a general understanding of the architecture of Solaris communication, the following sections describe how these services are provided. This architecture is described using the terminology that appears in the Solaris communication manuals. The only previous knowledge you require is an understanding of data communication and of the basic structure of the Solaris operating system.

2.1 Performance range of CMX and CCPs

The family of Solaris communication products offers the appropriate transport profiles for all major data networks. A number of program interfaces are provided to create application programs for communication.

To transmit data between the different types of transport profiles and program interfaces, CMX presents the TS applications with a uniform picture of the transport system. The advantage of this is that you can develop, TS applications independently of the transport system. The decision as to which CCP profiles are to be used for the applications is not made by the configuration until execution time (see section “Connection setup process” on page 50).

CMX and CCPs require appropriate system limits for unrestricted communication. In this regard, please note the information in the chapter “Installation and startup” on page 53, as well as in the Release Notice.

Administration and diagnostics

CMX offers commands for querying information on the utilization levels of communication resources and on the configuration and limits of the communication products. With additional commands the system administrator can query diagnostic information should problems arise. These functions can be activated easily using the menu interface CMXCUI (see chapter “Configuration and administration in the menu” on page 59).

Root authorization is no longer required to operate these functions. The entire range of configuration, administrative and maintenace tasks relating to CMX can be executed under the user role *cmxadm* as well. Every user of the system with authorization for this role can administer CMX. In the following sections, the terms system administrator and administration are synonymous with the user role cmxadm.

2.2 Network interfaces and transport profiles

CMX supports the connection of Solaris systems to all networks relevant to the market. This means integration in the most important communication architectures TCP/IP, OSI, TRANSDATA NEA, and SNA and all the usual physical networks. The diagram below illustrates the connection options.

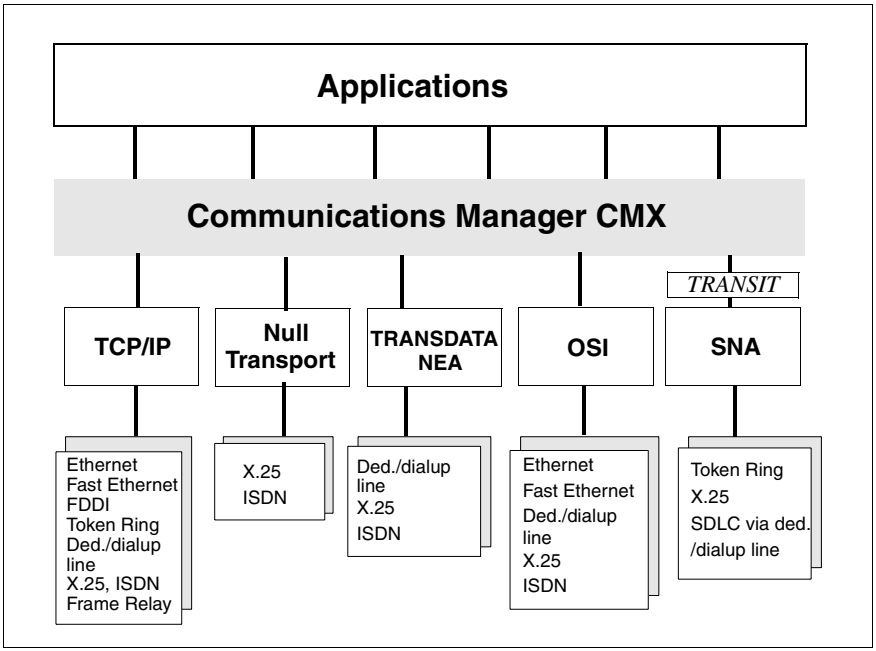


Figure 2: Network interfaces and transport profiles

In order for computers to communicate in local area networks or over wide area networks, they must be accessible using unique addresses. However, addressing depends on the protocols used and is therefore different in the individual network architectures. CMX supports the use of TCP/IP, ISO, and TRANSDATA NEA network addresses and thereby enables network-independent communication. Below is a description of the network architectures and of the components and features of the most important addresses.

2.2.1 TCP/IP architecture

TCP/IP can be operated over local and wide area networks of all types.

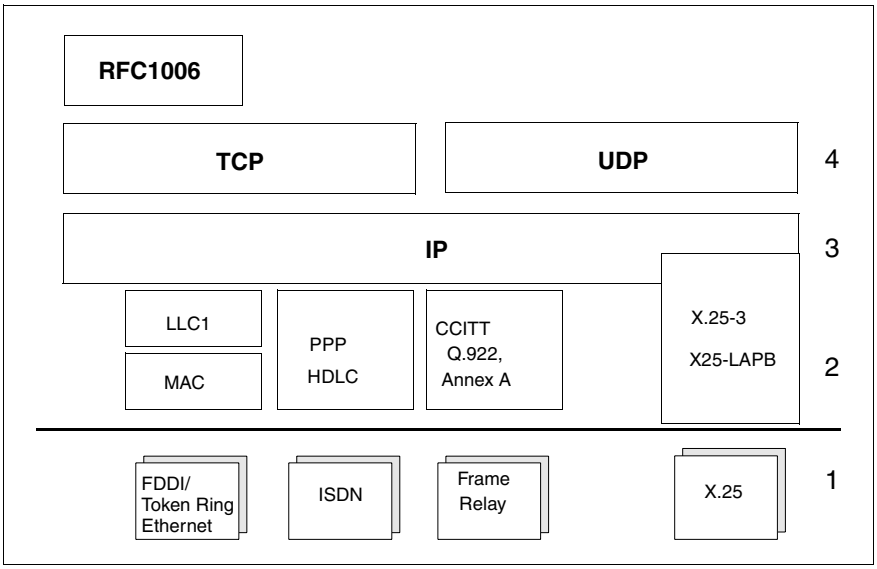


Figure 3: TCP/IP protocol stack

Layer 2 of the TCP/IP transport profile for LANs is split into the MAC sublayer and the LLC sublayer. The respective physical LAN is controlled in the MAC sublayer (Medium Access Control), while the messages are exchanged in the LLC sublayer (Logical Link Control).

In Token Ring and FDDI networks, IP uses the connectionless protocol LLC1 with an IP-specific extension called SNAP. In Ethernet LANs, depending on the MAC variant, no LLC or LLC1 protocol is used.

HDLC (with or without T.70-3) or the point-to-point protocol (PPP) is used for IP via ISDN. The CCITT protocol Q.922 Annex A applies above a Frame Relay subnetwork, while X25-LAPB is used over packet-switching networks.

TCP/IP addresses

An application TCP or UDP in the TCP/IP network (= Internet) is addressed by its port number and the IP address of the computer.

IP address

IPv4 and IPv6 addresses have the following differences:

– IPv4 addresses have a length of 32 bit. During inputting and outputting on screen these are represented and displayed as 8-bit decimals separated by a dot (e.g. 139.22.112.88).

– An IPv6 address has a length of 128 bit. In text displays, each 16-bit piece is shown as a hexadecimal value separated by a colon.

  Multiple adjoining hexadecimal numbers with the value 0 may be compressed using two consecutive colons once within a single address. (e. g. FE80::280:17FF:FE28:7B08).

  In special cases where an IPv6 address is derived from an IPv4 address, the section of the IPv4 address may be represented using the IPv4 method of representation (e.g. ::FFFF:139.22.112.88).

Port number

A port number contains 16 bit and shown in text displays as decimal integers.

When assigning port numbers to your own applications, note that particular port numbers are reserved for standard applications. For example, the application TELNET is addressed via port number 23, and the application FTP via port number 21.

Only port numbers greater than 1024 should be configured for your own use. The port numbers reserved worldwide are published regularly by the International Electrical Commission (IEC) as RFCs (Request for Comment).

RFC1006

The Internet standard RFC1006 defines how a further OSI transport service can be realized by adding another layer to the protocol stack above TCP. The TCP port number 102 is assigned and dedicated to this service. OSI applications that use this service also address the TCP/IP address via a T-selector (see section “OSI architecture” on page 17.

On partner systems, however, RFC1006 implementations occur which are not addressed via the triple mode with the IP address , TCP port number 102 and T-selector.
Instead they are addressed either **without** a T-selector component, using only a IP address and a TCP port number, or **with** a T-selector and a TCP port number. In both cases it is imperative that the TCPport number is not 102.
This implementation concerns CMX 3.0, CMX 4.0 and PCMX in particular. The current CMX version supports communication with such partners.

2.2.2 TRANSDATA NEA architecture

The architecture of the TRANSDATA NEA transport system is designed for telephone, X.21, and X.25 wide area networks (WAN) as well as ISDN. The NEATE protocol, which essentially contains the class 2 and 3 functions of ISO protocol 8073, is used on the transport layer.

The connectionless NEAN protocol is used on the network layer. On the data-link layer, the HDLC protocol is used for circuit-switching networks and the X.25 LAPB protocol for packet-switching networks. The same protocols can be used for NEA via ISDN.

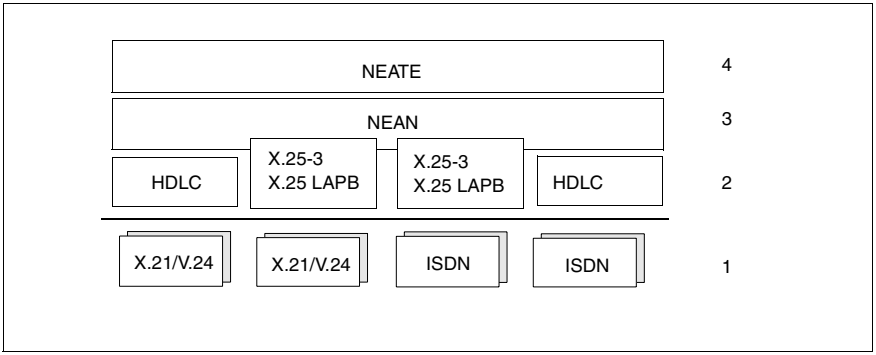


Figure 4: Architecture of the TRANSDATA NEA transport system

TRANSDATA NEA addresses

TRANSDATA NEA networks can be split into a maximum of 256 subnetworks (regions). Up to 256 computers can be addressed within a region, and 2046 stations can be operated on each computer. In the TRANSDATA NEA concept, the term ‘station’ refers to both data stations and applications. Data stations can be printers or terminals.

The hierarchy of region and computer in TRANSDATA NEA networks is reflected in the addressing.

Network address

The network address is used to address computers in the TRANSDATA NEA network. It comprises the processor number and the region number (e.g. 1/18).

The processor number uniquely identifies a computer within a region. Each region is assigned a region number which is unique throughout the entire TRANSDATA NEA network.

The processor number and region number lie in the value range between 0 and 255. In this way, 65536 computers can be uniquely addressed within a TRANSDATA NEA network.

2.2.3 OSI architecture

A range of OSI protocol profiles were defined by the various national and international organizations for the individual WAN types. The CCP profiles offer what are in practice the most important of these profiles.

The connection-oriented transport protocol IS8073, class 0 or 2, which is based on the connection-oriented network service X.25 or on T.70-3, is generally used on Layer 4 in the WAN (and ISDN).

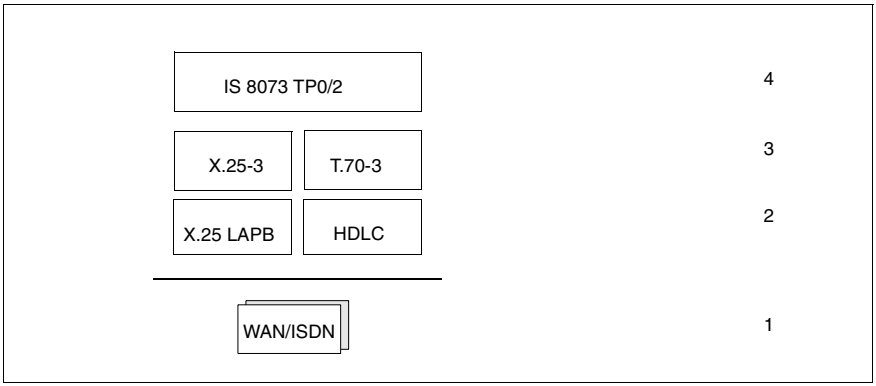


Figure 5: OSI protocol stacks

OSI transport addresses

OSI transport addresses comprise the OSI-NSAP address and a transport selector.

OSI-NSAP address

OSI addresses comprise the components AFI (Authority and Format Identifier), IDI (Initial Domain Identifier), and DSP (Domain Specific Part). (For further information, turn to section “Address components and their formats” on page 86.) In many cases however, particularly with WAN connections, the addresses of the underlying subnetwork (e.g. X.25 addresses) are used. In such cases, the OSI-NSAP addresses are not required.

Transport selector

An application uses the transport selector (T-selector) to attach to the OSI transport service. With an incoming connection request, the transport system uses the T-selector to identify the application called.

2.3 Interfaces and applications

Applications that use the services of a transport system, irrespective of the platform on which they run, are generally termed **TS applications** throughout this manual. TS applications can access CMX services via a number of programming interfaces. The CMX application can access a transport system via a programming interface with uses an ICMX, XTI, TLI or an NLI application.

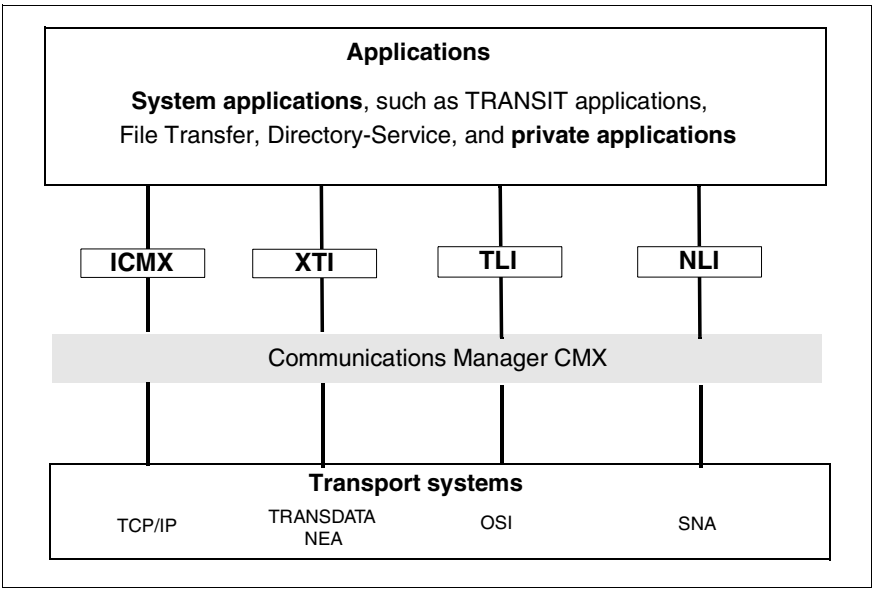


Figure 6: Programming interface for CMX and applications

The ICMX interface supplied with the CMX product enables access to all transport systems offering a transport service in accordance with IS 8072. These include the pure OSI transport systems via WAN, the widely used variant RFC1006 providing the OSI service via TCP/IP and TRANSDATA NEA.

The ICMX interface is provided for the main UNIX systems and BS2000/OSD. Implementations for Windows are also provided. Typical examples for ICMX applications are *open*UTM, TRANSIT and applications designed for the end-user (e.g.EMDS, *open*FT or MAIL.X).

The interface X/Open Transport Interface (XTI) opens access to TCP and UDP as well as to OSI transport services. This interface is also provided for all the main UNIX systems. Applications produced by independent software companies frequently use this interface.

Transport Layer Interface (TLI) is another non-proprietary interface to the transport layer in UNIX systems. TLI applications can be run via the CCP transport systems provided they use transport services in accordance with IS 8072. (see chapter “Using TLI applications” on page 361).

CMX supports the communication functions of the Network Layer Interface (NLI), which enables direct access to X.25 networks on SUN systems. You must obtain special authorization in order to be able to use this interface.

It is general practice to identify applications by the transport system they use. Applications using the OSI transport system, for example, are frequently called OSI applications.

2.4 Solaris communication products

This section lists the transport profiles implemented by CMX and the communication products, and describes the functions of the *Communication Services*.

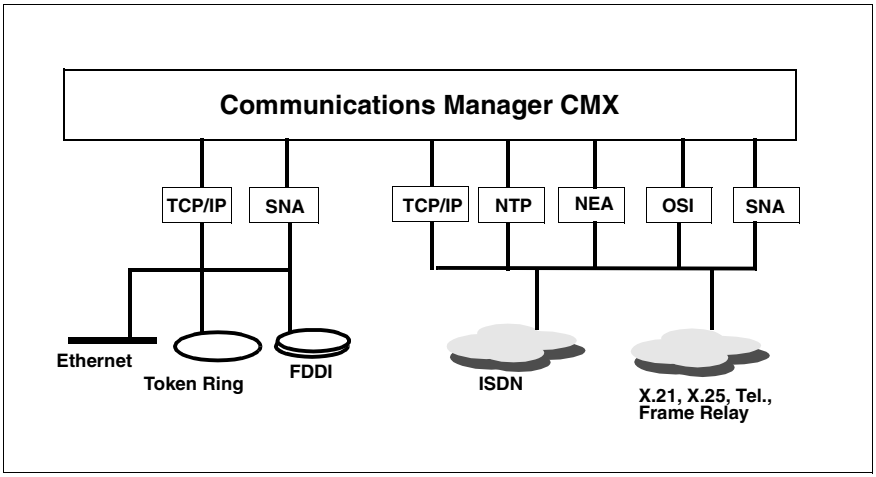


Figure 7: Network interfaces via CMX

The Solaris communication products offer the following functions and services:

– The appropriate transport profiles for all the main data networks are offered in the form of the CCP family *(Communication Control Programs)*. A CCP can provide one or more transport profiles. Because of their implementation as CCPs, transport profiles are referred to below as *CCP profiles*.

– Several program interfaces are offered for TS applications: the ICMX(L) and ICMX(NEA) interfaces, which are components of CMX (see the “Programming Applications” manual [1] and the manufacturer-independent XTI interface).

– All components required for local configuration and administration are likewise contained in the CMX product (see chapter “Configuration and administration in the menu” on page 59).

2.4.1 Transport profiles

Below is an overview of the transport profiles provided by the Solaris communication products:

– The TCP/IP transport service, which is contained in the basic Solaris system.

– The implementation of the ISO transport service in accordance with ISO8072 class 0 via TCP/IP by the convergence protocol RFC1006. This transport profile is a component of the CMX product and is described in detail in the chapter “Configuring connections via RFC1006” on page 207.

– Direct access to X.25 packet-switching networks.

– The TCP/IP transport service via X.25 or Frame Relay.

– Connection oriented OSI transport service with RFC1006 via TCP and X.25 or Frame Relay.

– The TCP/IP transport service via ISDN.

– Connection oriented OSI transport service with RFC1006 via TCP and ISDN.

– The SNA transport service via ISDN.

– The OSI transport service in accordance with ISO8072 classes 0 and 2 for OSI connections over wide area networks.

– The OSI transport service in accordance with ISO8072 classes 0 and 2 for OSI connections via ISDN.

– The SNA transport service via WAN.

– The TRANSDATA NEA transport service via ISDN, X.25, dedicated or dial-up lines.

To implement all of the transport profiles, the CMX product is required. In Solaris communication, the implementation of a transport profile is referred to as a CCP profile. The CCP products are described in detail in the relevant manuals.

2.4.2 Routing service

The **routing service** expands the Solaris communication functions. You can also use your Solaris system as a router.

CS-ROUTE can be used to connect LAN islands using the TCP/IP protocol via WAN/ISDN. CS-ROUTE can coordinate with other routers using the routing protocol OSPF. CS-ROUTE operates via Frame Relay, X.25 networks, or ISDN connections and also supports the point-to-point protocol (PPP). The routing service is also required for the communication of local TCP/IP applications via WAN/ ISDN.

2.5 Architecture of CCP profiles

This section describes the implementation of the CCP profiles offered with CMX and the CCP products.

When classifying CCP profiles, a basic distinction is made between **local area networks (LANs)** and **wide area networks (WANs)**. A LAN is a network covering a relatively small area which is dependent on the technology used, but which enables high-speed transmission and thereby permits the rapid exchange of large volumes of data. The area covered is often limited to one storey, a building, or a building complex. A LAN is operated and managed privately.

Although a WAN, on the other hand, generally offers a lower transmission speed than a LAN, it does provide the option of global communication. Examples are the X.25 networks or the Integrated Services Digital Networks (ISDN) of network operators such as Deutsche Telekom, British Telecom, France Télécom, or AT&T.

A CCP comprises two components:

– A **transport service provider (TSP)**), which offers the services of the transport layer and part of the network layer (Layer 3c in the OSI Reference Model).

– A **subnetwork profile**, which implements the services for supporting subnetwork interfaces (Layers 1, 2, and 3a in the OSI Reference Model).

A TSP refers to the components of CCP profiles which are required to control subnetwork profiles. These components have a particular transport protocol and the respective transport service for the different network architectures. The following TSPs are available:

– TSP RFC1006 for the OSI transport service viaTCP/IP

  TSP RFC1006 enables TCP/IP to be used as an OSI-equivalent network service. To be able to use this TSP, you must specify address information which is managed in the transport name service (TNS, see section “Addressing transport system applications in TNS” on page 37). When communicating with CS-ROUTE, additional information must be specified in the forwarding support service (FSS, see section “Addressing partner systems in the FSS” on page 45).

– Null Transport for ISDN and X.25 communication

  TSP Null Transport (NTP) offers direct access to the services of the subnetwork.

– TRANSDATA NEA-TSP for the TRANSDATA architecture

  TRANSDATA NEA-TSP provides the transport service in the TRANSDATA network. To be able to use this service, you must define address information (and possibly partner-specific service features), which is managed in the forwarding support service (FSS, see section “Addressing partner systems in the FSS” on page 45) and in the transport name service (TNS, see section “Addressing transport system applications in TNS” on page 37).

– OSI TP0/2 for OSI communication in the ISDN and other wide area networks

  OSI TP0/2 is the TSP for an OSI environment with the OSI transport service of classes 0 and 2. To be able to use this service, you must define address information (and possibly partner-specific service features), which is managed in the forwarding support service (FSS, see section “Addressing partner systems in the FSS” on page 45) and in the transport name service (TNS, see section “Addressing transport system applications in TNS” on page 37).

CMX defines a TSP access point for each transport service provider (TSP). A TSP access point defines the access point of the CMX automaton (central component of CMX) to the transport service provider.

At these access points, CMX offers the communication components a uniform view of the transport system.

To enable TS applications to communicate via a TSP, the TSP must be ready for operation and must grant the CMX automaton access via a *TSP access point*. The TSP access point is particularly important for maintenance and diagnostic functions (see chapter “Administration and maintenance” on page 227).

A subnetwork profile refers to the loadware loaded on the communication controller (CC).The subnetwork profile controls the CCs for the respective subnetwork. The configuration file (CF) for the subnetwork profile defines the features of your local subnetwork interface, e.g. your own ISDN call number, the protocols to be set when the connection is established, and the X.25 features of the transition to the X.25 network or to an X.25 partner on the ISDN.

At system startup, the subnetwork profile assigned to the CC, including the assigned configuration file, is loaded on the CC. With a single configuration file, you can configure the subnetwork profile such that your system can simultaneously use different transport services via one and the same CC on a subnetwork interface.

The components TSP and subnetwork profile are executed differently depending on the type of CCP. The various implementations are described in the following diagram and subsequent text.

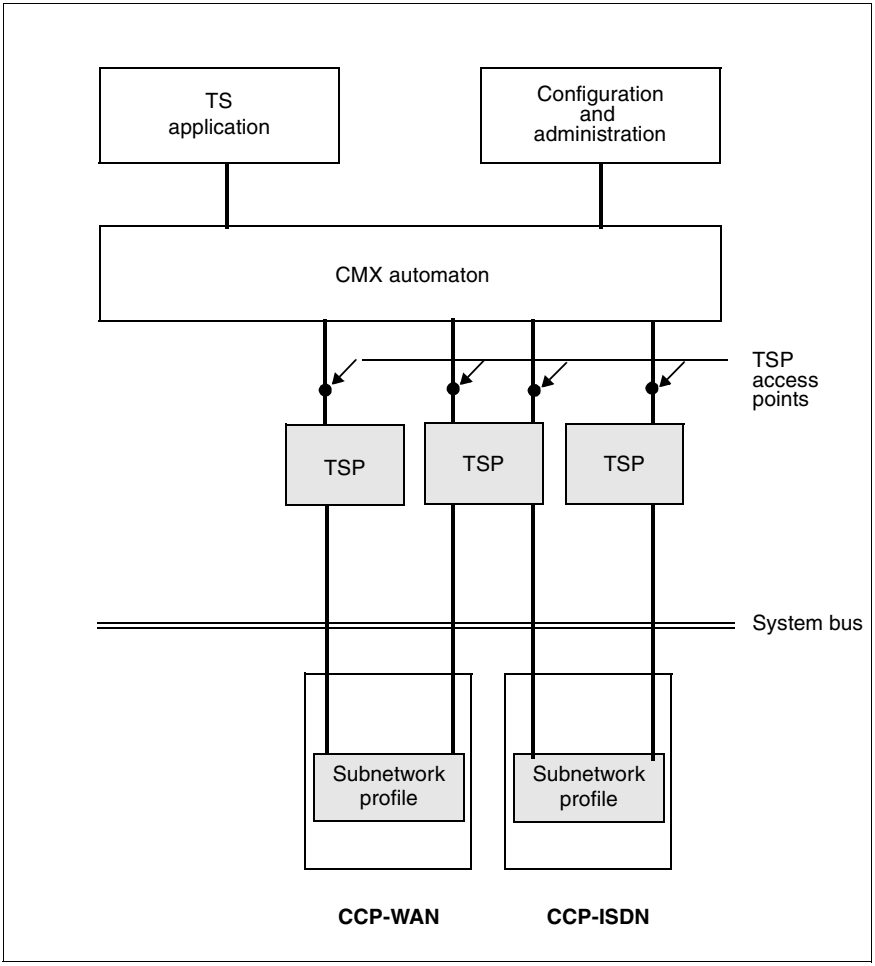


Figure 8: Implementation of CCP profiles

2.5.1 LAN profiles

The TSP RFC1006 and the TCP/IP (sub)network are implemented as Solaris kernel components. The network is accessed via intelligent **communication controllers (CCs)**, which are managed by Solaris.

The advantage of this solution is that you can simultaneously run two profiles via the connection supplied by the basic Solaris system without additional CC hardware:

– RFC1006 convergence protocol via TCP/IP

– TCP/IP without convergence protocol

The TSP RFC1006 is configured and administered using CMX.

2.5.2 WAN profiles

The TSPs Null Transport, OSI TP0/2 and TRANSDATA NEA are also components of the Solaris kernel.

Subnetwork profiles run on programmable CCs. Both TSPs and subnetwork profiles are configured and administered using CMX. As there is normally no need for a paired assignment between TSPs and subnetwork profiles (e.g. the TRANSDATA NEA and OSI TP0/2 can be based on the same X.25 subnetwork profile), TSPs and subnetwork profiles are configured and administered independently of each other.

With the option of running subnetwork profiles via different TSPs, the utilization of lines and CCs is optimized.

The subnetwork profile of the WAN CCPs supports connection to X.25 networks, Frame Relay networks, as well as analog and digital dedicated and dial-up lines.

2.5.3 ISDN profiles

The software structure corresponds to the WAN profiles. Here too, various TSPs can simultaneously use the same subnetwork profile.

2.5.4 SNA profiles

Some of the SNA protocols are contained in the TRANSIT protocols. For a complete SNA connection, the suitable TRANSIT product is therefore required in addition to CMX and the respective CCP product.

2.6 Areas of application

This section contains examples of heterogeneous network architectures with typical areas of application.

TCP/IP via ISDN

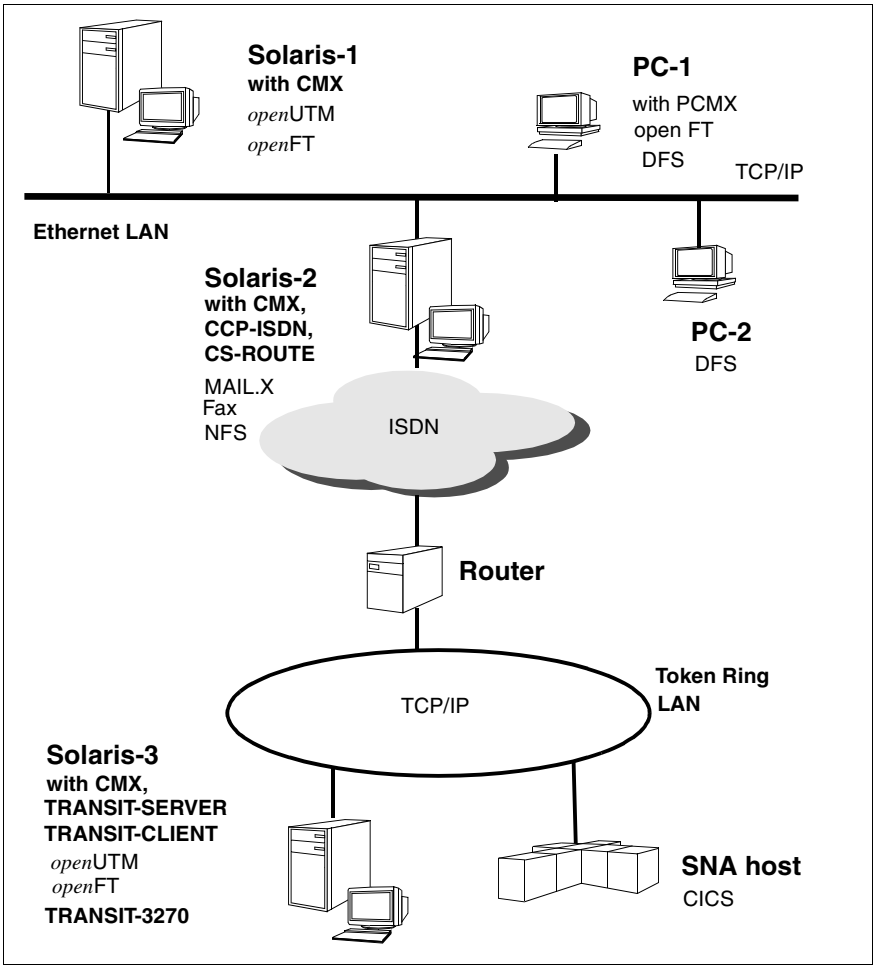


Figure 9: TCP/IP applications via ISDN

In the illustrated configuration, two local area networks (Ethernet and Token Ring) – each using the TCP/IP protocol – are linked via ISDN. The LAN-WAN connection is enabled either via a dedicated router or via a Solaris system running CS-ROUTE software. Data is exchanged using the IP protocol between all of the applications, so that the ISDN network remains invisible to the applications.

System Solaris-2 can be implemented as a server for PC clients. The server system provides network services via the LAN, for example, and also via ISDN with the aid of CS-ROUTE. The DFS (distributed file service) software is required on the PCs for this purpose. MAIL applications can also exchange data (here between MAIL.X on Solaris-1 and MAIL.D on PC-1) without the need for additional communication software on top of CMX.

The SNA host and Solaris systems are connected to the Token Ring LAN. They also communicate with each other via TCP/IP. With the software TRANSIT-SERVER and TRANSIT-CLIENT, for example, you can run your Solaris system as an SNA terminal for interactive and transaction processing mode (LU6.2 functionality).

*open*UTM on system Solaris-3 enables distributed transaction processing (e.g. when accessing CICS in the illustrated SNA system or in communication with an *open*UTM application on Solaris-1 in the Ethernet LAN).

Similarly, file transfer applications on Solaris-1 and Solaris-3 can communicate with each other via *open*FT. In this case, you will need CMX on Solaris-1 and Solaris-3, as well as CMX and CCP-ISDN on Solaris-2.

A similar configuration is conceivable both via ISDN and via other wide area networks (X.25, X.21). In the latter case, the CCP-WAN software must be installed instead of an ISDN product.

Connecting Solaris systems with SNA hosts

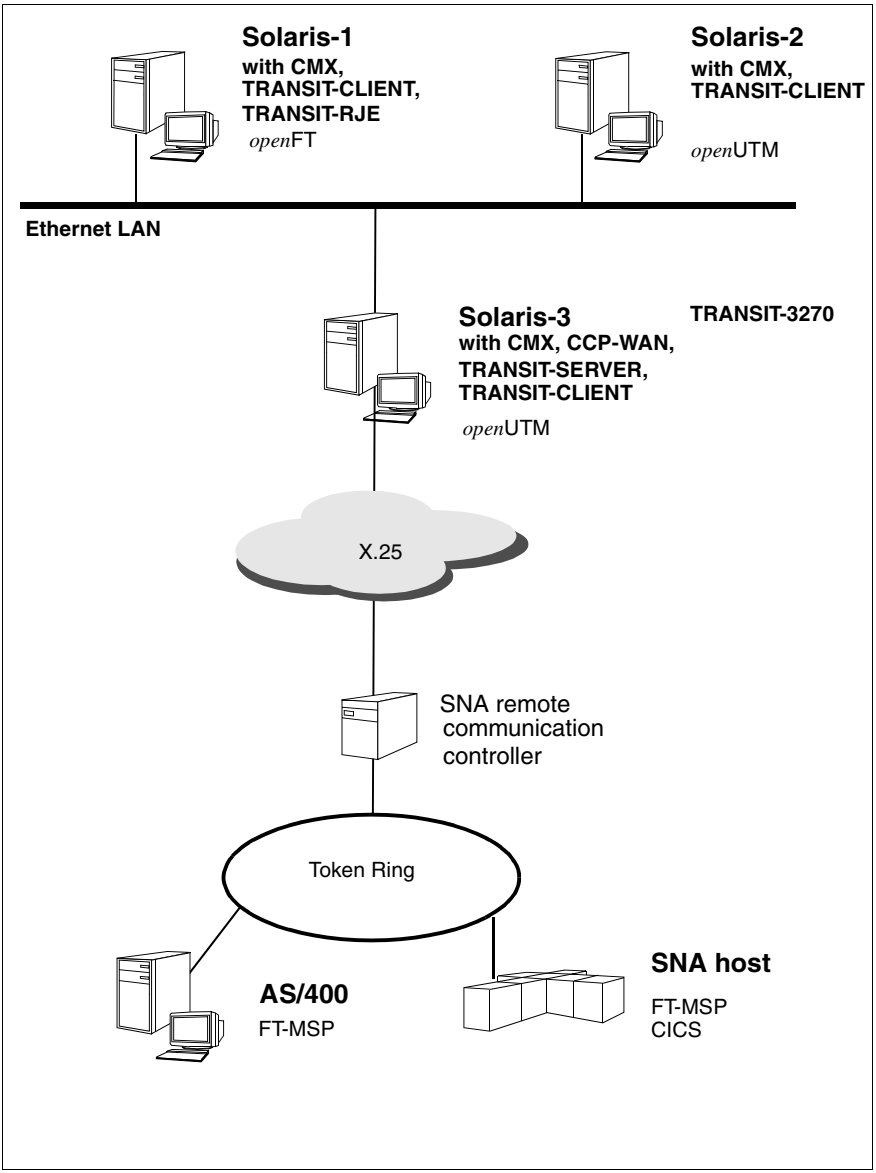


Figure 10: Connecting Solaris systems with an SNA environment

In the illustrated configuration, a local SNA network (Token Ring) is connected to an Ethernet LAN via an X.25 WAN. The Solaris systems are located in the Ethernet LAN.

Solaris systems 1, 2 and 3, which are integrated in an Ethernet LAN, can use different SNA protocols (LU1, LU2, LU3, LU6.1, LU6.2) to communicate with SNA systems that are accessible via the X.25 network. The system Solaris-3 has direct access to the X.25 network. The TRANSIT-SERVER product must be installed on this system; TRANSIT-CLIENT (or alternatively TRANSIT-CPIC) must be installed on Solaris systems 1 or 2.

Applications such as TRANSIT-RJE (Remote Job Entry) or TRANSIT-3270 (emulation of a 3270 data display terminal) can therefore run on Solaris systems 1 and 2.

For file transfer applications, you will need the *open*FT product on your Solaris system (here: Solaris-1); FT-MSP must be installed on the SNA host. For the AS/400 system, you will need the FT-400 product for file transfer applications.

You communicate with CICS applications on the SNA host via *open*UTM on the Solaris system (here: Solaris-2). TRANSIT-CLIENT forms the bridge between TRANSIT-SERVER on the one side and *open*FT or *open*UTM on the other.

The same communication relationships are possible if the SNA system is not attached to the Token Ring LAN rather has direct access to the X.25 network.

Connecting LANs via the X.25 network

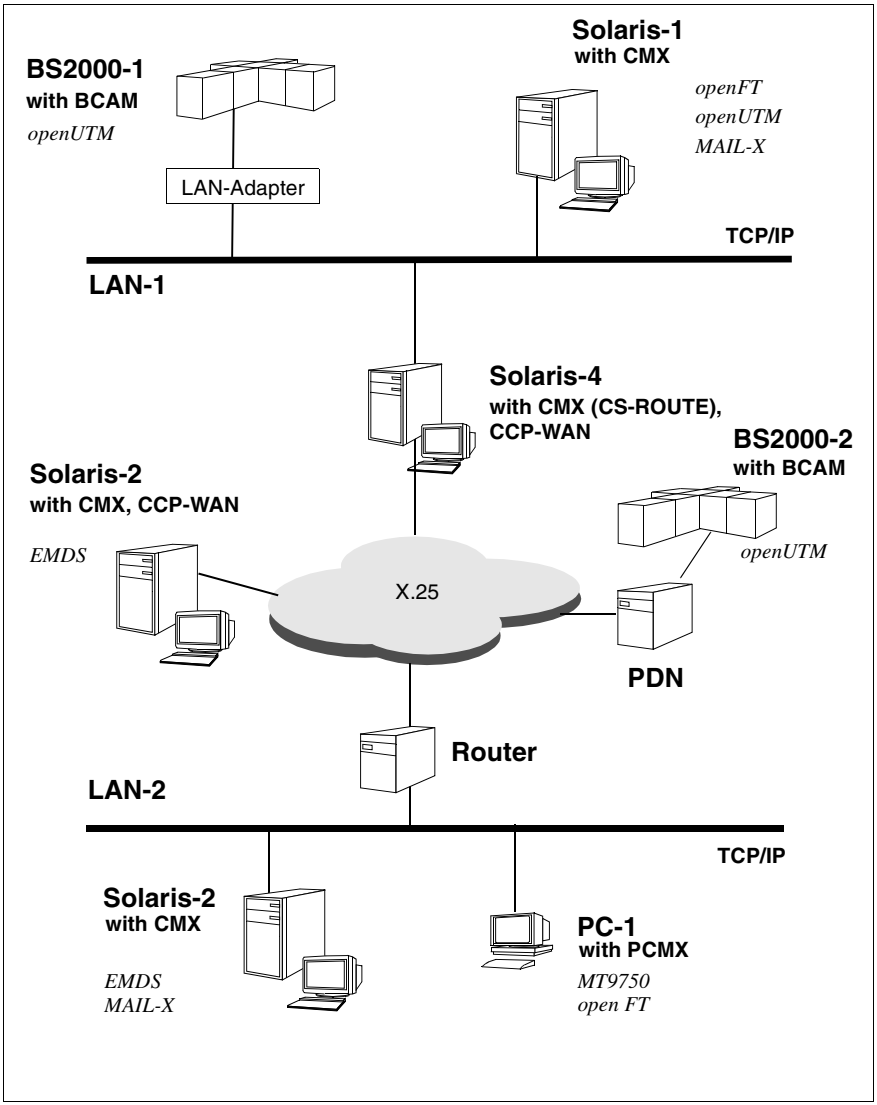


Figure 11: Connecting LANs via X.25

3 The user role cmxadm

Solaris introduced an authorization infrastructure named RBAC (role based access control). It implements a type of functional access control that ensures that only system users equipped with the appropriate authorization may execute privileged system operations. Additionally, execution of these operations is no longer tied to the *root* ID of the system administrator and can be assigned to other users of the system. The administration of CMX can thus also be conducted using a specified ID.

3.1 Central concepts

To facilitate an understanding of the model outlined below, we now give a brief explanation of the following concepts. Detailled information on RBAC can be found in the system administration manuals of Solaris 8/9 (see “Solaris 8/9: System Administration Guide, Volume 2” [7]).

Role

A role defines a particular system user who is assigned a specific array of functions drawn from the system’s set of commands. The execution of a function coincides with the executing program or script checking, if the user who initiated the function is actually authorized to do so. When you create a user (with the *useradd* command) or modify an existing user (with the *usermod* command) you can assign one or more of roles to the user. The command *roles* allows users to see which roles have been assigned to them. Access to a role is achieved using the *su* command on the local system.

Authorization

Authorization defines the permission to execute certain actions on the system. Permissions are represented by a text string. The program or script executing the action checks if the user initiating the action has the appropriate permission to do so. The command *auths* displays current authorizations assigned to a user.

Profile

A profile of priviliges describes the portion of functions that a user is allowed to execute on the system. These functions are determined implicitely by a number of authorizations and additional commands which may be executed with specific security attributes (e.g. execution of privileged commands with the actual of effective UID *root*). The command *profiles* lists the current profiles assigned to a user.

3.2 CMX installation: extensions of the RBAC data structures

The installation of CMX with the package *SMAWcmx* introduces the following RBAC specific objects to the system:

1. Authorization: *com.fujitsu-siemens.cmx.oam*. Authorization grants the necessary privileges to respective processes thereby permitting these to administrate CMX. CMX programs that modify operational parameters or start/stop-components use this authorization to verify whether the initiating process actually has the authorization to do so.
2. Rights profile: *CMX Administration*. The authorization *com.fujitsu-siemens.cmx.oam* is assigned to this profile. This profile also authorizes the execution of some privileged user commands with defined security attributes.
3. The user role *cmxadm* to which the rights profile *CMX-Administration* is assigned. The user role is installed without a password. The initial login requires the user to specify a password. As default, the user group *cmxadm* is assigned to this user role and to the directory */var/opt/SMAWcmx/adm/log*.

However, it is also possible to adminster these objects centrally via the name services within a cluster of systems.

When CMX V6.0 is first installed, the program assumes that the user ID *cmxadm* does not yet exist on the system. If this already exists the installation will be interrupted. The uninstall procedure of CMX V6.0 leaves the ID on the system.

In order to use the cmxadm role, the system administrator must conduct the following procedures:

1. Specify a password for the user role *cmxadm*. Password entries can be adjusted to locally applicable administration rules after CMX has been installed.

2. Defining users who are permitted to administer CMX, for example, via the command line :

   ```
   usermod -R cmxadm hugo
   ```

   granting the user hugo CMX administration privileges.

The rights which have been granted to a user to administer CMX can be withdrawn at a later date if necessary.

3.3 Functions of the user role cmxadm

In addition to the set of commands available to all users, the user role *cmxadm.* also has access to the configuration, administration and maintenance functions of CMX/CCP. The functions may be initiated using the CMXCUI graphic panel or the Command Line Interface.

The user role *cmxadm* does not include the privilege to install the CMX/CCP software nor the right to initiate a system shutdown in order to replace controllers. These operations remain the reserved privilege of the supervising system administrator. It is nevertheless possible to extend the spectrum of functions allocated to the user role *cmxadm* by assigning corresponding rights profiles.

The user role *cmxadm* is an optional feature. It is up to each individual system operator to decide whether to re-allocate CMX administration to the user role *cmxadm*. All OAM tasks can still be executed under the administrator ID *root.* Simultaneous administration is also possible.

3.4 CMX administration under cmxadm

The administration of CMX under the user role *cmxadm* may be conducted on the local system or remotely. An authorized user can use the *su* command to switch to the user role *cmxadm* which the takes place after successful authentication.

A login via *rlogin* is possible provided that the user names and the corresponding IDs are identical throughout the system cluster. The system to be administered checks if the user ID logging on has the required privileges to adopt this user role, before checking the authentication itself.

4 Addressing concept

The data you need to configure CCP and CMX is managed by three different system components:

– the Transport Name Service (TNS)

– the Forwarding Support Service (FSS)

– the configuration files (CF) of the subnetwork profiles

The following sections describe the function of these components in the addressing procedure.

4.1 Addressing transport system applications in TNS

All networks and transport systems require special addressing information in order to be able to address the communication partners. CMX provides a service called the Transport Name Service (TNS), which you can use to manage the names and addresses of TS applications, regardless of the communications interface they use (ICMX or XTI).

TNS reads the address information from a directory called the “TS directory” (transport service directory). The address information for each TS application is stored in the TS directory under its symbolic name, known as the GLOBAL NAME. The TS directory contains information on all the TS applications resident in the local system and on the potential communication partners in remote systems.

A TS application only uses its own GLOBAL NAME and the GLOBAL NAMES of its communication partners.

In other words, the TNS makes the TS applications independent of the configuration environment in terms of the communications hardware and software. This independence relates to the following, for example:

– the type and number of communication controllers (CC) installed on your system

– the topology of the network in which your system is integrated

– the CCP profile running on your system

When the TNS is in use, it is no longer necessary for any of these features and data to be taken into consideration within a TS application, irrespective of whether they arise at the place of execution, the system running the partner application, or en route to this system (addressing, routing).

The TNS manages the configuration dependencies listed above in the TS directory, where they are stored in the form of properties of the TS applications. The applications are identified by means of a hierarchically structured name, their GLOBAL NAME. The program interfaces ICMX(L) and XTI provide a range of query functions with which a TS application can access these features. This means that the ICMX or XTI application does not have to process physical addresses. The application thus becomes easier to use in new environments, and is independent of changes in the network.

The main TNS elements - the GLOBAL NAME, the properties, and the TS directory - are described in the following sections. For information on how to manage a TS directory, see section “Address management in TS directories” on page 38.

For the sake of clarity, fixed terms such as the GLOBAL NAME or the names of properties of TS applications, are written in uppercase letters in the body of the text. TS applications resident in the local end system are referred to below as local TS applications. Similarly, TS applications, resident in remote end systems are called remote TS applications.

4.1.1 Address management in TS directories

The TNS manages all properties of the TS applications in the Transport Service Directory (TS directory). The TS directory is made up of entries, each of which contains the properties of one TS application. Each entry is identified by the GLOBAL NAME. The TS directory contains entries for all TS applications in the local end system and all TS applications in remote end systems which are used as communication partners.

4.1.2 Identification by GLOBAL NAME

In order to establish communication relationships, TS applications must be able to address each other, in the same way as subscribers in a telephone network communicate using their telephone numbers. Just as the telephone network operator allocates telephone numbers to the subscribers, you explicitly allocate

the TS applications their identifications: you assign a unique GLOBAL NAME to each local and each remote TS application. In this case, ‘global’ means that the name applies independently of the networks used.

A GLOBAL NAME is a hierarchically structured application name. This name can be split into a maximum of 5 parts (name parts 1 through 5). Of these, name part 1 is the highest in the hierarchy, while name part 5 is the lowest. As an example, think of a (worldwide!) telephone directory with country code, local area code, and telephone numbers, or an address book specifying nationality, zip code, postal subdistrict, street, house number, and name of the recipient.

Naming conventions apply to some standard applications. For example:

– The Enterprise File Transfer openFT uses the GLOBAL NAMES *$FJAM, $FJAM001, $FJAM002* etc.

– The product EMDS uses the GLOBAL NAMES *dss_000*, *drs_000, dss_001, drs_001* etc. for its applications.

A network here constitutes all systems which are addressed according to a particular schema, for example the TRANSDATA NEA network (addressing via processor number and region number, e.g. *23/355*) or the Internet (addressing via IP address, e.g. *139.22.195.99*). Each system in a network is identified uniquely by means of its network address (synonym: *NSAP address*). A system which is integrated in several networks has a specific network address for each of these networks.

4.1.3 Address information in the GLOBAL NAME

When a connection is addressed, a distinction is made between local (=residing on the local system) and remote (= residing on the partner system) TS applications. All are addressed by their GLOBAL NAMES; however, they differ in the address information assigned to the GLOBAL NAME.

The TS directory must contain the GLOBAL NAMES of all local TS applications and all TS applications on remote systems with which a local TS application is to communicate. You can assign specific properties to each of these GLOBAL NAMES, i.e. each leaf in the naming tree. The particular properties you can assign to a TS application depend on whether the TS application resides in the local system or in a remote end system.

The properties LOCAL NAME and TRANSPORT ADDRESS are particularly important for communication. You must assign the property LOCAL NAME to each TS application in the local system, and you must assign the property TRANSPORT ADDRESS to each TS application in a remote end system.

Regardless of whether the TS application is local or remote, you can also enter the session component and the presentation component.

The hierarchical structure of the GLOBAL NAME defines the arrangement of all GLOBAL NAMES in a naming tree. A leaf (leaf entity) in the naming tree corresponds to a TS application. A selection of properties can be assigned to a leaf, e.g. TRANSPORT ADDRESS.

The path from the root of the naming tree to the leaf is defined by the GLOBAL NAME of the TS application. The name can consist of up to 5 name parts which specify the path from the root of the tree via the (maximum of 4) nodes to the leaf. Name parts can also be omitted. An example of a complete naming tree with all its name parts is given below:

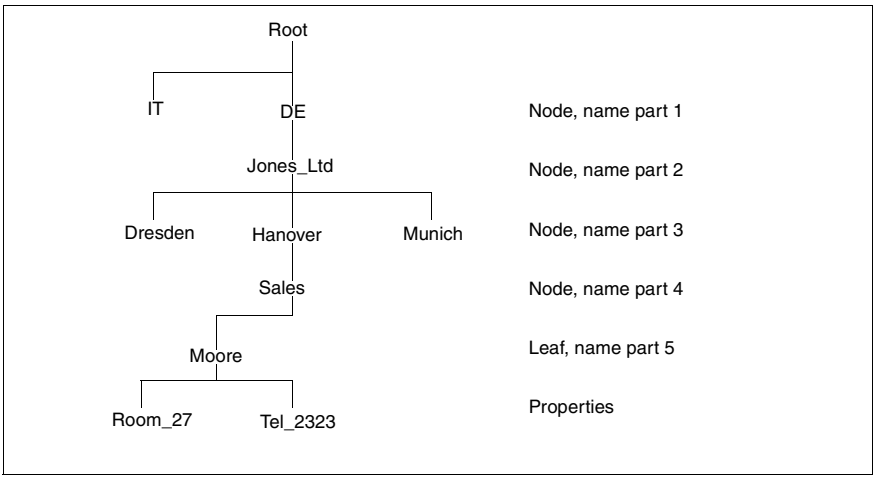


Figure 12: Example of a naming tree

The GLOBAL NAME corresponding to figure 12 looks like this in TNS notation:

```
Moore.Sales.Hanover.Jones_Ltd.DE
```

Note that name part 5 appears first.

Summary of the features of a GLOBAL NAME

- A GLOBAL NAME is a path in the naming tree extending from the root to a leaf
- The name parts are the path components
- The nodes and leaf are defined when the GLOBAL NAME is created
- Name parts 1 to 4 can be path components leading to a leaf
- Name part 5 cannot be a path component leading to a node
- A further node or leaf can be joined to a node, due attention being paid to the hierarchy
- Properties can only be assigned to a leaf

In the naming tree only the leaves are created. A node with no leaves cannot be explicitly created. A node is, however, created implicitly when leaves are created, and deleted when all the leaves assigned to the node are deleted.

Structuring GLOBAL NAMES

Exactly how the naming tree is structured, whether with or without differentiation according to the root and node and leaf, depends on the specific application in hand and the overall configuration of all TS applications. It is at the network administrator's discretion to decide how "deep" to make the tree structure. The number of TS applications will also play a part in determining the structure: for a small number of applications a "flat" tree without nodes will suffice; for many it will prove expedient to structure with nodes in order to utilize the benefits these offer with regard to clarity, access optimization, etc. It is advisable to structure the tree along organizational or topological lines.

Meaning of the name parts

The nomenclature of the name parts is based on the proposals of the international standardization bodies ISO, CCITT, and ECMA.

The following assignment applies to a complete naming tree:

Name part	Designation	Meaning	Length in bytes	Position in the tree
1	TS_COUNTRY	Country	2	Node, leaf
2	TS_ADMD	Administrative domain	16	Node, leaf
3	TS_PRMD	Private domain	16	Node, leaf
4	TS_OU	Organizational unit	10	Node, leaf
5	TS_PN	Personal name	30	Leaf

Table 1: Meaning of the name parts

4.1.3.1 Local TS application

To be able to communicate, a local *TS application* must sign on to the transport system. In so doing it must indicate which transport service providers (TSP) it wants to use. The various TSPs are identified by different address formats. In some cases, the address format also designates a particular combination of TSP and transport profile or address variant.

Example:

– Address format WANSBKA (for TSP OSI TP0/2) for OSI connections via WAN according to ISO8072.

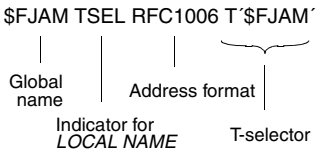
– Address format RFC1006 (for TSP RFC1006) for host-to-host connections via TCP/IP with RFC1006 convergence protocol.

The address information to be entered for the various address formats may vary. It could be any of the following:

– the station name for an NEA transport profile
– LU name and LU number for a SNA transport profile
– T-selector for a OSI transport profile
– TCP port number, supposing the RFC1006- standard port number 102 is not used for addressing.

Entries for local TS applications always begin with the indicator TSEL.

Example



You therefore assign a set of T-selectors to the GLOBAL NAME of a local TS application, together with the address formats. The sum of these entries is called the LOCAL NAME.

Example: The application identified by the GLOBAL NAME *$FJAM* is to be made known to the TSPs RFC1006 (address format RFC1006), TRANSDATA NEA (address format WANNEA), and the local loopback (address format LOOPSBKA). In this case, three TSEL entries must be specified as the LOCAL NAME:

```
$FJAM \
   TSEL RFC1006 T´$FJAM´
   TSEL WANSBKA T´$FJAM´
   TSEL LOOPSBKA T´$FJAM´
```

4.1.3.2 Remote TS application

The following information is required to address a *remote TS application*:

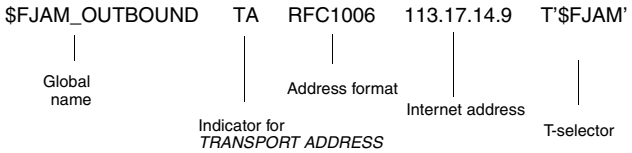
– the remote system on which the application is running (network address)

– the TSP via which it can be reached (address format of subnetwork interface)

– how it is identified on the remote system (T-selector of the remote application)

In CMX, this information is called the TRANSPORT ADDRESS. The TRANSPORT ADDRESS is assigned to a GLOBAL NAME, which identifies the remote TS application.

Example



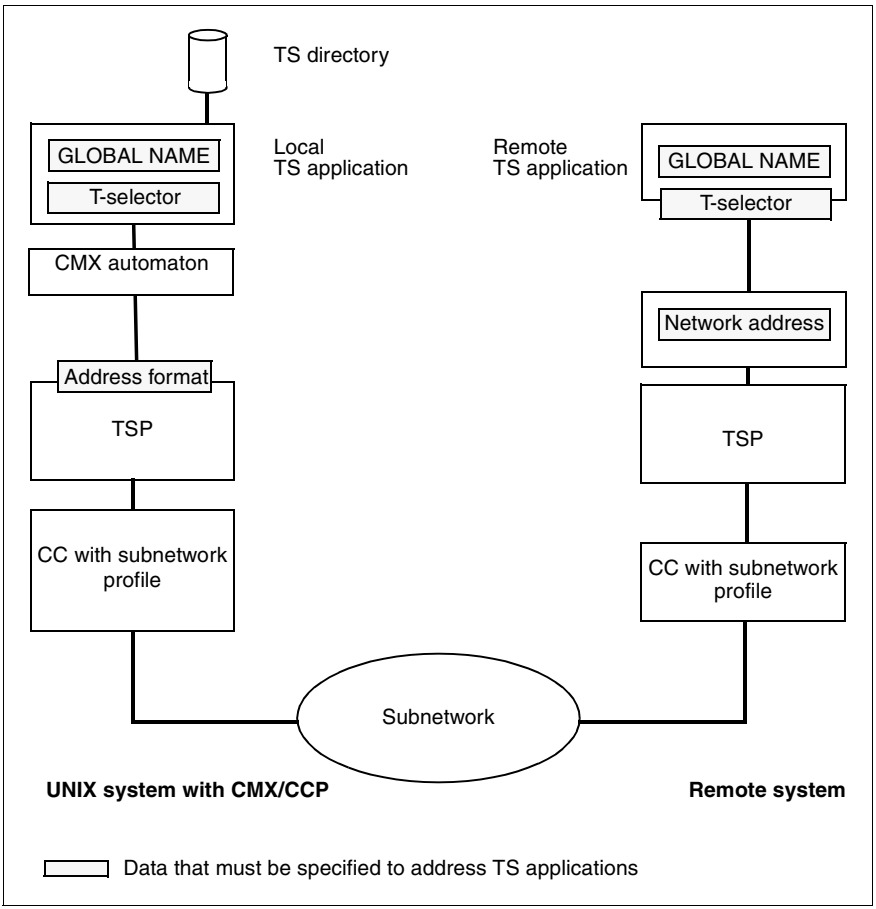


Figure 13: Addressing in the TNS

4.2 Addressing partner systems in the FSS

The Forwarding Support Service (FSS) supplements the addressing functions of the TNS. The TNS names the communication partners and supplies their addresses. The FSS manages the properties of the route to the partner. Such properties include

– the DTE address or call number via which an NEA partner can be reached

– X.25 facilities (e.g. charge reversal) that can be negotiated with an X.25 partner

Typical FSS entries relate to the local network address (e.g. the local NEA address), remote network addresses (e.g. remote NEA address), addresses in subnetworks (telephone network, X.25 network, etc.) via which the remote partner is reached, as well as the physical lines (subnetwork ID) that provide access to the subnetwork. All of this information and the respective interrelationships are stored in an FSS configuration. When a connection is established, the transport service providers (TSPs) access the information in the FSS configuration.

Every piece of information you want to enter is stored by your system's *Forwarding Support Service* as an object of a particular class. So for example, a remote network address (NSAP address) is linked to an NSAP object, a remote subnetwork address (SNPA address) and the routes associated with a remote SNPA address (i.e. the combinations of remote SNPA address and associated local subnetwork interfaces) are linked to an SNPAROUTES object. The entries in the CMXCUI are based on these object classes.

You create and manage the active FSS configuration using the CMXCUI (see chapter "Configuration and administration in the menu"). You can also edit the corresponding *forwarding support configuration file* in *fsconfig* format (see section "Creating FSS configuration file (fsconfig format)" on page 126). However, you should only do this if you have to define a large volume of configuration data.

The following overview shows the assignment between the most commonly needed address information and the FSS object classes and CMXCUI entries:

Address information	FSS Object Class	Entry on CMXCUI
Local network address (NSAP)	LOCNSAP	Local - Local Host...
Route (combination of local subnetwork ID + remote SNPA)	SNPAROUTES	Route - Routes to Remote Subnetwork Interfaces...
Remote network address (NSAP)	NSAP	NSAPs - Remote Hosts...

Table 2: Allocation of address information to object classes

Below you will find a description of how this information can be managed using the individual object classes. The possible attributes for each of these object classes are described in detail in the section “Overview of object classes and their attributes” on page 108.

4.2.1 Network addresses

Local network addresses

The local network address must be configured for each network (NEA, OSI, Internet) via which your local system is to communicate. In the FSS configuration file, enter the local network addresses using the object class LOCNSAP (see section “Overview of object classes and their attributes” on page 108).

Network addresses of partner systems

You configure the partner systems you want to communicate with by first of all entering their network addresses. In the CMXCUI, you do this in the “Remote Hosts” window, and in the FSS configuration file, you specify NSAP objects.

4.2.2 Subnetwork interfaces and routes

For an active connection setup to a remote system, a route in the subnetwork (e.g. X.25 subnetwork) must be assigned to one of the network addresses of the remote system (e.g. the IP address). Only then can be remote system be reached. A route is defined by its starting point and its endpoint. The starting point of the route is a local subnetwork interface; the endpoint is the subnetwork interface of the partner system or of the next intermediate system.

Routes with the same endpoint and equivalent starting points need not be distinguished from each other, rather are configured simultaneously as a group. On the local side, this is done by specifying the subnetwork ID used to combine the equivalent local subnetwork interfaces. For the partner side, the remote subnetwork interface is entered in unchanged form as the endpoint of the route. In the FSB, such equivalent routes are represented by an SNPAROUTES object.

Assigning the subnetwork ID

You must assign a subnetwork ID for each local subnetwork interface. If two subnetwork interfaces provide access to the same subnetwork, you can assign the same subnetwork ID to both (this is done in the configuration files of the subnetwork profiles, see the manuals “CMX/CCP, ISDN Communication” [3] and “CMX/CCP, WAN Communication” [4]). It is then up to the system to choose the outgoing subnetwork interface during the active connection setup.

If you only want to establish connections to partners via specific subnetwork interfaces, assign different subnetwork IDs to two subnetwork interfaces leading to the same subnetwork.

Facilities

Certain facilities (e.g. X.25 or ISDN facilities like charge reversal, closed user group) can be assigned to each route and each remote subnetwork interface. These facilities are defined in a FACIL object and assigned to a route or remote subnetwork interface. By configuring these facilities in the FSS, individual or combined facilities can easily be set, queried, and modified during operation.

4.2.3 Determining the route

When a transport connection is established, the CMX automaton and the TSPs determine the communication path between a local and a remote TS application. They do this by making selections and producing maps. In the following diagram it is assumed that there is as yet no network connection to the host on which the remote TS application is running.

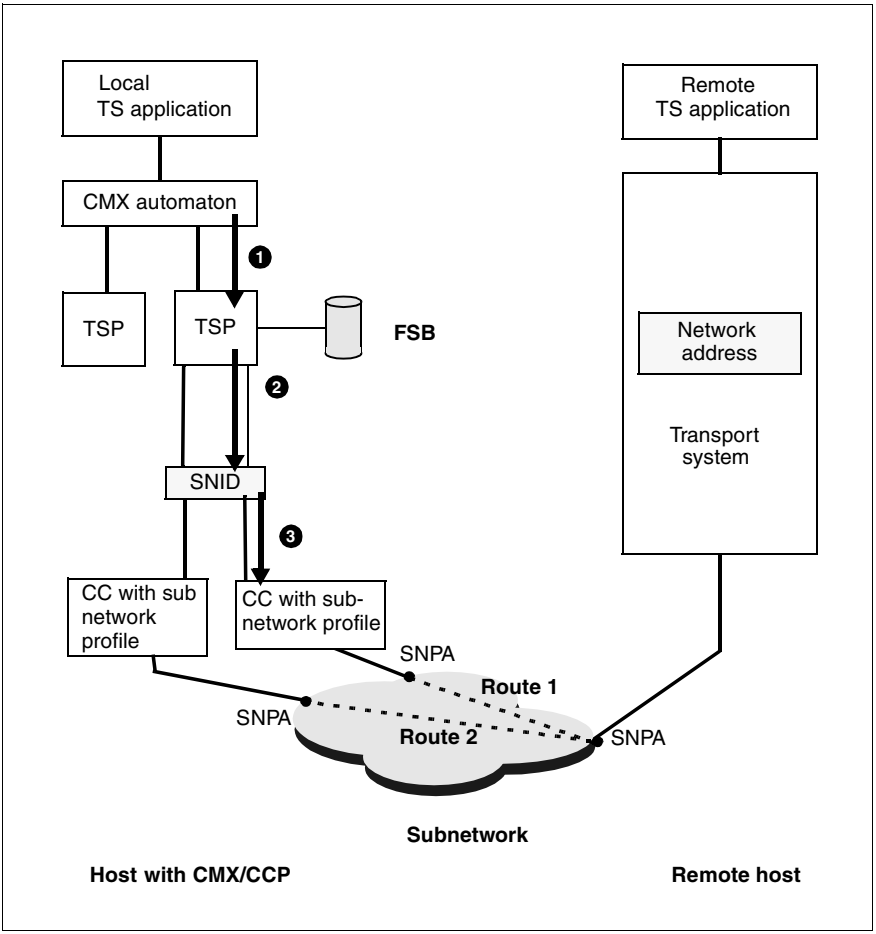


Figure 14: Concept of establishing routes

The relationships indicated by the arrows in figure 14 are described below.

(1) Selecting the TSP

The TRANSPORT ADDRESS contains, in coded form, the network (NEA, OSI, Internet) via which the transport connection is to be set up. The CMX automaton uses this information to select the appropriate TSP.

There are usually several subnetworks available to the TSP for setting up a network connection. A subnetwork constitutes all the resources (e.g. transmission paths, switching centers) which allow communication within a group of systems, for example a Datex-P network, an Ethernet segment, or a dedicated line. Each of these subnetworks has one or more access points *(subnetwork points of attachment, SNPA)*, which are identified by SNPA addresses (= subnetwork addresses). An SNPA address can be, for example, a PVC number in a Datex-P network, the address of an Ethernet connection, or a combination of a CC identification and the number of a line of this CC.

The local subnetwork interfaces (SNPAs), which provide access to the same subnetwork, should be regarded as equivalent, since the same remote systems can be accessed via all of them. You must allocate a common symbolic identifier, the subnetwork ID (SNID) to each such group of local SNPAs. The SNID describes the type of subnetwork involved (see the above example) and the group of access points to this subnetwork. An SNID stands, for example, for two connections to the public Datex-P network or for two dedicated ISDN lines which lead to the same remote system.

(2) Mapping remote network address to subnetwork ID and remote SNPA address

To establish a network connection, the TSP must set up a route. To do this, it derives the SNID and the remote SNPA address from the remote network address (which it reads from the TRANSPORT ADDRESS). Depending on the CCP profile, the TSP takes one of the following approaches:

– The remote network address already contains the remote SNPA address and specifications from which the TSP can construct the SNID.

– An routing protocol is used within the network. This means that each system regularly informs the others of its network address and SNPA address. In this way, each protocol entity compiles an address book, which helps it to map network addresses to SNPA addresses when establishing the network connection.

– The TSP maps the remote network address to SNID and remote SNPA address by accessing a database (see below).

(3) Selecting the SNPA

If the SNID identifies a group consisting of more than one local SNPA, the TSP must select a local SNPA from this group. It then uses this SNPA to set up the new route and hence the new network connection.

4.3 Connection setup process

In the following you will see an example of how the information from the various databases flows together. The example illustrates the process of establishing a connection via an NEA network. The numbers in the black circles refer to subsequent descriptions of the steps involved in accessing the configuration information.

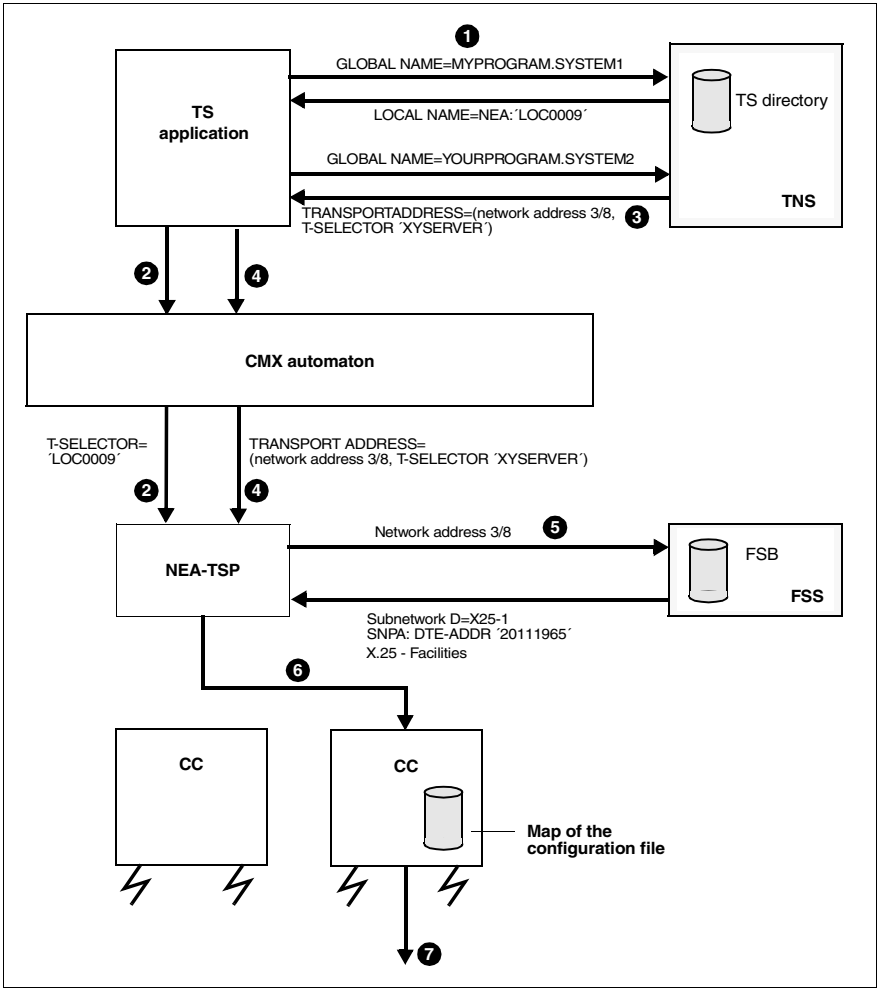


Figure 15: Establishment of transport connection via TRANSDATA NEA network

Accessing the configuration data

1. With help from the TNS, the TS application creates its GLOBAL NAME and maps it onto its LOCAL NAME, which in this case only contains a T-selector for the NEA network.
2. The TS application logs itself in using this LOCAL NAME.

3. The TS application asks the TNS for the TRANSPORT ADDRESS belonging to the GLOBAL NAME of the partner application. This consists of the NSAP address of the remote system and the T-selector of the application.
4. The TS application passes this TRANSPORT ADDRESS to the TSP when the connection is requested.
5. It is assumed that no network connection exists yet to the remote system specified by the NSAP address. The TSP passes to the FSS the network address of the remote system, which is a part of the TRANSPORT ADDRESS (see definitions of terms above) and receives the SNID and the remote SNPA address, which in this case is a DTE address (PVC number in a Datex-P network). The TSP also receives the X.25 facilities that have been configured for this ROUTE.
6. By comparing the SNID, the TSP determines the CC via which the connection is to be set up. When the CCs are being loaded, the TSP finds out which SNIDs are assigned to which CCs. The information comes from the configuration file of the subnetwork profile. The CC selects a line from the line group identified by the SNID, and sends the connection request to the corresponding CC.
7. The subnetwork profile on the CC sets up a subnetwork connection. As soon as this exists, the TSP builds a transport connection via it.

Configuring transport connections

Using the CMX user interfaces, you configure the applications and network partners and set operating parameters for TSPs and controllers for your own system. You have two options here:

– You can work with the character-oriented menu (CMXCUI).

– You can enter configuration data via the command interface and then make this data known to the system (fssadm, tnsxcom).

Configuring using an editor

In some areas of application it is advantageous if the configuration data can be entered in special formats via text files rather than using the CMX menu. This is always the case when a large number of similarly named objects is to be defined (mass configuration). For this reason, you have the alternative of entering configuration data for the TNS and the FSS via files. This approach requires specialist knowledge and is described in the chapter “Configuration in expert mode” on page 69.

5 Installation and startup

5.1 Installing CMX

The CMX product and the CCPs (Communication Control Programs) that build on CMX consist of several software packages (in accordance with UNIX SVR4).

Under Solaris, CMX can be installed, updated and deinstalled during normal system operation without having to reboot the system. CMX can also be installed in a Solaris system by means of Live Upgrade (see page 54).

The CMX/CCP communication software is installed from CD using the *Web Start Wizard*. This wizard offers two installation options.

– Installation based on product selection in which all packages of the product are installed

– Custom installation in which customers select the product components they want to install to suit their specific communication infrastructure

Individual product components can be installed later without impairing normal CMX operation when extending the communication profile. It is also possible to install individual packages but this requires expert knowledge of the dependencies between the packages.

Before performing installation, update or deinstallation, you must close all CMX applications and stop all components affected. Otherwise the process is aborted.

After successful installation/updating you must start the software, see page 56.

Please refer to the Release Notice for more information on hardware and software dependencies and installation.

Proceed as follows when installing CMX using the *Web Start Wizard*.

► Place the installation CD in the CD-ROM drive.

  The installation CD is automatically mounted.

► Click on one of the README icons to read the CMX Release Notices before you start installation.

► After you have read the Release Notice, click on the *Installer* icon.

  A welcome window is displayed.

- Click on *Next*.

  A series of windows guides you through the installation procedure. The Installation Summary window is displayed at the end of the installation procedure.

- Click on Details to display the installation log, which provides information on whether or not installation was successful, and in the event of an error, information on how to proceed and detailed diagnostic information.

5.2 Live Upgrade and CMX

Solaris Live Upgrade [Sol_LU] is a procedure for establishing an alternative system environment (including upgraded Solaris and other system software) on an existing system. This method of upgrading a system has following advantages. System downtime is reduced to the minimum and all existing user and configuration data on the system are retained. Live Upgrade copies the entire current system environment into the new, replacement environment prior to upgrading any software.

If the CMX product is already installed in the current environment, the copy process of Live Upgrade transfers it to the new environment. CMX is **Live Upgrade-capable**; in other words, the CMX version can be updated in the new boot environment during Live Upgrade.

If Solaris is updated in a Live Upgrade, e.g. from Version 8 to Version 9, you must install the corresponding CMX version in the updated boot environment, also by means of Live Upgrade. Otherwise, the installed CMX will not be started.

Live Upgrade installation of CMX

The installation CD supports Live Upgrade as an alternative installation option during normal system operation.

Please refer to the Release Notice for more information on Live Upgrade installation.

Synchronizing CMX files

The first step during a Live Upgrade is to copy all system filesystems. Any subsequent modifications made to the filesystems in the new boot environment, are thus lost once the new environment is activated (reboot).

You should note the following:

1. The CMX configuration and the CCPs are stored in the system file system.
   - ► To avoid a Live Upgrade causing loss of configuration changes, do not modify the CMX-/CCP configuration once the Live Upgrade has started; wait until the new boot environment is activated.
2. Start and shutdown of controllers as well the individual components of the software configuration CMX/CCP are stored in log files.
   - ► To ensure a complete log of these components during a Live Upgrade, add the synchronizing option *-s* when activating the new boot environment.

     This makes sure that the changes made in the old boot environment are adopted in the new one.
   - ► To ensure that the log files are also transferred, add the following entries to the file */etc/lu/synchlist*:

```
/var/opt/SMAWcmx/adm/log            OVERWRITE
/opt/SMAW/SMAWcmx/lib/ccp/diagfiles OVERWRITE
```

5.3 Operating the product

Starting

The installed components are started automatically when the system is booted. If you have installed or updated software during normal system operation, you can start the software directly using the *cmx boot* command without having to reboot the system.

Once you have started CMX, you can use its function and start its applications.

Stopping

You can stop the software during normal system operation using the *cmx shutdown* command. Before you do this, you must detach all applications attached to CMX or alternatively you must terminate these applications.

Stopping the software is a prerequisite for deinstalling or updating the communication software.

Limits

The maximum number of processes, communication applications, and connections supported by CMX throughout the system depends on the system platform and the revision level of CMX. The applicable values can be found in the Release Notice.

The maximum values set in your installation can be queried using the *cmxinfo* command (see section “Information on CMX configuration (cmxinfo)” on page 239). It is possible to reduce the preset values using the *cmxtune* command (see section “Changing limits for the CMX automaton (cmxtune)” on page 290); however, this is only recommended if you are optimizing the memory within the context of agreed tuning measures.

The maximum number of transport connections supported by a single TSP likewise depends on the system platform and the revision level of the respective product. The values for the TSPs RFC1006 and Null Transport (NTP) can be found in the CMX Release Notice, and in the Release Notice of the respective CCP product for the other TSPs. Note that you may have to adapt system parameters in Solaris if the number of transport connections operated simultaneously exceeds a certain quantity. Information in this regard is also provided in the Release Notice.

Online man pages

CMX supplies online man pages in English. This means that information on CMX commands and file formats can be displayed on the screen.

6 Configuration and administration in the menu

As the system administrator, you have the option of configuring and administering your software and the desired communication applications in the CMX menu (CMXCUI).

We recommend that you configure and manage the system using the CMXCUI.

The character-oriented menu CMXCUI is based on the FMLI interface.

Before you carry out the configuration, you must decide which communication paths you want to implement for your system. You should create an address plan for your network so that you know the respective systems and TS applications when you are configuring the addresses.

6.1 Overview of the character-oriented user interface CMXCUI

The CMXCUI is a user interface via which you can administer and configure your system and access information. The user interface is available in both English and German.

Each time you call CMX, the system checks which CCP products and subnetwork profiles are installed on the system. The CMX main menu is displayed, from which you can reach the subordinate menus.

This section provides a description of the menu interface and an overview of the menu options.

6.1.1 Menu interface

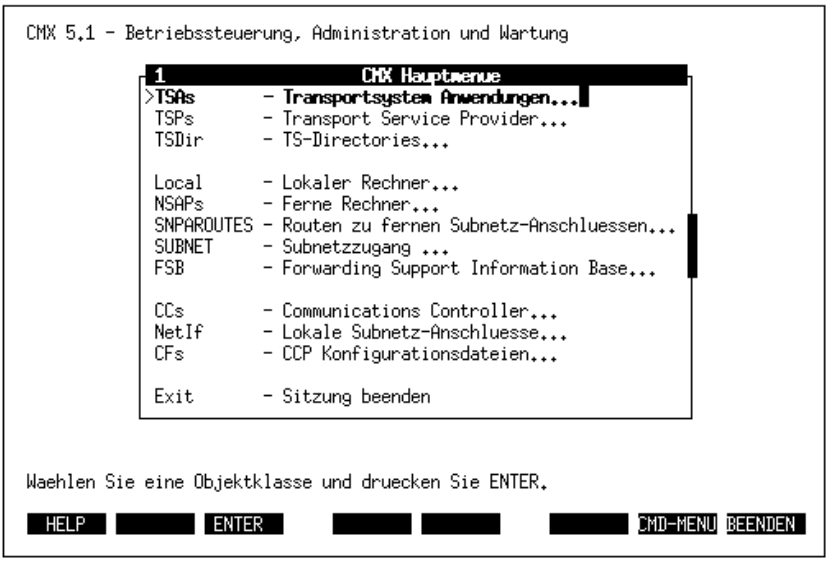


Figure 16: CMXCUI main menu

The function keys F1 to F8 on your keyboard enable you to use the functions that appear at the bottom of the menu. Some of these keys are assigned differently in the submenus. Within help screens, for example, the F2 and F3 keys might represent the functions NEXTPAGE+ and PREVPAGE-. A brief overview of the functions is given below.

In order to use the function keys F1 to F8, an appropriate terminal emulation must be available. If you have problems with the terminal emulation, the key combination CTRL F n can be of assistance.

F1 is generally assigned the HELP function. Pressing it displays information on the current entry.

F2 is assigned the function CHOICES, MARK, or ADD and displays the options you can select. In some of the submenus you can use this key to mark entries for multiple selection. If you get the MARK key on your screen you must always select the desired entry with F2. Otherwise you can select it with F3 ENTER.

Multiple selection is not possible with all actions (e.g. for “Edit”). In this case the action is only performed for the first object marked.

F3 means either ENTER, CONFIRM, or SAVE. In the first case, you can select a menu item, in the second you can confirm and save your input.

F4 is assigned the EDIT or NEW function in certain submenus.

F5 is assigned the DELETE function in certain submenus. The entry in which the cursor is positioned is deleted.

F6 is assigned the CANCEL function, which closes the current window and returns you to the next menu up. This key cancels any entries you have made which have not been saved. It is only effective within the current window. Entries in submenus remain untouched.

F7 means CMD-MENU.

F8 means EXIT.

If F8 is displayed, you can use it to switch to the alternative or the standard function key map.

In the following forms you are automatically switched to the alternative function key map:

– in forms in which the field contents can be modified interactively

– in forms in which a referenced object can be newly generated

– in forms containing a list of objects of the same type

In these cases the keys have the following functions:

Form type	**F2**	**F3**	**F4**	**F5**	**F7**
Modify	CHOICES	CONFIRM	EDIT		CMD-MENU
Add	CHOICES	CONFIRM	NEW...		CMD-MENU
List	ADD	CONFIRM	EDIT	DELETE	CMD-MENU

Table 3: Alternative function key map

F1 is always assigned the HELP function; F6 is always CANCEL.

To create a new entry you must press ENTER to confirm the form/menu with the existing objects without marking an object. As soon as you mark an object the menu option *Create* becomes inactive.

Actions which cannot be selected within a form/menu are grayed out.

6.1.2 Menu options

This section briefly describes the options of the CMX menu and their respective actions. The main menu is divided into three work areas for:

– configuring applications

– configuring partner systems

– configuring and administering lines and interfaces

The options available for configuring partner systems are oriented towards the object classes defined for the forwarding support base (FSB). Each entry in this area corresponds to an object of the FSB. For more details, see chapter “Architecture of Solaris communication” on page 21 and section “Operating the product” on page 56 in this manual.

TSAs - managing transport system applications

You can manage local and remote TS applications using the *Transport System Applications*... menu option.

Using the *Delete TS application* option, you can delete selected entries for TS applications from the TS directory.

The *Display properties*... option enables you to display the properties of TS applications. Here you must specify whether the GLOBAL NAME is to contain “exactly” or “at least” the specified name parts.

In the latter case, the system also selects TS applications which have more name parts on lower levels of the name hierarchy. See section “Address information in the GLOBAL NAME” on page 39. As a placeholder you can enter * or ?.

When you have filled out the form, the relevant TS applications are displayed together with their properties in a text window. This information can also be written to a file.

The same rules apply to the *Display GLOBAL NAMEs*... option as to displaying properties. The names of the TS applications which fit the specified pattern are displayed in a text window.

With the *Assign LOCAL NAME* ... option you can allocate a name to any local TS application.

With the *Assign TRANSPORT ADDRESS* ... option you can assign TRANSPORT ADDRESSES to remote TS applications.

TSPs - transport service providers

The *Transport Service Providers*... menu option gives you a table containing the existing TSPs, plus their TSP type, version and status. To select a TSP, press the ENTER key.

You can activate *(Start)* or deactivate *(Stop)* the selected TSP.

Note that using the *Stop* option aborts all communications activity run by the deactivated TSP.

The *Restart* action allows you to reactivate your TSP after you have changed your configuration.

You can also set or cancel an automatic start for the selected TSP when the system is booted *(Prepare automatic start* or *Cancel automatic start)*.

TSDir - managing TS directories

Information on TS applications is kept in TS directories. Each time you work with the menu, one of these directories can be used. Various actions are possible with TS directories, e.g. they can be saved to a data volume. The default directory is called *DIR1* (see section “Managing TS directories” on page 75) and is used for current operation of TS applications.

You can use the menu option *Create a backup copy*... to back up an existing TS directory.

Using the menu option *Restore a backup copy...,* you can restore a TS directory which you backed up earlier. You can also convert a text representation of the TS directory entries in *tnsxfrm* format into a TS directory (see chapter “Configuration in expert mode” on page 69).

The *Delete TS directory DIRn* option deletes an existing TS directory.

If you delete TS directory *DIR1*, you may find that CMX applications terminate prematurely due to problems with mapping names to addresses.

Using the *Make DIRn working TS directory* option, you can switch to an alternative TS directory for the current session.

Using the option *Exchange with TS directory DIR1*, you can declare an alternative TS directory, e.g. *DIR3*, to be the default directory *DIR1* for the current session, and at the same time the original TS directory *DIR1* becomes the alternative TS directory *DIR3*.

Note that swapping the directories may cause current applications to abort.

The option *Show detailed information* calls up information on TS directories. This information contains the date of the last modification and statistical data on the name parts and properties in the TS directory.

Local - displaying and modifying local hosts

The *Local - Local Host*... option allows you to display the network addresses of your own system (see also “Network addresses” on page 46). The format of the possible addresses is described in section “Address components and their formats” on page 86.

NSAPs - entering remote hosts

This option allows you to create, modify, delete and display the network addresses and attributes of partner systems (see also “Network addresses” on page 46). You must allocate to each partner system a symbolic name via which you can later access the stored information.

SNPAROUTES - defining routes to remote subnetwork interfaces

This option allows you to enter routing information required for reaching the partner systems. This includes the subnetwork ID, which determines the local subnetwork interfaces via which the partner system can be reached, as well as the subnetwork address of the partner system (see also section “Subnetwork interfaces and routes” on page 47). You can also configure facilities for a route. You can create, modify, delete and display routes.

SUBNET - subnetwork access

The *SUBNET - Subnetwork access...* option allows you to define access protection for each subnetwork of your system. Refer to the manuals "CMX/CCP, ISDN Communication" [3] and "CMX/CCP, WAN Communication" [4] for more information.

FSB - saving and reentering the FSB

The *FSB - Forwarding Support Information Base* option allows you to save and reenter the information stored in the FSB on partner systems and on the local network address.

Before you change your configuration you should save your data using the *Create a backup copy*... option.

Re-reading means that an FSS configuration file is read in as a new FSB configuration. A new configuration number is allocated and this new generation is defined as the current configuration. The data to be read in must be in text format (see section “Creating FSS configuration file (fsconfig format)” on page 126). Once it has been read in, it is converted to binary format.

CCs - Communications Controller

The configuration files are assigned and loaded via the *CCs - Communications Controller*... menu option.

The following actions are possible:

– calling information on the CC hardware
– loading a CC
– unloading a CC
– changing the configuration
– producing a memory dump for the CC
– changing to expert mode to receive information for diagnosis

The CC is identified by means of a letter (W for WAN and ISDN) and a digit.

Show Detailed Information

This option gives you information on the CC hardware, e.g. memory size, hardware and firmware version, etc.

Load CC

This option allows you to load the assigned subnetwork profile and to start up the configuration file.

Caution!

Using this option aborts current communication activity.

Unload CC

This option allows you to deactivate the subnetwork profile currently loaded on the CC.

Caution!

Using this option aborts current communication activity.

Change Configuration

This option allows you to assign a subnetwork profile and a configuration file to the CC.

Dump

When you select this option, a form appears in which you can enter parameters for producing a dump. A dump must be edited in expert mode (see the bstv command *format* in the manuals “CMX/CCP, ISDN Communication” [3] and “CMX/CCP, WAN Communication” [4]).

Enter expert mode

This option takes you into interactive mode, in which you can enter commands for diagnosis. The possible commands and their parameters depend on the subnetwork profile and are described in the appropriate manuals (see “CMX/CCP, ISDN Communication” [3] and “CMX/CCP, WAN Communication” [4]).

NetIf - Subnetwork Interfaces

This menu option enables you to display all subnetwork interfaces configured on your system. You will receive a list of all existing CC identifications and the associated configured subnetwork interfaces on the CCs. When you have selected a subnetwork interface with ENTER, you can activate or deactivate the subnetwork interface in the menu that follows.

CFs - CCP Configuration Files

This option enables you to reach the CCP-specific menu, with which you can configure your subnetwork profiles. You can create, modify, delete, print, save and re-read in CCP configuration files. For how to do this, see the manuals for the CCP products (“CMX/CCP, ISDN Communication” [3] and “CMX/CCP, WAN Communication” [4]).

Exit - Quit Session

This option takes you out of the CMX menu.

6.1.3 The configuration procedure

When you have installed CMX (see chapter “Installation and startup”), you will have a basic configuration on your system and your local network addresses will be known to the system. With the first configuration you are advised to proceed as follows:

– create CCP configuration files
– allocate configuration file and subnetwork profile to a CC (change configuration)
– load CC
– define route to remote subnetwork interfaces
– enter remote network addresses
– enter TS applications

7 Configuration in expert mode

This chapter provides information on the CMX command interface for configuring TS applications, network partners, and local routes. The commands *tnsxcom* for configuration in the Transport Name Service (TNS) and *fssadm* for the Forwarding Support Service (FSS) are described here.

While you manage the TS applications and their TRANSPORT ADDRESSES using the TNS, you can administer the network addresses of end systems as well as routes to these systems using the FSS.

In the configuration files of the subnetwork profiles and with the help of facilities entries in the FSS (object class FACIL), operating parameters are defined for protocols on Layers 1 through 3a, e.g. the transmission speed, values for monitoring times. In the FSS, such facilities can be permanently assigned to particular routes (SNPAROUTES object class).

Please note that a subnetwork ID must be defined for each subnetwork interface in a configuration file (KOGS for Communication Controller). The subnetwork ID must then be entered when configuring the routes in the FSS.

If a subnetwork profile is loaded on a CC, the contents of the configuration file are also loaded. The subnetwork profile running on the CC accesses the contents of this map of the configuration file.

A detailed description can be found in the manuals “CMX/CCP, WAN Communication” [4] and “CMX/CCP, ISDN Communication“ [3]. The section “Configuration procedure” explains the configuration of TS applications in the TNS. The section “Configuring with tnsxcom” on page 75 documents the configuration of network addresses and routes, as well as partner-specific operating parameters in the FSS.

7.1 Configuration procedure

The following two sections provide an overview of the steps involved in configuring with *tnsxcom* and *fssadm* on the command line interface.

7.1.1 TNS: application-specific configuration

You can create a TNS configuration file in *tnsxfrm* format using any editor, and create a TS directory from this file in command mode or using the “Restore a backup copy” option in the CMXCUI. The syntax of this file, which is to serve as a database for a TS directory, is described in the section “Syntax of the TNS configuration file” on page 77. The main steps involved in configuring in the TNS are outlined below.

- Check the configuration file.

  Use *tnsxcom -s␣file* to perform a syntax check on the *file* file (check mode). The TNS logs any syntax errors. The TS directory remains unchanged.

- Create a new TS directory.

  Use the *tnsxcom␣-l␣file* command to insert the entries from the *file* file into a previously empty TS directory (load mode).

- Check and update the TS directory.

  Use *tnsxcom -S␣file* to check and update a TS directory (check/update mode). As with the -*s* option, the syntax check is first run on the entire *file* file. If no syntax errors are detected in *file*, then *tnsxcom* updates the TS directory in a second run.

- Expand the TS directory.

  Use *tnsxcom -u␣file* to add the entries from the *file* file to an existing TS directory (update mode). If *file* contains entries for GLOBAL NAMES that already exist in the TS directory, the old entries are overwritten or supplemented.

- Administer interactively.

  Instead of preparing the entries in a *file* file and then transferring them to *tnsxcom*, you can also edit the configuration file interactively after calling *tnsxcom -i*. Apart from entering new GLOBAL NAMES, the following input is also possible in interactive mode:

  – Delete entry for TS application from the TS directory.

    You delete the entire entry of a TS application from the TS directory using the command:

    ```
    Global_name␣DEL
    ```

All properties assigned to the TS application *Global_name* and the GLOBAL NAME itself are deleted from the TS directory. The TS application is then no longer known to the TNS.

– Display the properties of a TS application.

With the following command, the entries of a TS application that have been entered or modified can be displayed on the screen for checking:

```
Global_name␣DISP
```

– Switch TS directory.

Use the following command to switch to the file in another TS directory:

```
DIR␣n
```

For *n*, specify the number of the TS directory you want to switch to.

The following input records in the file then relate to this TS directory. This file is edited until you explicitly switch to another TS directory or terminate input with CRTL D.

► Select TS directory.

TNS supports up to 9 different TS directories. All actions listed above refer to the directory DIR1. Using the *-d␣* num (num=1,2...9) option, you can also edit other directories.

7.1.2 FSS: partner-specific configuration

A partner-specific configuration means modifying or extending the FSB configuration that was created when CMX was installed.

For the first configuration, it is advisable to create an FSS configuration file. The precise syntax of the entries in the configuration file is described in section “Creating FSS configuration file (fsconfig format)” on page 126.

A Forwarding Support Information Base (FSB) is then generated from the configuration file. A number of such FSB configurations can be generated; only one FSB configuration is used by the FSS at any one time. This FSB is then called the active FSB configuration.

Before you perform the configuration, you must decide which communication paths you want to implement for your system. You should create an address plan for your network, so that you know the addresses of the relevant systems and TS applications during the configuration.

The following two procedures are recommended for extending the FSB configuration:

1. Create and edit the configuration file, then create and activate the FSB configuration.
2. Modify the configuration entries directly in the active FSB configuration.

The two methods are described in detail in the following sections.

Creating and editing the configuration file

Using the *fssadm* command, a configuration file can be created from an existing FSB configuration during operation. This configuration file can be edited using an editor, and a new FSB configuration can then be created from the edited file. This configuration can either be activated immediately or with the next system start.

The object classes and attributes are described in detail in section “Configuring with fssadm” on page 104.

For the method outlined above, carry out the following steps:

- Query the existing FSB configurations with

  ```
  fssadm get FSBGEN
  ```

- Select the number of the FSB configuration you want to modify and create a configuration file from this FSB configuration using the command

  ```
  fssadm create config-file gen-nr=value path=value
  ```

  gen-nr is the selected FSB configuration; *path* is the path name under which the configuration file will be stored.

- Edit the configuration file by entering the desired data. In so doing, observe the syntax rules, which are described as the file format *fsconfig* in section “Creating FSS configuration file (fsconfig format)” on page 126.

- Check the configuration file for input errors using the command:

  ```
  fssadm check config-file path=value
  ```

  path is the path name of the configuration file. If an error is detected with the *fssadm check* command, correct the error recorded in */var/opt/SMAWcmx//adm/log/fsin_log* and reenter the command.

► Create a new FSB configuration with

```
fssadm create FSBGEN gen-nr=value path=value
```

gen-nr is the number of the new FSB configuration.

If the modifications do not result in inconsistent data which may still apply in particular communication components, and if there is no danger of aborting existing network or transport connections, the FSB configuration can be activated immediately using the command:

```
fssadm set FSBGEN gen-nr=value use=ACTIVE
```

If, for example, you have changed the local network addresses for TRANSDATA NEA and OSI TP0/2, you must subsequently restart these TSPs.

► Use the command

```
fssadm set FSBGEN gen-nr=value use=NEXT-ACTIVE
```

to set the new FSB configuration as the one to be activated with the next FSS or system start.

► Then restart the FSS and the modified communication components.

Modifying the configuration during operation

You can enter the partner-specific data directly into the active FSB configuration during operation. These configuration modifications are effective immediately. However, please note the following:

As a result of the modification, information that has already been used to establish existing network or transport connections and is still stored in the protocol entities, may be inconsistent with the new information entered in the FSB. Depending on the object and attribute affected, this may mean that a network or transport connection is not established or that the new information is ignored for a certain period or until the next system start. An example would be the local network address for NEA (LOCNSAP attribute nea-addr) or for OSI TP0/2 (LOCNSAP attribute osi-addr). If you have changed either of these addresses, you must subsequently restart the corresponding TSP.

- To modify configured objects, enter:

  `fssadm set` *objectclass attribute*

  Example:

  ```
  fssadm set SNPAROUTES name=route1 subnet=X25-1 dte-addr=
  23456
  ```

- To create new objects, enter:

  `fssadm create` *objectclass attribute*

  Example:

  ```
  fssadm create NSAP name=partner1  internet-addr=
  129.22.11.8 \
     net=INTERNET snpa-list=route1
  ```

- To delete configured objects, enter:

  `fssadm delete` *objectclass attribute*

  Example:

  ```
  fssadm delete NSAP name=partner1
  ```

  If an error occurs when executing the *fssadm* command, evaluate the error message and issue the corrected command. As soon as *fssadm* has been executed without error, the configuration modifications entered become effective.

The relevant object classes and attributes are listed in section “Overview of object classes and their attributes” on page 108. Detailed information on the *fssadm* command and the *fsconfig* file format is provided in section “Configuring with fssadm” on page 103.

7.2 Configuring with tnsxcom

TS directories are created, updated, and read on shell level using the TNS compiler tnsxcom. tnsxcom compiles input records, which you transfer in *tnsxfrm* format to the compiler, into the format of the TS directory and enters the created entries into the TS directory. tnsxcom also reads the entries of the TS directories and compiles these into a printable format. tnsxcom is called using the *tnsxcom* command. The syntax of *tnsxcom* is outlined in the command catalog (section "TS directory: create, update, output (tnsxcom)" on page 307). This section describes:

– which actions can be executed with tnsxcom

– how you must enter the properties that are to be included in the TS directory (format of input for tnsxcom)

– the format in which the addresses and T-selectors must be specified for the various transport systems

– how GLOBAL NAMES in the input records can be expanded en bloc, e.g. if you want to make changes to a branch of the naming tree (see section "Names with the same high-order name parts" on page 96)

– how tnsxcom operates

7.2.1 Managing TS directories

The TNS supports up to 9 different TS directories simultaneously with the identifications 1-9. The TS directories are stored in the file system as the directories *DIR1*, *DIR2*, ... *DIR9*. TS directory *DIR1* is the default directory accessed as standard by the TS applications. The other TS directories can be used as backup copies or as experimental versions, for example.

For information on how to create, modify, and query information on a TS directory, turn to section "Input rules for TNS files" on page 95.

The following actions can be executed with tnsxcom:

– Create new TS directories.

  You can create new TS directories DIR<n> (<n> = 1,...,9) using tnsxcom. To do this, you create a file using any editor. In this file you enter all TS applications that are to be entered in this TS directory, together with their properties, in the *tnsxfrm* format of tnsxcom. Then call tnsxcom in load mode (tsnxcom -l *file*). From the records in the file, it creates the entries for the TS directory and writes these into the previously empty TS directory. This TS directory (DIR<n>) must not exist beforehand. In particular, you must not create it with *mkdir*. The new TS DIR<n> is created implicitly by tnsxcom. The files of the TS directory are created by tnsxcom.

– Update TS directories.

  Using tnsxfrm you can incorporate new TS applications into an existing TS directory or delete entries for TS applications from the TS directory. You can assign new properties to TS applications, or modify and delete properties.

  When updating a TS directory, proceed in the same way as when creating a new TS directory and entering the changes in a file. In this case, you must call *tnsxcom* in update mode (see section “Syntax of the TNS configuration file” on page 77). However, you can also update a TS directory by transferring the modifications to tnsxfrm interactively via standard input. To do this, call *tnsxcom* in interactive mode (see section “TS directory” on page 99). The format of the input is the same in both modes. Moreover, the format of the input is independent of whether you are creating a new TS directory or updating an existing one. tnsxfrm must simply be called in the appropriate mode.

– Read TS directories.

  A TS directory primarily comprises non-printable characters. If you want to read a TS directory, you can convert it to a printable format using tnsxfrm and write it to a file. This file can again be used as input for tnsxfrm.

  With this function you can port a TS directory of another computer to your computer. You must write it to a file on the remote system with tnsxfrm and import this file onto your system. Here you recompile the TS directory with tnsxfrm.

  Before compiling, check whether TS applications residing on your system are entered as local TS applications in the entries of the TS directory. The TS applications residing on the remote system must be entered accordingly as remote TS applications.

The TS directory created on the remote system can also be inserted into a TS directory that already exists on your system. The TS directory then has the correct structure for the TNS, even if the initial TS directory was created with an older version of TNS.

Specify particular options in the *tnsxcom* command to determine which of these actions are to be performed by tnsxfrm.

7.2.2 Syntax of the TNS configuration file

All entries to be incorporated in the TS directory must be transferred in the form of the following input records.

```
[global_name␣]type[␣data fields]
```

global_name, *type* and *data* are denote fields. Square brackets indicate that the fields are optional.

The meaning of the individual fields within an input record is described below.

7.2.2.1 GLOBAL NAME

In the *global_name* field you specify the GLOBAL NAME of the TS application to which the property described in the following fields is to be assigned. If you define more than one record for a TS application, the GLOBAL NAME need only be specified in the first record. You can leave the *name* field blank in the subsequent records. The records must, however, follow one another directly. In other words, if a record's *name* field is blank, the value last specified in a record applies.

The GLOBAL NAME comprises 1 to 5 hierarchically arranged name parts *Npi* (i = 1, 2, 3, 4, 5). *Np1* is the name part belonging to the highest hierarchical level (TS_COUNTRY), *Np5* to the lowest (TS_PN). See also section “Address information in the GLOBAL NAME” on page 39.

Uppercase and lowercase characters have different meanings (case sensitivity). If the name parts contain special characters (see section “Characters with a special meaning” on page 95) whose special meaning would cause an ambiguity of syntax, such special characters must be escaped by means of a \ (backslash) or by enclosing them in single quotes.

In case of doubt you should escape all special characters. Wherever an escape is superfluous it is ignored. The maximum lengths for *Npi* are:

Name part *Npi*	1	2	3	4	5
Length in bytes	2	16	16	10	30

Table 4: Maximum length of the name parts

The arrangement of *'Npi's* in *name* is by ascending hierarchy from left to right, the *'Npi's* being separated by . (period) (thus: Np5.Np4.Np3.Np2.Np1). Blank *'Npi's* (i = 1, 2, 3, 4, 5) are permissible but the separator . (period), following them must always be included. For example: .Np4..Np2.Np1

If a GLOBAL NAME ends in at least one . (period), then it is absolute, i.e. it is positioned directly under the ROOT of the naming tree. If it does not end in a . (period), it is relative. Relative GLOBAL NAMES have an origin added if an origin has been defined for them (see also section "Names with the same high-order name parts" on page 96). Valid examples of GLOBAL NAMES are:

Np5
: only name part 5, relative (possibly to ROOT)

Np5.
: only name part 5, absolute

Np5.Np4
: only name parts 5 and 4, relative (possibly to ROOT)

Np5....Np1.
: only name parts 5 and 1, absolute

..Np3
: only name part 3, relative (possibly to ROOT)

.Np4..Np2.
: only name parts 4 and 2, absolute

7.2.2.2 Type of application

The entry for type and properties of the application has the following syntax:

```
type[␣data fields]
```

The value for *type* determines the type of entry (also known as *property*); i.e. *type* specifies whether it is a LOCAL NAME or a TRANSPORT ADDRESS, a *session component*, or a *presentation component*. The value of the property must be specified in the *data* fields, e.g. the TRANSPORT ADDRESS. The individual *data* fields must be separated from each other by spaces.

Possible values for *type* are:

– TSEL for transport selector entry of a local application
– TA for TRANSPORT ADDRESS of a remote application
– PSEL for presentation selector
– SSEL for session selector

Records with which you want to create new entries for TS applications and records with which you want to modify or extend existing entries have the same format. If a TS application or a property of a TS application does not yet exist in the TS directory, a new entry is created from the specifications of the record. If an entry already exists, it is modified in accordance with the specifications in the record.

At least one record must be transferred for each property you want to enter. Each TSEL entry of a LOCAL NAME must be transferred in a separate record, for example. Each record corresponds to a logical line. If it is necessary for a record to span several lines, the line end must be escaped with a \ character (backslash) or the specifications must be enclosed in () (parentheses).

It is not possible to incorporate only the GLOBAL NAME of a TS application into the TS directory without assigning a property to this name.

A description of how to enter the individual properties is given below.

7.2.3 LOCAL NAME

The LOCAL NAME of a TS application comprises one or more TSEL entries (one TSEL entry per transport system via which the TS application is to communicate). You must transfer a record for each T-selector you want to assign to the local TS application *global_name*. The record must be structured as follows:

```
[Global_name␣]TSEL[␣addrform[␣data_field_with_T-selector]]
```

The same specifications are permitted for *addrform* as when entering the TRANSPORT ADDRESS. The value specified in the *data* field is incorporated in the TS directory as the T-selector for the transport system *addrform*. If the TS directory for this transport system already contains a T-selector in the LOCAL NAME, this is overwritten by the new value. If the LOCAL NAME does not yet contain a T-selector for this transport system, this T-selector is added to the previous LOCAL NAME. If the *data* field is empty, an existing T-selector for the corresponding transport system is deleted from the entry in the TS directory.

In this case, neither a warning nor an error message is output if such a T-selector does not exist in the LOCAL NAME. If the T-selector you want to delete is the only component of the LOCAL NAME, the property LOCAL NAME is deleted for this TS application.

The restricted length of the *LOCAL NAME* means that up to eight different T-selectors can be accommodated. T-selectors that are identical for several transport systems only take up one of the eight memory locations. Exceptions to this are the transport systems with *addrform* LANINET and EMSNA, i.e. the T-selectors belonging to these transport systems are always distinct from each other, regardless of their value.

T-selectors can be specified in various forms (see the following examples and section “Address components and their formats” on page 86). Their length is restricted to 10 characters (in TRANSDATA format: 8 characters). The special characters ' (single quote) and \ (backslash) must be escaped with \ (backslash) if they are to be included in the T-selector.

The following table contains the permitted specifications for T-selectors with the various address formats. For information on the meaning of the address formats and T-selector formats, as well as the representation formats for T-selectors, turn to section “Address formats” on page 84.

Address format	T-selector format
EMSNA	LU name, LU number
LANINET	Port number
LOOPSBKA	T-selector
RFC1006	T-selector
SDLCSBKA	Station name
TRSNA	T-selector
WANNEA	Station name
WANSBKA	T-selector
WAN3SBKA	T-selector

Table 5: Address and T-selector formats

Example for LOCAL NAME:

```
Global_name  type addrform T-selector
-------------------------------------------------------------------
loopleer     TSEL LOOPSBKA V''          ; Empty T-selector
laninet      TSEL LANINET  A'4712'      ; Decimal port number
rfc1006      TSEL RFC1006  A'Cologne'
iso          TSEL WANSBKA  E'wansbka'   ; To be stored in EBCDIC
x28          TSEL WAN3SBKA A'WAN3'      ; To be stored in ASCII
wan1         TSEL WANNEA                ; Delete station name
```

7.2.4 TRANSPORT ADDRESS

Records used to define the TRANSPORT ADDRESS of a remote TS application have the following format:

```
[Global_name␣]TA[␣addrform[␣data_fields_with_address_
components]]
```

TA is the indicator for a TRANSPORT ADDRESS.

The address format *addrform* specifies the type of transport system used. When a TRANSPORT ADDRESS is entered, TNS always create an entry for the transport system as well. The address components are transferred in the *data* fields that follow.

If you want to modify a TRANSPORT ADDRESS that is already entered in the TS directory, specify the new address components in the *data* fields. The entry in the TS directory is then overwritten by the new TRANSPORT ADDRESS.

If you want to delete a TRANSPORT ADDRESS from the TS directory, remove the entries for the address components. The TRANSPORT ADDRESS is then deleted from the TS directory for this TS application.

The specifications permitted for the *addrform* address format and the address components to be specified with *addrform* are listed below. Address components in square brackets are optional. The meanings of the address formats and address components, and the format in which you must transfer the address components, are described in section “Address components and their formats” on page 86.

addrform	**Address components**
EMSNA	LU name, processor/region number
LANINET	IP address or *HOST* hostname, port number
LOOPSBKA	T-selector
RFC1006	IP address or *HOST* hostname [*PORT* portnumber], T-selector
SDLCSBKA	Dial number, [WAN CC/line identifier]
TRSNA	Sym-dest-name, T-selector
WANNEA	Station name, processor/region number, [WAN CC/line identifier]
WANSBKA	OSI-NSAP, T-selector [TPI] [TPC] [WAN CC/line identifier]
WAN3SBKA	SNPA information [T-selector] [WAN CC/line identifier]

Table 6: Address formats and associated address components

Example of TRANSPORT ADDRESS

```
Global_name  type addrform Address components
-------------------------------------------------------------------
neate         TA   WANNEA    T'$DIALOG' 1/18 WAN 1:1 2:3
X25          (TA   WANSBKA   X.121 4589004033 ; DTE address(IDI)
                             A'dtxp-33-01'    ; T-selector
                             2/0              ; TPC)
tcp/ip        TA   LANINET   128.0.1.23 A'4711'
rfc1006       TA   RFC1006   HOST D018B016 A'Cologne'
```

7.2.5 Session component

The session component expands the TRANSPORT ADDRESS of a remote TS application into an SSAP address or adds an S-selector to the LOCAL NAME of a local TS application. The SSAP address is the address of a TS application in the Session Layer (Layer 5 of the OSI Reference Model).

The session component of a TS application is transferred in the following record:

```
[global_name␣]SSEL[␣data field with S-selector]
```

If the *data* field is empty, the session component is removed from the TRANSPORT ADDRESS or LOCAL NAME property. If an S-selector already exists for this TS application in the TS directory, this “old” value is overwritten by the specified value. The TS application need not have a TRANSPORT ADDRESS or a LOCAL NAME already assigned to it before the S-selector is entered.

The table below contains the permitted specifications for the S-selector.

S-selector	**Meaning**
SSEL	Delete SSEL entry
SSEL V''	Blank entry for S-selector
SSEL *ssel*	ssel representation formats: A'string': string of max. 16 characters; stored in ASCII E'string': string of max. 16 characters; stored in EBCDIC X'string': even number of hex digits in string; max. 32 T'string': complies with TRANSDATA conventions

Table 7: Permitted specifications for the S-selector

7.2.6 Presentation component

The presentation component expands the TRANSPORT ADDRESS or the SSAP address of a remote TS application into a PSAP address. In a local TS application the presentation component adds a P-selector to the LOCAL NAME.

The PSAP address is the address of a TS application in the Presentation Layer (Layer 6 of the OSI Reference Model).

The presentation component of a TS application is transferred in the following record:

```
[name␣]PSEL[␣data field with P-selector]
```

If the *data* field is empty, the presentation component is removed from the TRANSPORT ADDRESS or LOCAL NAME property. If the TS directory already contains a presentation component for the specified TS application, the “old” value is overwritten by the new one. The TS application need not have a TRANSPORT ADDRESS or a LOCAL NAME assigned to it before the P-selector is entered. If the TRANSPORT ADDRESS or LOCAL NAME does not yet contain a session component, a blank entry (SSEL V”) is automatically generated for the session component when the presentation component is entered.

The table below contains the specifications allowed for the P-selector.

P-selector	**Meaning**
PSEL	Delete PSEL entry
PSEL V”	Blank entry for P-selector
PSEL *psel*	psel representation formats: A'string': string of max. 16 chars.; stored in ASCII E'string': string of max. 16 chars.; stored in EBCDIC X'string': even number of max. 32 hex digits in string T'string': complies with TRANSDATA conventions

Table 8: Permitted specifications for the P-selector

7.2.7 Address formats

When defining the TRANSPORT ADDRESSES and TSEL entries of the LOCAL NAME, you must transfer the *addrform* address format of the transport system to which the subsequent address component or T-selector information relates.

The table below lists the different values for *addrform*, the associated transport system, the corresponding CCP name in the CMX menu, and all associated address components.

Please observe the explanations in section “Syntax of the TNS configuration file” on page 77 where you will find specifications of the address components to be used in each individual case.

The meaning of the address components and their representation formats are described below the table. The address components are listed in alphabetical order.

addrform	Transport system for	CCP profile in menu	Address components
EMSNA	Interconnection with TRANSIT via SNA backbone	TRANSIT LU0	LU name processor/region
LANINET	Host-to-host connection via TCP/IP. 1)	TCP/IP RFC1006	IP address or *HOST* hostname port number
LOOPSBKA	Process-to-process communication with CMX	CMX-LOCAL	T-selector
RFC1006	Host-to-host connection via TCP/IP with RFC1006 convergence protocol 1)	TCP/IP RFC1006	IP address or *HOST* hostname [*PORT* portnumber] T-selector
SDLCSBKA	Station-to-station connection to SNA networks via SDLC	WAN-SDLC	Call number WAN CC/line identifier
TRSNA	Host-to-host connection via transparent SNA network	TRANSIT LU6.2	Sym-dest-name T-selector
WANNEA	Host-to-host connection in WAN and ISDN with protocol NEATE, NEAN	WAN-NEA WAN-NX25 ISDN-NEA ISDN-NX25	Station name processor/region WAN CC/line identifier

Table 9: Address formats, transport systems, and address components

addrform	Transport system for	CCP profile in menu	Address components
WANSBKA	Host-to-host connection in WAN and ISDN with ISO protocols of class 0 and 2 (SBKA profile)	WAN-CONS ISDN-CONS	SNPA information or OSI-NSAP T-selector TPI TPC WAN CC/line identifier
WAN3SBKA	Heterogeneous interconnection in X.25 network without transport protocol	WAN-X.25 ISDN-X.25	SNPA information T-selector WAN CC/line identifier

Table 9: Address formats, transport systems, and address components

The address components that describe the network address of an end system (Ethernet address, IP address, host-name, processor, region, sym-dest-name) are transferred as an alphanumeric character string.

The address components that describe the address of a TS application within the local system (T-selector, station name etc.) are transferred as a string enclosed in single quotes. A format indicator must be specified in front of the string (see the following section).

7.2.7.1 Address components and their formats

This section explains the address components and their representation formats that you specify when entering TS applications.

ETHERNET address

Specify the ETHERNET address of the end system on which the remote TS application resides.

Representation format:

Specify exactly 12 hexadecimal digits [0-9,A-F,a-f].

IP address

Specify the IPv4 or the IPv6 address of the remote end system.

Representation format:

In case of IPv4 specify exactly 4 decimal numbers between 0 and 255. These digits must be separated by the special character. (period).

Example: 123.0.3.98

In the case of IPv6 specify the 128-bit address by up to 8 hexadecimal numbers, each of which represents a 16-bit section, separated by the special character (colon). Adjoining hexadecimal numbers with the value 0 may be compressed as two consecutive colons (::) once for each address. The section in IPv6 addresses derived from IPv4 addresses may be represented by the IPv4 method of representation defined above.

Example:

fe80::280:17ff:fe28:7b08
::ffff:123.0.3.98

hostname

Enter the hostname (with the leading keyword HOST) of the remote system.

Representation format:

Enter max. 60 characters in ASCII format.

Example:

PGTW1339 or V116.mch.sni.de

LU name

For a TS application in an SNA system, enter the VTAM application name of the SNA application. For a TS application in a TRANSDATA computer accessible via an SNA system, enter the station name of the TS application.

Representation format:

The LU name must be transferred as a string (‘LU_name’). Only the TRANSDATA format is permitted; you specify this with the format indicator T. The station name must not be more than 8 characters long. Blanks are automatically added to shorter entries. Longer entries are rejected as errors.

LU number

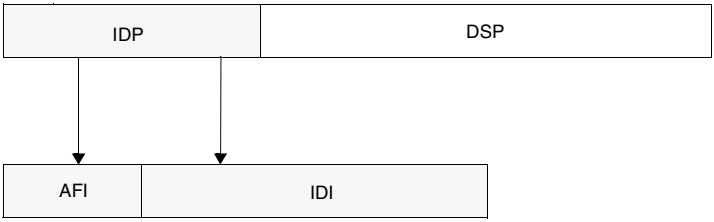
You can enter the LU number as an alternative to the LU name. Enter the LU number (Locaddr) specified in the TRANSIT configuration for the TRANSIT port of the TS application.

Representation format:

Enter a decimal number between 1 and 255.

OSI-NSAP

The structure of the OSI network addresses to be specified here is described in ISO8348/Add.2. It is shown below in abbreviated form:



IDP (Initial Domain Part)

This consists of two parts: the AFI and the IDI. *AFI (Authority and Format Identifier)*

The AFI identifier determines the structure and length of the IDP.

The following are defined:

– the IDI format
– the institution that defines the IDI values
– the gap digits used in coding the IDP
– the abstract syntax of the Domain Specific Part (DSP)

IDI (Initial Domain Identifier)

The IDI describes the addressing area and the entity for allocating the DSP.

The following are defined:

– the addressing area from which the DSP values originate
– the institution responsible for allocating the DSP values in this area

DSP (Domain Specific Part)

The DSP specifies the address in detail. The meaning of the DSP is determined by the institution identified by the IDI. The abstract syntax is determined by the AFI.

The table below specifies the minimum/maximum number of digits for the IDP (i.e. 2-digit AFI and its IDI) and the DSP. Please note that, in principle, only even DSP digits are permitted for the binary DSP syntax (even if the maximum value is not reached).

IDI format	AFI	IDP min.	IDP max.	DSP syntax	DSP max.
X.121	36	3	16	dec	24 decimal digits
X.121	37	3	16	bin	12 x 2 hex. digits
X.121	52	3	16	dec	24 decimal digits
X.121	53	3	16	bin	12 x 2 hex. digits
ISO_DCC	38	5	5	dec	35 decimal digits
ISO_DCC	39	5	5	bin	17 x 2 hex. digits
F.69	40	3	10	dec	30 decimal digits
F.69	41	3	10	bin	15 x 2 hex. digits
F.69	54	3	10	dec	30 decimal digits
F.69	55	3	10	bin	15 x 2 hex. digits
E.163	42	3	14	dec	26 decimal digits
E.163	43	3	14	bin	13 x 2 hex. digits
E.163	56	3	14	dec	26 decimal digits
E.163	57	3	14	bin	13 x 2 hex. digits
E.164	44	3	17	dec	23 decimal digits
E.164	45	3	17	bin	11 x 2 hex. digits
E.164	58	3	17	dec	23 decimal digits
E.164	59	3	17	bin	11 x 2 hex. digits
ISO_ICD	46	6	6	dec	34 decimal digits
ISO_ICD	47	6	6	bin	17 x 2 hex. digits
Local	48	2	2	dec	38 decimal digits
Local	49	2	2	bin	19 x 2 hex. digits
Local	50	2	2	bin	19 x 2 hex. digits
Local	51	2	2	bin	19 x 2 hex. digits

Table 10: Minimum/maximum number of digits for IDP and DSP.
bin = binary DSP syntax. dec = decimal DSP syntax

AFI and IDI are entered in direct succession. IDP and DSP are separated from each other by '+'.

Example:

49+1234569876 or 38123+556678

Port number

Specify the TCP port number for the TS application.

Representation format:

Specify a decimal number between 1 and 32767. In the case of LANINET, you must transfer the port number in the form of a string with format indicator A (ASCII).

Example:

A'4712'

Processor

Enter the processor number of the communication computer or host in which the remote TS application is resident.

Representation format:

For WANNEA, enter a decimal number between 0 and 255; for EMSNA enter a decimal number between 0 and 31. The processor number is specified together with the region number in the format: processor number/region number (e.g. 7/16).

Region

Enter the region number for the communication computer or host in which the remote TS application is resident.

Representation format:

Enter a decimal number between 0 and 255. The region number is specified together with the processor number in the format processor number/region number (e.g. 7/16).

Dial number

Enter the dial number under which you can reach the partner system.

Representation format:

Specify a maximum of 17 decimal digits.

SNPA information

The abbreviation SNPA stands for Subnetwork Point of Attachment and denotes the access point to a subnetwork. Enter the dial number, DTE address or PVC number via which you access the partner.

Representation format:

The SNPA information contains a connection number, prefixed by a subnetwork-specific keyword.

- for host-to-host connection in WAN or ISDN with ISO protocols of class 0/2 (address format WANSBKA):

  E.164 <ISDN_number>
  20-digit ISDN number

  E.163 <telephone_number>
  24-digit telephone number

  X.121 <IDI>
  17-digit X.25-DTE address

  PVC <PVC_number>
  X.25-PVC number(1 - 4095)

  X.31 <ISDN_number> X.121 <IDI>
  multi-stage with 20-digit ISDN number and 17-digit X.25-DTE address

  X.32 <dial_number> X.121 <IDI>
  multi-stage with 24-digit telephone number and 17-digit X.25-DTE address

- for heterogeneous host-to-host connection in X.25 network without transport protocol (address format WAN3SBKA):

  E.164 <ISDN_number>
  20-digit ISDN number

  X.121 <IDI>
  17-digit X.25-DTE address

  PVC <PVC_number>
  X.25-PVC number (1 - 4095)

  X.31 <ISDN_number> X.121 <IDI>
  multi-stage with 20-digit ISDN number and 17-digit X.25-DTE address

X.32 <dial_number> X.121 <IDI>
multi-stage with 24-digit telephone number and 17-digit X.25-DTE address

Example:

```
E.163 08963641625
X.121 123456
PVC 123
```

Station name

Enter the station name (T-selector) from the NEA address. The TS application uses the station name to attach to the transport system in the end system on which it resides.

Representation format:

The station name must be transferred in the form of a string preceded by a format indicator. Format indicators T, A, E and X are allowed. For the T-selector of the address format SDLCSBKA, however, only the format indicator T is permitted. The format indicator specifies the format to be given in the string (see section “Configuring with fssadm” on page 103).

In all formats, the station name must not exceed 8 characters. With format indicator X, this corresponds to 16 hexadecimal characters.

In TRANSDATA format, shorter entries are padded with blanks, otherwise with NIL. Longer entries are rejected as errors.

Sym-dest-name

Enter the symbolic destination name from the TRANSIT configuration. The sym-dest-name designates the LU6.2 program (LU = logical unit) for TRANSIT-LU6.2 on the partner LU.

Representation format:

The sym-dest-name must be transferred as a string of exactly 8 characters. The string may only contain uppercase letters [A-Z] and digits [0-9].

TPI (optional)

Enter the Transport Protocol Identification (TPI) for the TRANSPORT ADDRESS with address format WANSBKA if the TPI is expected for connection setup with the remote TS application.

CCP-WAN does not interpret the TPI on T-CONNECT.request, but in this case extracts the TRANSPORT ADDRESS without checking it and enters it in the “call user data” field of the call request packet.

Representation format:

You must transfer the TPI in the form of a string with format indicator X (hexadecimal). The value must comprise an even number of hexadecimal digits. You may not specify more than 32 hexadecimal digits.

TPC (optional)

By entering the transport protocol class (TPC), you can control the negotiation of the transport protocol class in compliance with ISO8073 on *T-CONNECT.request*. If you do not enter a TPC, the default value set by the CCP configuration applies (2/2).

Representation format:

You can specify *2/0, 2/2, 0/0* or *0/-* for TPC. These values have the following meanings:

– 2/0 preferably class 2, alternatively class 0
– 2/2 preferably class 2, alternatively class 2
– 0/0 preferably class 0, alternatively class 0
– 0/- only class 0, no alternative

T-selector

In the end system in which it resides, the TS application uses the T-selector to attach to the transport system.

Representation format:

The T-selector must be specified as a ‘T-selector’ string with the preceding format indicator T, A, E, X or V. The format indicator sets the format of the string or its coding. In hexadecimal format (format indicator X) an even number of digits must be specified, and not more than 20 (max. 64 in the TRANSPORT ADDRESS). In ASCII format (A) and EBCDIC format (E), you may specify no more than 10 characters (32 in the TRANSPORT ADDRESS). In TRANSDATA format (T) you may specify no more than 8 characters.

If you specify format indicator V, the entry for the T-selector is ignored and a blank entry for the T-selector is incorporated in the TS directory.

Format indicators

The various format indicators have the following meaning:

T (TRANSDATA format)
: The T-selector is specified in TRANSDATA format for station names, i.e. the string must only contain uppercase letters, digits, and the special characters '$', '#' and '@', must not be more than 8 characters long, and must not begin with a digit. The T-selector is then stored internally in EBCDIC.DF.03 (international/German DF version 03) and is padded to 8 positions with blanks.

A (ASCII character format)
: Each character entered is stored in ISO 7-Bit code. The character string must be a maximum of 10 characters (32 in the TRANSPORT ADRESS) or 8 characters long, depending on the choice of transport system.

E (EBCDIC character format)
: Each character entered is stored in EBCDIC code EBCDIC.DF.03 (international/German DF version 03). The character string must be a maximum of 10 characters (32 in the TRANSPORT ADRESS) or 8 characters long, depending on the choice of transport system.

X (hexadecimal format)
: The T-selector is transferred as a hexadecimal string. The string must contain an even number of hexadecimal digits [0-9,A-F,a-f]. Each pair of digits is stored as one byte (character), where the 1st digit represents the value of the high-order bit and the 2nd digit represents the value of the lower-order bit. For example, X'3a' corresponds to the bit representation '0011 1010' (highest-order bit on the far left).

V (blank format)
: You can use this format indicator to generate dummy entries. In this case the T-selector exists but has no value.

  You generate a null entry by specifying V''. If you enter a non-blank string after V, it is ignored.

WAN CC/line identifier

CCs and lines that can be used for the connection.

Representation format:

List of CC numbers (separated by blanks). If desired, a list of line numbers, separated by commas, can be specified for each CC number, separated each time by a colon. A line number identifies a line connection on the CC.

The line numbers 0, 1, 2, 3, 4, 32, 33 and 34 are permissible, as are CC numbers ranging from 1 through 255. The configuration of your system will determine which combinations are appropriate. Please also read the instructions in the manuals “CMX/CCP, WAN Communication” [4] and “CMX/CCP, ISDN Communication” [3] and the Release Notices. This list is prefixed by the keyword WAN.

Example:

WAN 1:1,2 2:33

7.2.8 Input rules for TNS files

7.2.8.1 Characters with a special meaning

Apart from the blank, the following characters also have a special meaning:

$ The dollar symbol introduces an INCLUDE, ORIGIN, or VERSION statement.

$ must be escaped if $INCLUDE, $ORIGIN, or $VERSION is defined as the GLOBAL NAME.

; A semicolon introduces a comment. The rest of the line is then ignored.

() Parentheses can be used to group together fields which are spread over more than one line as a single input record. In particular, this allows comments regarding a field to be included anywhere within the input record (a comment generally indicates the end of a line).

The following example is *a single* input record used to specify the TRANSPORT ADDRESS of the TS application X.25.

```
X\.25 (  TA WANSBKA
X.121 45890040033    ; DTE address
A'dtxp-33-01'        ; T-selector
2/0                  ; Transport Protocol Class (TPC)
)
```

\ The backslash is used to escape the special meaning of the character that follows it. If the character following the \ has no special meaning anyway, the \ is ignored.

. A period is used to separate name parts in the specification of the GLOBAL NAME.

' Single quote; in strings enclosed in single quotes, the characters $; ( ) . * @ and the blank lose their special meaning. They then have their face value.

Strings for T-, S- and P-selectors must always be enclosed in single quotes.

* The asterisk is reserved for future uses. It is not permissible to use * as the only character of a name part in the GLOBAL NAME. In this particular case, it is not possible for * to be escaped.

@ The “commercial at” symbol is reserved for future uses.

The special meaning of a character is escaped by a preceding \ (backslash) or by being enclosed in single quotes. The line separator is ignored if preceded by a \ or if escaped by means of () (parentheses).

7.2.8.2 Names with the same high-order name parts

If you want to define entries for more than one TS application associated with leaves on a single branch of the naming tree, you do not need to repeat the common name parts of the GLOBAL NAMEs in the *name* specification every time. You can specify the common name parts as the “origin”. A . (period) and the value of origin are then added to all names that have been specified relatively in the *name* fields. This means that you only have to specify those name parts not included in the origin in the *name* fields. The relative value specified for *name* together with the value of origin must form a syntactically correct GLOBAL NAME.

A GLOBAL NAME in the *name* field is relative to an origin whenever it does *not* end in a . (period). If it does end in a . (period), then it is absolute (relative to ROOT). A GLOBAL NAME specified absolutely is not extended even if you specify an origin. If no origin is specified then any specification for a GLOBAL NAME is absolute (relative to ROOT).

Example

```
Field contents       Value                  Resulting
of name          of origin              GLOBAL NAME
-------------------------------------------------------
myappl            myhost.sttz.Mch-P.D
myappl.myhost.sttz.Mch-P.D
np5.np4           .np2                      np5.np4..np2
```

You can specify the value of *origin* as follows by means on a control line in the input file:

$ORIGIN␣origin

For *origin*, you enter the origin which is to be added to all relative name specifications. *$ORIGIN␣origin* must be the only content of the record. If no value is specified for *origin*, the GLOBAL NAMES of the subsequent input records are not extended. A $ORIGIN statement changes the value defined for *origin* when a command is entered in *tnsxfrm* format.

The origin definition using $ORIGIN applies only up to the next $ORIGIN statement or to the end of this file.

7.2.8.3 Nesting input files

You can divide your inputs for the TNS over several files, for example to separate the TNS entries by product.

The files can be nested with $INCLUDE statements. An $INCLUDE statement in an input file is replaced by the contents of the specified file.

An *$INCLUDE* statement is a record which consists solely of:

$INCLUDE␣file

For *file*, you must specify the name of the file to be inserted. *file* must consist of entries in *tnsxfrm* format. It is permissible for *file* to contain further $INCLUDE statements. These may not, however, trigger any direct or indirect recursion. A maximum of ten $INCLUDE statements may be nested.

The value defined for the origin ($ORIGIN statement) is inherited by the subordinate INCLUDE level. When you return to the higher-ranking INCLUDE level, the original value for the origin of this level is restored.

7.2.8.4 Specifying the version for format and syntax

A $VERSION statement is a record containing only the following:

$VERSION␣version

The version number 5.1 must be specified for *version*.

7.2.8.5 Migration

This section is only of interest if you want to migrate your TNS configuration from a Reliant UNIX system with earlier CMX versions to the Solaris system.

The syntax of the *tnsxfrm* format has changed from CMX V3.0 to V4.0 in some address formats; however, there were no incompatible changes from CMX 4.0 to 5.x. TNS offers migration to the format of CMX V5.x for files that were created under CMX V3.0. This migration is initiated automatically. TNS identifies the version of the file format using the $VERSION record (see section “Specifying the version for format and syntax” on page 98). Below is a comparison of the relevant address formats in CMX V3.0 and CMX V5.x syntax. No migration is necessary from CMX V4.0 to CMX V5.x.

If you are working with RFC1006, refer also to the information in chapter “Configuring connections via RFC1006” on page 207.

```
3.0: TA ISDNSBKA idi tsel [tpi] [tpc]
5.0: TA WANSBKA E.164 idi tsel [tpi] [tpc]

3.0: TA WANSBKA idi tsel [tpi] [tpc]
5.0: TA WANSBKA X.121 idi tsel [tpi] [tpc]

3.0: TA WAN3SBKA idi
5.0: TA WAN3SBKA X.121 idi

3.0: TA ISDNSBKA LNR line tsel [tpi] [tpc] CC Wijk
5.0: TA WANSBKA tsel [tpi] [tpc] WAN i:line j:line k:line

3.0: TA WANSBKA PVC pvcNumber LNR line tsel [tpi] [tpc] CC Wijk
5.0: TA WANSBKA PVC pvcNumber tsel [tpi] [tpc] WAN i:line j:line
k:line

3.0: TA WANSBKA LNR line tsel [tpi] [tpc] CC Wijk
5.0: TA WANSBKA tsel [tpi] [tpc] WAN i:line j:line k:line

3.0: TA WAN3SBKA PVC pvcNumber LNR line CC Wijk
5.0: TA WAN3SBKA PVC pvcNumber WAN i:line j:line k:line
```

```
3.0: (MSA ISDNSBKA idi1
     TA WANSBKA idi2 tsel)
5.0: TA WANSBKA X.31 idi1 X.121 idi2 tsel
```

7.2.9 TS directory

Using the *tnsxcom* command, you can transfer files in *tnsxfrm* format to TS directories. You can set various modes for functions such as syntax checking, updating, or creating new TS directories. The command has the following syntax (abbreviated form, further details and options on page 307):

tnsxcom [␣**-d**␣num] [␣modus] [␣file ...]

The options have the following meaning:

-d␣num

Number of the TS directory to be processed. You can enter the numbers 1 through 9. If no value is specified, 1 is set (corresponds to DIR1).

modus

The following options are possible for *modus*:

-l LOAD mode

tnsxcom takes the entries individually from the *file* file and fills the TS directory (previously empty) with the syntactically correct entries.

-s CHECK mode

tnsxcom applies only the syntax check to the *file* file and logs any syntax errors found. The TS directory is not updated.

-S CHECK_UPD mode

As with the *-s* option, the syntax check is made on the entire *file* file in a first run. If no syntax errors are found in *file*, *tnsxcom* then updates the TS directory in a second run.

-u UPDATE mode

tnsxcom takes the entries individually from the editable *file* file and merges the syntactically correct entries into the TS directory by defining entries which were not previously present or updating existing entries. (Option *-u* is the default value for *option*.)

-i INTERACTIVE mode

tnsxcom reads the entries in *tnsxfrm* format from stdin after it has indicated that it is ready for input by outputting a prompt character, and merges the entries in the TS directory. Entries that did not previously exist in the TS directory are inserted; entries that already exist are updated.

file ...

Name of the file with entries in *tnsxfrm* format which, if *option* = *-l*, *-s*, *-S* or *-u*, is to be evaluated by *tnsxcom*. More than one file can be specified.

If *option* = *d*, specify the name of the file in which *tnsxcom* is to edit the contents of the TS directory.

Example

The following call transfers the entries from the *input.dir* file into the previously empty TS directory 2:

```
tnsxcom -d 2 -l input.dir
```

7.2.9.1 Deleting an entry for a TS application from the TS directory

If you want to delete the entire entry of a TS application from the TS directory, transfer the following record to tnsxcom.

```
[name␣]DEL
```

TNSXCOM then deletes the GLOBAL NAME and all properties assigned to the TS application *name* from the TS directory.

The TS application is then no longer known to the TNS. See also the *tnsxdel* command in section “Deleting TNS entries (tnsxdel)” on page 311.

7.2.9.2 Displaying the properties of a TS application

If you make your entries for tnsxcom in interactive mode, the TS application properties currently entered in the TS directory can be displayed on the screen. In this case, transfer a record in the following format:

```
[name␣]DISP
```

In this way, the previously entered or modified entries of a TS application can be displayed for checking.

If you specify a record with the above format in a file which you then transfer to tnsxcom, a warning is output during compiling and the entry is ignored by tnsxcom.

7.2.9.3 Specifying the TS directory

Using an input record in the file, you can switch to a different TS directory. The record must have the following format:

```
DIR␣n
```

For *n*, specify the number of the TS directory you want to switch to.

The subsequent records then refer to this TS directory, which is processed until you explicitly switch to another TS directory or until input is terminated.

7.2.9.4 Example of tnsxcom entries

The following sample file is intended to clarify the syntax of *tnsxcom*:

```
; RFC1006  transport address of an application accessible via
; the IP address 10.25.1.27 with T-selector in TRANSDATA format
;
; name   type data
rfcanw01 TA   RFC1006 10.25.1.27 PORT 102 T'RFCANW01'
;
; The NEA partner $DIALOG in processor 1/18,
;
;name   type data
wan1anw TA   WANNEA T'$DIALOG' 1/18
;
; Two applications that communicate with each other
; by means of process-to-process communication
;
```

```
;name   type data
ipclok  TA   LOOPSBKA A'IPC-LOK'
        TSEL LOOPSBKA A'IPC-LOK'
ipcrem  TA   LOOPSBKA A'IPC-REM'
        TSEL LOOPSBKA A'IPC-REM'
;
;Transport address of a WAN partner accessible via OSI
;transport protocol and via DTE address 123456.
;
;name   type data
wananw01 TA   WANSBKA X.121 123456 A'ANW01'
```

7.2.9.5 Special cases for TNS entries

Multi stage access (X.25 access via ISDN or telephone network)

The TNS entry contains the X.25 address and the (ISDN or telephone) dial number for the outgoing call. With incoming calls, often only the X.25 address is transmitted. A second (dummy) entry must therefore be created for applications that identify the calling partner via a TNS call.

Example

Entry for outgoing call with telephone number:

```
tel_out TA WANSBKA X.32 23456 X.121 65432 A´remote_appl‘
```

Entry for outgoing call with ISDN number:

```
isdn_out TA WANSBKA X.31 23456 X.121 65432 A´remote_appl‘
```

Entry for incoming calls:

```
tel/isdn_in TA WANSBKA X.121 65432 A´remote_appl‘
```

7.3 Configuring with fssadm

This section describes the configuration of the Forwarding Support Service (FSS) using the command line interface (CLI). To configure FSS objects, you can also use the character-oriented menu CMXCUI (see section “Overview of the character-oriented user interface CMXCUI” on page 59).

fssadm command mode is an expert mode and should only be used with the appropriate knowledge. To aid comprehension, read the information on the FSS addressing concept in section “Addressing partner systems in the FSS” on page 45. Details on the configuration procedure can be found in section “Configuration procedure” on page 69.

In an FSS configuration, data is stored in the form of objects (e.g. routes, network addresses, operating parameters, see section “Addressing partner systems in the FSS” on page 45), to which particular attributes are assigned.

The description of a configuration is stored as an FSB configuration in the database of the FSS, the Forwarding Support Information Base (FSB).

The actual entries in the database are generated by creating or modifying the objects of the prescribed object classes. A range of attributes is assigned to each object class. When creating or modifying objects, these attributes must be assigned the current values.

In addition to creating objects with the *fssadm* command, you can create a configuration as a file in *fsconfig* format (see section “Creating FSS configuration file (fsconfig format)” on page 126). An existing configuration can be modified using the *fssadm* command (see section “Configuration procedure” on page 69).

Actions

You can perform various actions using the *fssadm* command:

– With *create* you create an object.

Example:

```
fssadm create GNSAP name=NEA_REG12 nea-addr-pattern=*/12 \
   snpa-list=route1
```

– With *get* you display an object. If you specify attribute values, *fssadm* only displays objects with these attribute values.

Example:

```
fssadm get NSAP nea-addr=1/18
```

– With *set* you modify a configured object. In this case, the object is either unique in itself or is uniquely defined by the specified attribute values and the command line syntax:

Example:

```
fssadm set NSAP name=BS2000-2 nea-addr=2/14 net=NEA \
   snpa-list=route2
```

– With *delete* you delete an object. It must be identifiable by the specified attributes.

Example:

```
fssadm delete NSAP name=NEAHOST1
```

Specification of several attributes, one of which already uniquely identifies an object, will be rejected by *fssadm*.

– With *check* you check the validity of the FSB configuration or FSS configuration file.

Object classes

The table below indicates which actions in the *fssadm* command can be applied to a particular object class and which attributes are assigned to the object class.

Object class	Actions	Attributes
FSBGEN	create | delete | set | get | check	gen-nr, path, id, version, date-time, print, use
config-file	create | check	gen-nr, path,
LOCNSAP	set | get	gen-nr, name, nea-addr, osi-addr, internet-addr, tui-name
NSAP	create | delete | set | get	gen-nr, name, nea-addr, osi-addr, internet-addr, net, r6-impl, r6-tpdusize, r6-drtpdu, r6-aktpdu, access, hop-nsap, snpa-list, type, subnet
GNSAP	create | delete | set | get	gen-nr, name, nea-addr-pattern, net, access, snpa-list, type, subnet
FACIL	create | delete | set | get	gen-nr, name, short-id, facil, admit, npid, compress, ppp-accm, ppp-profile, ppp-auth-params, ppp-auth-protocol, t70-profile, isdn-cug, isdn-throughput, isdn-ra, isdn-partner-prot, x25-octet-string, x25-throughput, x25-window-size, x25-packet-size, x25-cug, x25-cug-oa, x25-bcug, x25-revch, x25-transit-delay, x25-rpoa-selection, x25-fast-select, x25-nui, x31min-svc-to-Bchan, x25-description, fr-prio, fr-cir, fr-cbs, fr-ebs, fr-encaps, fr-max-transit-delay, in-max-idle, out-max-idle
SUBNET	create | delete | set | get	subnet, incoming-call, facil, osi-nsap-address
SNPAROUTES	create | delete | set | get	gen-nr, name, short-id, type, facil, subnet, nea-tunnel, mac-addr, dte-addr, pvc-nr, dial-nr, line-nr, isdn-nr, nailed-up-isdn, phone-nr, x31-dte-addr, x32-phone-nr, x31-pvc-nr, x31-msa, fr-pvc

Table 11: Actions, object classes, and attributes

Object class	Actions	Attributes
PPPAUTH	create | delete | set | get	gen-nr, name, short-id, loc-id, peer-id, pap-loc-pwd, pap-peer-pwd, chap-loc-secret, chap-peer-secret
logging-params	set | get	sent-records, got-records, fssd-kbytes, fssadm-kbytes
statistics	set | get	date-time, seconds, searches, hits, compares, stores
trace	set | get	level

Table 11: Actions, object classes, and attributes

Attributes

A distinction is made between mandatory and optional attributes. In total, the following groups of attributes can be formed:

1. Mandatory specification with *fssadm create* and when editing an *fsconfig* file.
2. Optional specification with *fssadm create* and when editing an *fsconfig* file.
3. Optional specification in *fsconfig* file.
4. Optional specification with *fssadm create*.
5. Attribute within a group, from which at least one must be specified.
6. Attribute within a group of attributes which are mutually exclusive.
7. Attribute within a group of attributes, some of which are mutually exclusive.
8. Redundant attribute which, although accepted by *fssadm create* and *fsconfig*, can be omitted because it can be derived from other attributes. Such an attribute may be useful as a filter in the *get* command.
9. Attribute that can only be specified as a filter in the *get* command.
10. Attribute that is displayed in the output for a *get* command but cannot be entered.
11. Attribute that is automatically specified in object generation and can only be entered with the commands *delete, set* and *get*.

In the following tables from page 108, the attributes are identified with the appropriate numbers.

Help on fssadm syntax

The help function described below only refers to the syntax of *fssadm*. It is possible that the syntax offered by the help function will be rejected following the semantic check. It may also happen that syntactically correct values or attribute combinations do not make any sense, e.g. because an evaluating function is not installed or not released.

You can obtain information on the syntax of the *fssadm* command with the following command input:

– *fssadm ?* outputs a general description of the syntax of *fssadm* and information on the help function.

– *fssadm action ?* outputs the object class to which an action can be applied.

– *fssadm action objectclass [[attributename=] attributevalue ...]?* completes the command with the attributes suitable to the specified context. In this case, the restriction is that the context is only considered for the attributes that follow the context in the output.

  Example: The input *fssadm create snparoutes type=isdn-nc ?* returns:

```
fssadm create SNPAROUTES <name> [<subnet>] type=ISDN-NC \
{<remsnpa> <nailed-up-isdn> } (min=0,max=1) [<facil>]
```

– *fssadm action objectclass [[attributename=] attributevalue ...] attributename= ?* outputs the syntax of the specified attribute in the specified context. The context is only considered for attributes that precede the queried attribute. Example: The input *fssadm create snparoutes subnet=isdn-1 type=?* returns:

```
<type>: ISDN | ISDN-[NC] | X31-M[SA] | X31-S[VC] | X31-P[VC]
```

If you enter the question mark (?), note the special meaning for the shell. The character may have to be escaped with a backslash (\).

7.3.1 Overview of object classes and their attributes

The following tables contain overviews of the configurable parameters for the various object classes, and describe their meaning. The object classes are arranged into four categories:

- Superior object classes that relate to the FSB configuration or FSS configuration file. This group includes: config-file, FSBGEN
- Object classes that are relevant to each configuration. These include: FACIL, SNPAROUTES, LOCNSAP, NSAP, GNSAP, SUBNET
- Object classes that are only significant for the product CS-ROUTE. These include: PPPAUTH
- Object classes that are used for maintenance and diagnostic purposes. These include: logging-params, statistics, trace

Object class config-file

An object of class *config-file* denotes an FSS configuration file. For an explanation, turn to section “Creating FSS configuration file (fsconfig format)” on page 126.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
gen-nr 2)	Decimal no. between 1 and 9999	Number of the FSB configuration from which a configuration file is to be created.
path 1)	Max. 63 characters	Path name of the configuration file. Mandatory attribute with *check*.

Table 12: Actions and attributes of the object class config-file

Object class FSBGEN: FSB configuration

Objects of the class *FSBGEN* denote an FSB configuration that was created from an FSS configuration file.

Attribute	Format	Meaning
id	Character string (max. 64 characters) *	Identification text. Output with *get*. Input only possible in *fsconfig* format.
gen-nr 4)	Decimal no. between 1 and 9999 or *NEXT-GEN-NR*	Number of the FSB configuration. Mandatory attribute with *set* and *check*. Simultaneous specification of *gen-nr* and *use* is not permitted with *get*. **
path	Max. 63 characters	Path name of the configuration file (only with *fssadm create*).
replace	YES | NO	Only with *create*: specifies whether an existing FSB configuration should be replaced by the new FSB configuration or not; also the active FSB configuration can be replaced. Optional attribute, default setting: NO
use	ACTIVE | NEXT-ACTIVE	Mandatory with *set*; optional with *get*. Simultaneous specification of *gen-nr* and *use* is not permitted with *get*. ***
version 10)	6-digit hexadecimal character string	Version of the FSB configuration (output with *get*).
date-time 10)	Month day hh:mm:ss year	Date and time of creation (output with *get*).
print	MINIMUM | VERBOSE	Scope of output of the command *fssadm check FSBGEN* (optional attribute).

Table 13: Attributes of the object class FSBGEN

* If the text is to contain blanks, the character string must be enclosed in double quotes (“).

** NEXT-GEN-NR means “next number not yet assigned”. This value is only permitted with *create* and is the default value.

*** use=ACTIVE means: the FSS is in active state and will use this FSS configuration for as long as this state continues.

use=NEXT-ACTIVE means: the FSS will use this configuration the next time it is activated.

Object class FACIL: Facilities

You can assign certain facilities (charge reversal, throughput rate) to each route (object class SNPAROUTES). These facilities are defined in a FACIL (facilities) object.

The attributes isdn-* and x25-* cannot be combined with the fr-* attributes.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
gen-nr 9)	Decimal no. between 1 and 9999	Number of the FSB configuration.
name 1)	1-15 characters: letters, digits, the special characters ´_´ and ´#´. A distinction is made between uppercase and lowercase letters. The 1st character must not be a digit or an underscore (_).	Name of the FACIL object.
short-id 11)	Decimal no. between 1 and 9999	Implicitly assigned short identifier.
facil 2)	See *name*	Name of another FACIL object.
admit 2)	BOTH_IN_AND_OUT | OUTGOING ONLY | INCOMING_ONLY | NEITHER_IN_NOR_OUT	Access control on subnetwork level.

Table 14: Attributes of the object class FACIL

Attribute	Format	Meaning
npid 2)	OSI-CONS | INTERNET | NEA | SNA/FR | FAX2/3 | PRIVATE.	Network protocol ID. The attribute is ineffective if ppp-profile=STANDARD is set.
compress 2)	TCP/IP | NO	Van-Jacobsen header compression
ppp-accm 2)	ALL_CNTRL_CHARS | NO_MAPPING | *	PPP with asynchronous procedure via ISDN
ppp-profile 2)	STANDARD | NO | GSM	Usage of point-to-point protocol.
ppp-auth-params 2)	see *name*	Name of a PPPAUTH object.
ppp-auth-protocol 2)	NO | PAP | CHAP	Authentification protocol
t70-profile 2)	YES | NO	Usage of protocol variant T.70 of CCP-WAN-CONS profile.
isdn-cug 2), 7)	Decimal no. between 0 and 65535	Closed user group.
isdn-throughput 2), 7)	9.6 | 64 | 128	Throughput
isdn-ra 2), 7)	X30/V110-SYN | V110-ASYN	Adaptation of transmission rate.
isdn-partner-prot 2), 7)	1TR6/TYPE1 | 1TR6/TYPE1A | SIMPLE	Adaptation of ISDN signalling.
fr-encaps 2), 7)	YES | NO	Protocol encapsulation in accordance with RFC1490.
fr-cir 2), 7)	0 to 2048 Kbit per second	Committed information rate.

Table 14: Attributes of the object class FACIL

Attribute	Format	Meaning
fr-cbs 2), 7)	0 to 2048 KBit	Committed burst size.
fr-ebs 2), 7)	0 to 2048 KBit	Exceeded burst size.
fr-prio 2), 7)	1 | 2 | 3 (1 = highest priority)	Priority
fr-max-transit-delay 2), 7)	1-65535 tenths of a second	Maximum transmission duration.
x25-octet-string 2), 7)	1-109 octets in hex format	DTE facilities in accordance with CCITT X.25 Annex G (IS8208).
x25-packet-size 2), 7)	Send direction[/receive direction] with the values for S/R: 16 | 32 | 64 |128 | 256 | 512 |1024 | 2048. If R not specified, R=S.	Packet size.
x25-window-size 2), 7)	Send direction[/receive direction] with the values for S/R: 1-127	Window size.
x25-throughput 2), 7)	Send direction[/receive direction] with the values for S/R in Kbit/s: 2,4 | 4,8 | 9,6 | 19,2 | 48 | 64	Throughput class.
x25-cug 2), 7)	0-9999. Leading zeros are analyzed: 1-2-digit input means ‘basic format’, 3-4-digit input means ‘extended format’.	Selection of a closed user group.
x25-cug-oa 2), 7)	0-9999. See x25-cug.	Selection of a closed user group with unrestricted outgoing call.
x25-bcug 2), 7)	0-9999. Leading zeros are not evaluated; “extended” format is always used	Selection of a bilaterally closed user group.

Table 14: Attributes of the object class FACIL

Attribute	Format	Meaning
x25-revch 2), 7)	BOTH_REQ_AND_ACC | REQUEST_ONLY | ACCEPT_ONLY | NEITHER_REQ_NOR_ACC	Request reverse charges or accept request for reversed charges.
x25-transit-delay 2), 7)	0-65534 milliseconds	Desired transmission time.
x25-fast-select 2), 7)	NO-RESTRICTION | RESTRICTION	Fast select (short dialog using the Call User Data field).
x25-rpoa 2), 7)	DNIC[+DNIC...] with up to 12 elements	Selection of a route via one (or more) private network operators identified by their DNIC (Data Network Identification Code).
x25-nui 2), 7)	Max. 16 printable characters (ASCII, EBCDIC) or max. 16 hexadecimal digit pairs: Format: *formind:nui-value* formind = A | E | X	Network User Identification.
x31min-svc-to-Bchan	n-TO-EACH | MAX-TO-EACH | MAX-TO-ONLY-ONE n-TO-EACH: n={1...127}	Seizure by SVC of the B-channels to an ISDN partner. Only for X.25-minimum integration or DTE-DTE links.
x25-description 2), 7)	Name of an XZSTW macro (TRANSDATA conventions)	Selection of a predefined description of the X.25 access.

Table 14: Attributes of the object class FACIL

* Using the parameter *ppp-accm* you specify the control characters to be transmitted transparently via mobile telephone network at connection setup. Apart from the two values ALL_CNTRL_CHARS and NO_MAPPING you can specify the control characters explicitly in abbreviated form or as hexadecimal string. You get the exact syntax using the help command *fssadm create facil ppp-accm=?*

The following table shows a list of the control characters:

Control character	**Hex**	**Meaning**
NUL	00	No operation
SOH	01	Start of Heading
STX	02	Start of Text
ETX	03	End of Text
EOT	04	End of Transmission
ENQ	05	Enquiry
ACK	06	Acknowledge
BEL	07	Bell
BS	08	Backspace
HT	09	Horizontal Tabulation
LF	0A	Line Feed
VT	0B	Vertical Tabulation
FF	0C	Form Feed
CR	0D	Carriage Return
SO	0E	Shift Out
SI	0F	Shift In
DLE	10	Data Link Escape
DC1	11	Device Control 1 (XON)
DC2	12	Device Control 2
DC3	13	Device Control 3 (XOFF)
DC4	14	Device Control 4
NAK	15	Negative Acknowledgement
SYN	16	Synchronous Idle

Table 15: Control characters for asynchronous PPP

Control character	Hex	Meaning
ETB	17	End of Transmission Block
CAN	18	Cancel
EM	19	End of Medium
SUB	1A	Substitute Character
ESC	1B	Escape
FS	1C	File Separator
GS	1D	Group Separator
RS	1E	Record Separator
US	1F	Unit Separator
SP	20	Space

Table 15: Control characters for asynchronous PPP

Object class LOCNSAP: Local NSAP addresses

The address of the local system is defined with the object class LOCNSAP.

Attribute	Format	Meaning
gen-nr 9)	Decimal no. between 1 and 9999	Number of the FSB configuration.
name 1)	1-32 printable, visible characters	Name of the LOCNSAP object.
nea-addr 5)	*p/r* with decimal digits *p* and *r* (0 ... 255)	NEA address: processor/region number.
osi-addr 5)	In accordance with ISO8348 Ad 2. See “OSI-NSAP” on page 88.	OSI-NSAP address.

Table 16: Attributes of the object class LOCNSAP

Attribute	Format	Meaning
internet-addr 5)	canonical representation of an IPv4 or IPv6 address, see page 15	IP address that represents the local system within the CS-GATE functional framework of address representation. If no special address representation function is to be configured for a particular IP interface, enter 0.0.0.0 here.

Table 16: Attributes of the object class LOCNSAP

Object class NSAP: Remote NSAP or remote network entity

Each end system or intermediate system for which you want to establish transport connections is represented by an NSAP object.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
gen-nr 9)	Decimal no. between 1 and 9999	Number of the FSB configuration.
name 1)	1-32 printable, visible characters	Name of the NSAP object.
nea-addr 5)	*p/r* with *p* and *r* (0 ... 255)	NEA address: processor/region number.
osi-addr 5)	In accordance with ISO8348 Ad 2. See “OSI-NSAP” on page 88.	OSI-NSAP address.
internet-addr 5)	canonical representation of an IPv4 or IPv6 address	IPv4 or IPv6 address of a remote NSAP
net 1) or 8)	NEA | INTERNET | OSI-CONS	Network used by the local system to reach the NSAP.
access 8)	DIRECT | DYNAMIC | HOP | NSAP-ADDR	Access to the SNPA address via which the NSAP can be reached.

Table 17: Attributes of the object class NSAP

Attribute	Format	Meaning
snpa-list 6) and conditionally 1)	*snpa+snpa+..+snpa* with max. 20 list items. *snpa:* name | name/weight *name*: see SNAPROUTES attribute *name* *weight:* number from 1-20. *	List of alternative SNPAROUTES objects that can be used to reach this NSAP. The priority can be specified with a value for *weight* (20 is the highest priority).
hop-nsap 6) and conditionally 1)	See name	Name of the NSAP object whose SNPA address is to be used.

Table 17: Attributes of the object class NSAP

* To improve clarity, blanks and new-line control characters can appear before or after the “+” character. In this case, the entire expression must be enclosed in double quotes (“).

Example

```
snpa-list=“Route1 + Route2“
```

Additional filter attributes for *get NSAP*

The following table contains additional attributes that can only be used as filters with *get*.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
subnet 9)	X25-*n* | X21-*n* | PT-*n* | FR-1...128 | PP-*n* | ISDN-*n* *n*= 1, .., 32	Subnetwork ID
type 9)	X25 | PVC | X21 | PP | ISDN | X21DIRECT | X31-MSA | PT| X31-SVC | X31-PVC | X32-PTMSA | FR | ISDN-NC	SNPA address type

Table 18: Additional filter criteria for get NSAP

The only NSAP objects output are those to which routes are assigned by the specified SNPA address type or with the specified subnetwork ID. With the *snpa-list* attribute, only the routes that comply with the filter criteria are displayed. If additional routes are assigned which do not fulfill the filter criteria, these are displayed in summary by "+*".

Object class SUBNET: Local subnetwork interface

An object of the class SUBNET represents a local subnetwork interface that is identified uniquely by means of a subnetwork ID, or a group of similar local subnetwork interfaces that are identified by a subnetwork ID (subnet attribute) common to all these interfaces.

The object is assigned values that are required for setting X.25 minimum integration for calls from unknown ISDN partners or for X.32 dialing for telephone calls, as well as for activating and deactivating access control.

Attribute	Format	Meaning
subnet	X25-*n* | X21-*n* | PT-*n* | ISDN-*n* *n*= 1, .., 32	Subnetwork ID
incoming-call	NONE | RESTRICTED | ALL	Switch for activating/deactivating access control.
x25-description	Name of an XZSTW macro (TRANSDATA conventions)	Selection of a predefined description of the X.25 access.
facil	see *name*	Name of an additional FACIL object
osi-nsap-address	As defined in ISO 8348 Ad 2, see also page 86	OSI NSAP address

Table 19: Attributes of the object class SUBNET

Object class SNPAROUTES: Route

You use an SNPAROUTES object to configure a route within a subnetwork. This is defined by its starting point and its endpoint. The starting point of the route is a local subnetwork interface, while the endpoint is the subnetwork interface of the remote system. A number of interfaces can be combined locally into a group if they lead to the same subnetwork. The starting point of the route is then defined by a subnetwork ID under which the desired subnetwork interfaces are combined.

The various subnetwork addresses are assigned to the subnetwork IDs as follows:

SNPA address type	**Subnetwork ID**
X25	PVC	X25-x
X21	X21DIRECT	X21-x
PP (point-to-point)	PP-x
PT (public telephone)	X32-PTMSA	PT-x
ISDN	ISDN-NC	X31-PVC	X31-SVC	X31-MSA	ISDN-x
FR	FR-x

Table 20: Assignment of subnetwork ID to SNPA address type

With ISDN permanent connections, the subnetwork ID can be assigned both to an interface and to an individual channel (B or D channel).

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	**Format**	**Meaning**
gen-nr 9)	Decimal no. between 1 and 9999	Number of the FSB configuration.
name 1)	1-15 characters: letters, digits, the special characters ´_´ and ´#´. A distinction is made between uppercase and lowercase letters. The 1st character must not be a digit or an underscore (_).	Name of the SNPAROUTES object.
short-id 11)	Decimal no. between 1 and 9999	Implicitly assigned short identifier.
subnet 1)	X25-*n*	X21-*n*	PT-*n*	FR-1...128	PP-*n*	ISDN-*n* *n*= 1, .., 32	Subnetwork ID.

Table 21: Attributes of the object class SNPAROUTES

Attribute	Format	Meaning
type 8) or 1)	X25 | PVC | X21 | PP | ISDN | X21DIRECT | X31-MSA | PT| X31-SVC | X31-PVC | FR-PVC | X32-PTMSA | ISDN-NC	SNPA address type.
facil 2)	See attribute *name*	Name of a referenced FACIL object.
dte-addr 5) and 6)	1-17 decimal digits	Address of remoteX.25-DTE.
pvc-nr 5) and 6)	*pvc/dte* *pvc:*␣decimal number (0 ... 4095) *dte:*␣1-17 decimal digits	X.25-PVC number and associated local DTE address.
dial-nr 5) and 6)	*dial-nr* | DIRECT / *dial-nr* *dial-nr:* 1-24 decimal digits or 1-24 any visible characters enclosed in single quotes (´)	Remote X.21 dial number. In the case of “Direct Mode”: local X.21 dial number.
phone-nr 5) and 6)	1-24 decimal digits or 1-24 any visible characters enclosed in single quotes (´)	Telephone number.
line-nr 2) and 6)	[CC-no./] line-no. line-no.: 1 | 2 | 3 | 4 CC-no.: 1-256	Optional attribute: line number for dedicated line (KOGS parameter LPUFADR).
x31-dte-addr 5) and 6)	rem-dte-addr[/loc-dte-addr]	X.31 maximum integration: address of the remote X.25-DTE and optionally the address of the local X.25-DTE.
x31-msa 5) and 6)	isdn-no./dte-addr	multi stage access: remote ISDN dial number/remote X.25-DTE address.
x32-phone-nr 5) and 6)	phone-no./x25-dte-addr	X.32 via telephone network.

Table 21: Attributes of the object class SNPAROUTES

Attribute	Format	Meaning
fr-pvc 5) and 6)	CC-no./line/pvc CC-no.: 1-256 line: 0-255 pvc: 1-65535	Frame Relay PVC.
nailed-up-isdn 2) and 6)	CC-no./line number	Optional attribute: ISDN dedicated connection.
isdn-nr 5) and 6)	1-20 decimal digits	ISDN dial-up connection
x31-pvc-nr 5) and 6)	pvc-no.[/loc-dte-addr]	X.31 maximum integration: PVC number and optionally the address of the corresponding local X.25-DTE.

Table 21: Attributes of the object class SNPAROUTES

Object class GNSAP: Generalized NSAP

A GNSAP object represents a group of NEA systems whose NEA addresses match a certain pattern.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
gen-nr 9)	Decimal no. between 1 and 9999	Number of the FSB configuration.
name 1)	1-32 printable, visible characters	Name of the GNSAP object.
nea-addr-pattern 1)	*/*r with *r* (0 ... 255 | *)	NEA address: processor/region number.

Table 22: Attributes of the object class GNSAP

Attribute	Format	Meaning
snpa-list 1)	*snpa+snpa+..+snpa* with max. 20 list items. *snpa:* name | name/weight *name*: see attribute *name* for SNPAROUTES *weight:* digit from 1-20. See also NSAP.	List of routes that can be used to reach NEA systems represented by this GNSAP. Priority can be specified with a value for *weight* (20 is the highest priority).

Table 22: Attributes of the object class GNSAP

When working with the *get* command, the attributes “gen-nr”, “type”, and “subnet” are also available (as with object class NSAP).

Object class PPPAUTH: Local Identification for PPP

This object class is needed only for configuration of CS-ROUTE in the case of communication via TCP/IP using PPP (Point-to-Point protocol). A PPPAUTH object contains information about access control by PAP (Password Authentication Protocol) and CHAP (Challenged Handshake Authentication Protocol).

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
gen-nr 9)	Decimal no. between 1 and 9999	Number of the FSB configuration.
name 1)	1-15 characters: letters, digits, the special characters ´_´ and ´#´. A distinction is made between uppercase and lowercase letters. The 1st character must not be a digit or an underscore (_).	Name of the PPPAUTH object.
short-id 11)	Decimal no. between 1 and 9999	Implicitly assigned short identifier
loc-id 2)	1-32 printable visible characters.	Local identification.

Table 23: Attributes of the object class PPPAUTH

Attribute	Format	Meaning
peer-id 2)	1-32 printable visible characters.	Partner identification.
pap-loc-pwd 2)	Name of a file (max. 63 characters) containing the password in readable text (max. 32 characters).	PAP password for the local system.
pap-peer-pwd 2)	Name of a file (max. 63 characters) containing the password in readable text (max. 32 characters).	PAP password for the partner system.
chap-loc-secret 2)	Name of a file (max. 63 characters) containing the CHAP secret in readable text (max. 255 characters).	CHAP secret for the local system.
chap-peer-secret 2)	Name of a file (max. 63 characters) containing the CHAP secret in readable text (max. 255 characters).	CHAP secret for the partner system.

Table 23: Attributes of the object class PPPAUTH

Object class statistics

The object class *statistics* is used to output and reset the cache statistics of the FSS.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
date-time 10)	hh:mm:ss	Date and time the statistics start.
seconds 10)	Decimal no.	Period in which statistics were gathered.
searches 10)	Decimal no.	Number of search queries for an object in the cache.
hits 10)	Decimal no.	Number of hits.

Table 24: Attributes of the object class statistics

Attribute	Format	Meaning
compares 10)	Decimal no.	Number of comparisons with objects in the cache which were performed during a search.
stores 10)	Decimal no.	Specifies how often an object has been stored in the cache.

Table 24: Attributes of the object class statistics

Object class logging-params

The object class logging-params is used to output and modify parameters for logging the *fssadm* and the FSS daemon. Only the actions *set* and *get* are possible.

The FSS daemon logs alternately in the files *fssd_log.A* and *fssd_log.B*. *fssadm* logs alternately in the files *fssadm_log.A* and *fssadm_log.B*.

The number after the attribute name indicates information that can be found in the section “Attributes” on page 106.

Attribute	Format	Meaning
sent-records 2)	YES | NO	Logging of data records loaded in the cache by the FSS daemon.
got-records 2)	YES | NO	Logging of data records output by *fssadm* with the *get* command.
fssd-kbytes 2)	1...9999 Kbyte	Switching the log file of the FSS daemon after ... Kbytes.
fssadm-kbytes 2)	1...9999 Kbyte	Switching the log file for *fssadm* after ... Kbytes.

Table 25: Attributes of the object class logging-params

FSS log files

fssadm_log.A, fssadm_log.B

Contains the following specifications:

- date and time
- command entered
- error message, if the command was rejected or an error occurred
- if you set the attribute *got-records=YES* in the *logging-params* object, all data records that were output to *stdout* are also logged

The output of *fssadm_log* is determined in a *logging_params* object.

fsin_log

Contains logging entries for the commands *fssadm check config-file* and *fssadm create FSBGEN*. The file contains error messages and warnings with line specification, as well as information on automatically created objects.

fsin_acc

Contains all accepted entries of the FSS configuration file in bold standard format.

fssd_log.A, fssd_log.B

Contains specifications on the start and stop of the FSS, daemons, kernel memory requirement of FSB objects (see also *logging_params*). Data is written alternately in files A and B.

The files are located under */var/opt/SMAWcmx/adm/log*.

7.3.2 Creating FSS configuration file (fsconfig format)

A configuration file comprises statements with which the object class and attributes of an object are specified or created. In addition, $INCLUDE statements can be used. An $INCLUDE statement specifies that a further configuration file is to be included. The nesting of $INCLUDE statements is permissible up to a depth of 10.

Within or between statements, you can enter comments. A comment begins with ';' and ends with an end-of-file character or a newline character.

A field within a statement ends with one or more newline characters, blanks or tabs. Statements can therefore be arranged in columns.

A statement ends with a newline character. A \ at the end of a line renders a newline character ineffective. You can also enclose a statement in parentheses if it is very long or if comments are to be included and therefore newline characters must be made ineffective. If more fields than necessary are specified, this is rejected as an error.

The basic features of the syntax of object classes, attribute names and attribute values are defined as follows:

– The object class, the attribute name and the symbolic value of an attribute and $INCLUDE can be specified in upper or lowercase letters.

– Attributes can be specified explicitly with *attribute-name=value*.

fsconfig describes the syntax of statements in a configuration file of the Forwarding Support Service. The configuration file can be written using any editor and can be used to create an FSB configuration (see section "Configuration procedure" on page 69).

An entry in an FSS configuration file has the following basic format:

```
Object class [attribute-name=]attribute-value  ...
```

Object class identifies the defined object class, e.g. *LOCNSAP* and *NSAP*.

Attribute-name identifies the defined attributes of the object class. A complete description of the object classes and their attributes can be found in section "Configuring with fssadm" on page 104.

Within an FSS configuration file, relationships can only form links to preceding attributes. References to subsequent entries are not permitted. It therefore makes sense to adhere to the following sequence when creating the file:

– FSBGEN
– PPPAUTH
– FACIL
– SNPAROUTES
– LOCNSAP
– NSAP
– GNSAP
– SUBNET

Each FSS configuration file must contain a LOCNSAP entry (LOCNSAP is the only mandatory entry in the file).

If no entry is specified, a default entry is created for the following object classes:

– FSBGEN with the attribute *id*
– tune-nsaps

You can use $INCLUDE statements to incorporate other files into your configuration file:

```
$INCLUDE pathname/filename
```

The following section contains sample entries of the FSS configuration file.

Examples

NSAP address of the local system which communicates via TRANSDATA NEA:

```
LOCNSAP  ( name=D018S265 nea-addr=1/18 )
```

Definition of a facilities object with the facility “reverse charges with outgoing X.25 connections only”:

```
FACIL name=charging x25-revch=REQUEST_ONLY
```

Route to the remote system with DTE address 1930000 via the local subnetwork interface with identification X25-1; assignment of facility “reverse charges”:

```
SNPAROUTES name=x25_prv subnet=X25-1  dte-addr=1930000 \
      facil=charging
```

Route to the remote system via line number 4 via the local subnetwork interface with identification PP-11:

```
SNPAROUTES name=ddv subnet=PP-11  [ line-nr=4 ]
```

Description of an NEA partner with NEA address 28/5 accessed via route *x25_prv*, which was defined in the SNPAROUTES object:

```
NSAP  name=PGTR0039 nea-addr=28/5 net=NEA access=DIRECT \
    snpa-list=x25_prv
```

Description of an NEA partner with NEA address 19/5 accessed via route *ddv*, which was defined in the SNPAROUTES object:

```
NSAP  name=D018V019 nea-addr=19/5 \
      net=NEA access=DIRECT snpa-list=ddv
```

7.4 Sample configuration

To enable communication between the applications using the transport system, you must configure CMX and CCP. To do this, you must clarify the following issues and implement the appropriate measures:

– Which TS applications are to communicate with each other?

  Name the TS applications running on the local and remote systems. Make the necessary entries in the CMX component TNS.

– How do these TS applications reach each other?

  Define the resources (e.g. lines) to be used to establish the possible communication relationships. The necessary information can be obtained from the network operator. Enter this information in the configuration files of the subnetwork profiles.

  Enter the addresses of partner systems and routes to these systems in the CMX component FSS if intermediate systems are involved or if the connection is established via TRANSDATA NEA or TCP/IP via WAN. For other profiles, the address entries are generally only made in the TNS.

– Which facilities must be set for the desired connections?

  On your system, set the operating parameters of the subnetwork profiles such that they can communicate correctly with their partner entities in other systems. These specifications can also be obtained from the network operator. A description of how to configure subnetwork profiles can be found in the manuals “CMX/CCP, ISDN Communication” [3] and “CMX/CCP, WAN Communication” [4]). Define the partner-specific operating parameters in the FSS.

The following tasks must be performed when configuring CMX:

– configure applications

– configure routes (via NEA and TCP/IP profiles via WAN)

– configure facilities

– configure partner systems

– create configuration files for WAN or ISDN subnetwork profiles

In addition, the WAN interfaces for IP must be configured for all participating systems.

A sample configuration is given below; in this case Solaris-1 is the local system and Solaris-2 is the remote system. Each of the Solaris systems is located in a local TCP/IP network. Both LANs are connected via an X.25 network and routing systems. STORES and PURCHASING are the names of the CMX applications that are to communicate with each other.

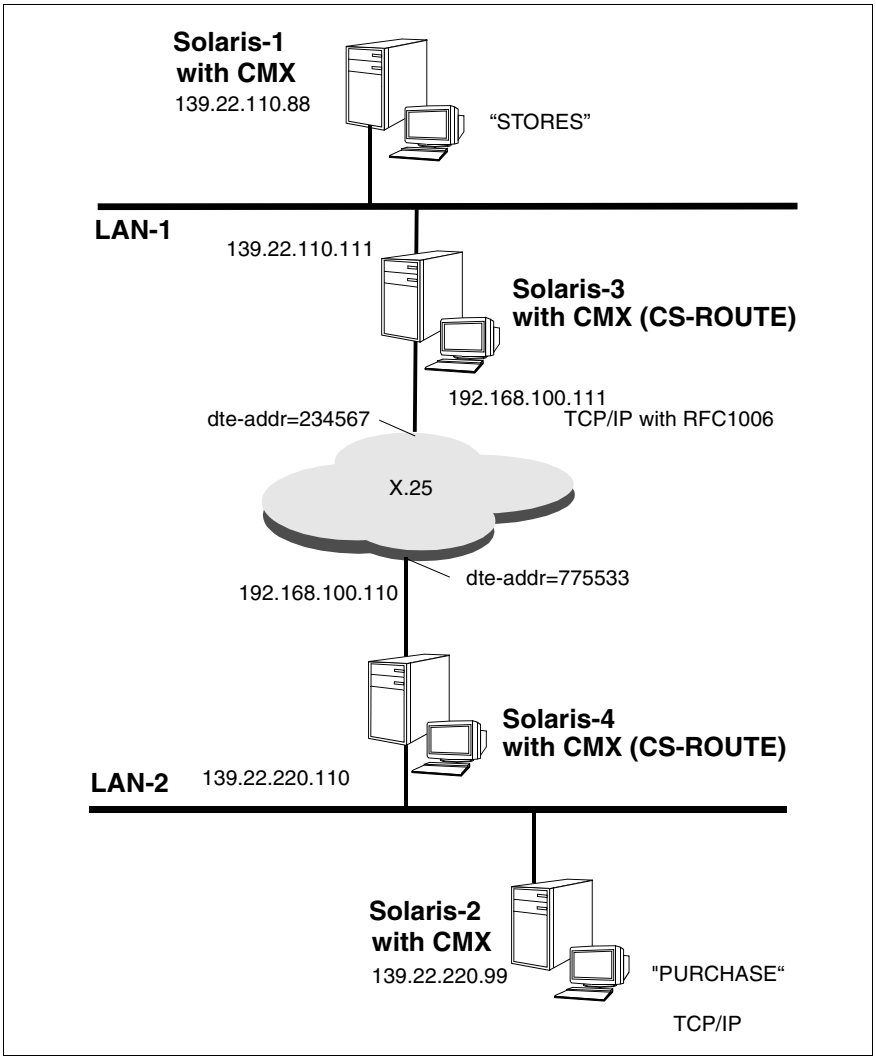


Figure 17: Sample configuration for LAN connection via X.25

7.4.1 Configuring applications

The GLOBAL NAMES of CMX applications should be defined as follows:

Name part	1	2	3	4	5
For the CMX application in system Solaris-1			Moore&Co	Solaris-1	STORES
For the CMX application in system Solaris-2			Moore&Co	Solaris-2	PURCHASING

Table 26: GLOBAL NAMES of sample applications

To enable the CMX application STORES to communicate with the CMX application PURCHASING, the system administrators of the end systems Solaris-1 and Solaris-2 must proceed as follows.

Tasks performed on system Solaris-1:

1. The CMX application STORES must be entered in the TS directory of system Solaris-1. STORES resides in the local system Solaris-1; the LOCAL NAME must therefore be specified for the GLOBAL NAME in Solaris-1. The following entry must be written to a file:

   STORES TSEL RFC1006 A´STORES´

   ► Incorporate the created entry into the TS directory (the default is DIR1) using the following command:

   ```
   tnsxcom –u filename
   ```

   The system administrator of Solaris-1 must now ask the system administrator of Solaris-2 to enter the CMX application STORES in the TS directory of Solaris-2 as a TS application in the remote system. The T-selector STORES must appear on Solaris-2 in the TRANSPORT ADDRESS of STORES (see step 2 for the remote system).

2. To enable a TS application on Solaris-1 to establish a connection with CMX application PURCHASING on Solaris-2, the CMX application PURCHASING must be entered in the TS directory on Solaris-1. From the point of view of Solaris-1, PURCHASING resides on the remote system Solaris-2; the GLOBAL NAME of PURCHASING, the transport system (CCP profile) to be used for the communication, and the TRANSPORT ADDRESS must therefore be specified in Solaris-2. The following entries must be written to a file:

PURCHASING TA RFC1006 139.22.220.99 A´PURCHASING´

► Incorporate the created entry into the TS directory (default is DIR1) using the following command:

```
tnsxcom –u filename
```

Tasks performed on system Solaris-2:

In Solaris-2, the CMX application PURCHASING must be entered as a TS application in the local end system and the CMX application STORES must be entered as a TS application in the remote end system.

1. The entries for PURCHASING must be incorporated as follows:

   PURCHASING TSEL RFC1006 A´PURCHASING´

   ► Incorporate the created entry into the TS directory (default is DIR1) using the following command:

   ```
   tnsxcom –u filename
   ```

2. The CMX application STORES resident on Solaris-2 must be configured as a remote application. The following entry must be written to a file:

   STORES TA RFC1006 139.22.110.88 A´STORES´

   ► Incorporate the created entry into the TS directory (default is DIR1) using the following command:

   ```
   tnsxcom –u filename
   ```

The T-selector STORES must match T-selector of the LOCAL NAME in the CMX application STORES in system Solaris-1.

7.4.2 Configuring routes

For TCP/IP connection of two applications via WAN (here X.25), address entries must be made and routes configured in the FSS for the intermediate systems involved (here Solaris-3 and Solaris-4). This always applies if the routing functions are implemented by the CMX routing service (CS-ROUTE).

To reach the remote system Solaris-2 from the system Solaris-1, you must enter routes in the intermediate systems Solaris-3 and Solaris-4 in order to ensure the accessibility of Solaris-1 and Solaris-2. In addition to the name of the route, the

necessary entry contains a subnetwork ID for the subnetwork to be used (here X25-1), which you must assign, as well as the DTE address of the other intermediate system.

Tasks performed on system Solaris-3:

- Specify a route to the intermediate system Solaris-4:

```
fssadm create SNPAROUTES name=route4 subnet=X25-1 dte-addr=
775533
```

Tasks performed on system Solaris-4:

- Specify a route to intermediate system Solaris-3:

```
fssadm create SNPAROUTES name=route3 subnet=X25-1 dte-addr=
234567
```

7.4.3 Setting facilities

Specific facilities can be assigned to each route and each remote subnetwork interface. For example, you can agree reverse charges so that the connection costs are borne by the partner system. The entry for this facility has the following form:

```
fssadm create FACIL name=charging x25-revch=REQUEST_ONLY
```

Then assign this facility to the configured route:

```
fssadm set SNPAROUTES name=route3 facil=charging
```

This facility can likewise be assigned to the other route, or any other facilities can be defined.

Note here that it is only possible to assign multiple facilities to routes in command mode.

7.4.4 Configuring remote systems

Remote systems that you want to reach via WAN using the TCP/IP protocol, must be entered in the FSS. After you have entered the route via which the remote subnetwork can be reached, enter the network address of the remote system in the FSS.

Tasks performed on system Solaris-3

- Enter an NSAP object which represents Solaris-4. The object contains the IP address of Solaris-4 as well as a reference to the route to be used:

```
fssadm create NSAP name=Solaris4 internet-addr=
192.168.100.110 snpa-list=route4
```

Tasks performed on system Solaris-4

- Similarly, enter an NSAP object which represents Solaris-3:

```
fssadm create NSAP name=Solaris3 internet-addr=
\192.168.100.111 snpa-list=route3
```

7.4.5 Configuring WAN interfaces for IP

Each WAN interface to be used for TCP/IP must be made known to the Solaris system. To do this, enter a WAN IP address.

Tasks performed on system Solaris-3

- The WAN interface is assigned a unique name and the associated IP address:

```
csr create if name=clwip0 ipaddr=192.168.100.111
```

  A file called clw.routes.clwip0 is created by the system.

- The IP entity must be informed that all packets to subnetwork 139.22.220 are to be routed via the local interface 192.168.100.111:

```
route add net 139.22.220 192.168.100.111 1
```

Tasks performed on system Solaris-4

- The WAN interface is assigned a unique name and the associated IP address:

  ```
  csr create if name=clwip0 ipaddr=192.168.100.110
  ```

  A file called clw.routes.clwip0 is created by the system.

- The IP entity must be informed that all packets to subnetwork 139.22.220 are to be routed via the local interface 192.168.100.110:

  ```
  route add net 139.22.110 192.168.100.110 1
  ```

Tasks performed on system Solaris-1

- Configure the route:

  ```
  route add net 139.22.220.0 139.22.110.111 1
  ```

  Alternatively, you can specify the system Solaris-3 as the default router in the */etc/defaultrouter* file. Enter the following:

  139.22.110.111

Tasks performed on system Solaris-2

- Configure the route:

  ```
  route add net 139.22.110.0 139.22.220.110 1
  ```

  Alternatively, you can specify the system Solaris-3 as the default router in the */etc/defaultrouter* file. Enter the following:

  139.22.220.110

8 Web-based CMX administration

CMX ships with the package *SMAWwca* (web-based CMX administration). *SMAWwca* supports access to CMX administration via the PRIMEPOWER ServerView Management GUI. The administration clients can run on Solaris systems (LAN Console or System Management Console) or on Windows systems.

To increase security, the connection between administration client and communication server can be encrypted using SSL/TLS. The add-on package *SMAMswca* that can be installed on a separate administration server is available for this purpose.

The figure below illustrates the different administration variants and indicates which packages must be installed on which systems.

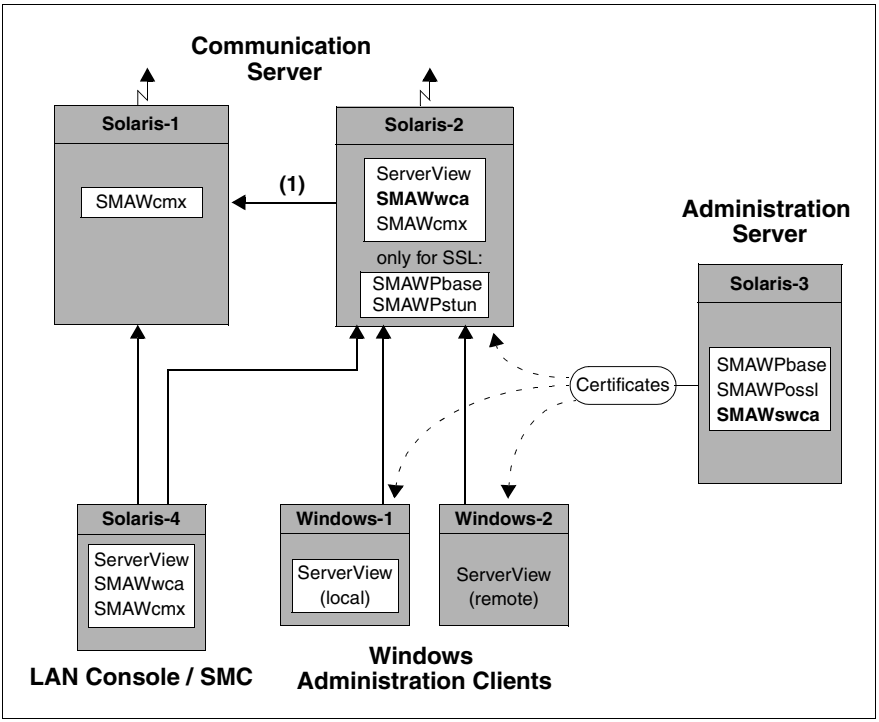


Figure 18: Web-based CMX administration

The Solaris-1 system has only the *SMAWcmx* package. By contrast, all available packages for secure web-based CMX administration are installed on the Solaris-2 system. The following options are therefore available.

– Solaris-2 can be administered directly over the web from all clients.

– The connection between Solaris-2 and the Windows clients Windows-1 and Windows-2 can be protected by means of SSL/TLS. To do this, a server certificate must be generated on the administration server Solaris-3 and then copied to Solaris-2. If self-signed root certificates are used, you must copy them onto the two Windows administration clients.

– Solaris-1 can now be administered directly from the LAN Console/SMC (Solaris-4) because it does not have *SMAWwca*.

  The Windows administration clients can administer Solaris-1 only indirectly via the Java application on Solaris-2; no encryption with SSL/TLS is possible on this connection (1).

8.1 Installation

Before you can administer the communication servers, you must first install the components described below on the communication servers and on the administration clients. You should proceed in the following order.

Communication server without SMAWwca

This server can be administered directly only via the LAN Console/SMC.

– Install the *SMAWcmx* package.

Communication server with SMAWwca

This server can be administered via the LAN console/SMC or Windows clients. Proceed as follows.

- Install ServerView (V2.2 or higher) from the Control DVD.
- Install the *SMAWcmx* package.
- Install the *SMAWwca* package.
  SMAWwca must be selected explicity in custom installation (*custom*).
- Make the necessary additions to the ServerView configuration file using the *add_cmxadm* command, see page 195.

Additional steps are necessary if you want to use SSL/TLS.

- Install *SMAWPbase* and *SMAWPstun* from the Control DVD.
- Copy the server certificate to */opt/SMAW/SMAWcmx/wca/stunnel/certs* (see page 159)

If you want to use IPSec, proceed as described in the section “Encryption with IPSec” on page 164.

Administration server

This server is only needed if you want to use SSL/TLS. It is used for certificate management only. This function can also be assumed by a communication server. Proceed as follows.

- Install first *SMAWPbase* and then *SMAWPossl* from the Control DVD.
- Install the *SMAWswca* package.
  SMAWswca must be selected explicitly in custom installation (*custom*).

LAN Console / System Management Console

You can administer all communication servers directly via the LAN Console or System Management Console. Proceed as follows.

- Install ServerView (V2.2 or higher) from the Control DVD.
- Install the *SMAWcmx* package.
- Install the *SMAWwca* package.
  SMAWwca must be selected explicitly in custom installation (*custom*).
  SMAWwca must always be installed after ServerView and *SMAWcmx*.
- Make the necessary additions to the ServerView configuration file using the *add_cmxadm* command, see page 195.

Windows client with local ServerView

ServerView is to be started as a local win32 application on this administration client. Proceed as follows.

- Install JRE (Java Runtime Environment) in Version 1.4.2 or higher. Java Web Start is therefore also installed. You can call the Java Web Start Application Manager after JRE has been installed by means of the *Execute* command in the Start menu. In the *Execute* window enter the *javaws* command.

  You can download JRE from the server from the ServerView download site (port 8883) if ServerView is installed there. See also figure 21 on page 148.

- Install ServerView as a win32 application.

  You can also download the self-extracting exe file from the server from the ServerView download site (port 8883), see figure 21 on page 148. Executing the exe file adds ServerView to the Windows Start menu.

- Create or add to the *WSAConfig* file as described on page 146.

- Add to the policy file, see page 144.

- If you want to use SSL/TLS:
  Import the root certificate, see page 160ff.

Windows client with remote ServerView

ServerView is to be started as a remote application on this administration client. Proceed as follows.

- Install JRE (Java Runtime Environment) in Version 1.4.2 or higher. Java Web Start is therefore also installed. You can call the Java Web Start Application Manager after JRE has been installed by means of the *Execute* command in the Start menu. In the *Execute* window enter the *javaws* command.

  You can download JRE from the server from the ServerView download site (port 8883) if ServerView is installed there. See also figure 21 on page 148.

- Add to the policy file, see page 144.

- If you want to use SSL/TLS:
  Import the root certificate, see page 160ff.

The ServerView application is started by entering the URL, see the section “Starting ServerView” on page 147.

You are recommended to keep tothe specified installation order. However, the ServerView product and the *SMAWwca*, *SMAWswca*, *SMAWPbase*, *SMAWPstun* and *SMAWPossl* packages can be installed at a later time.

Readme files

The *SMAWwca* readme files are located on the server on which *SMAWwca* was installed. You can access them as follows.

– On the client:
  Call up the CMX_ADM download site in the browser and click on the *README* option, see figure 19 on page 142.

– On the administered communication server:
  Enter the following command.
  `pg /opt/SMAW/SMAWcmx/wca/README`.

8.2 Configuring the client

ServerView and web-based CMX administration use Java™ Web Start technology. To permit execution of web-based CMX administration in this environment, you must create the configuration file *.java.policy*. If ServerView is to be started locally as a win32 application, you must also create the local ServerView configuration file *WSAConfig*.

Template files are shipped with the *SMAWwca* package for this purpose and you can download them from the server. After installing the package, enter the following URL in the browser of the administration client.
`http://`*comm-server*`:8881/CMX/CMX_ADM_download.htm`
(*comm-server* = communication server on which *SMAWwca* is installed).

You are taken to the following page:

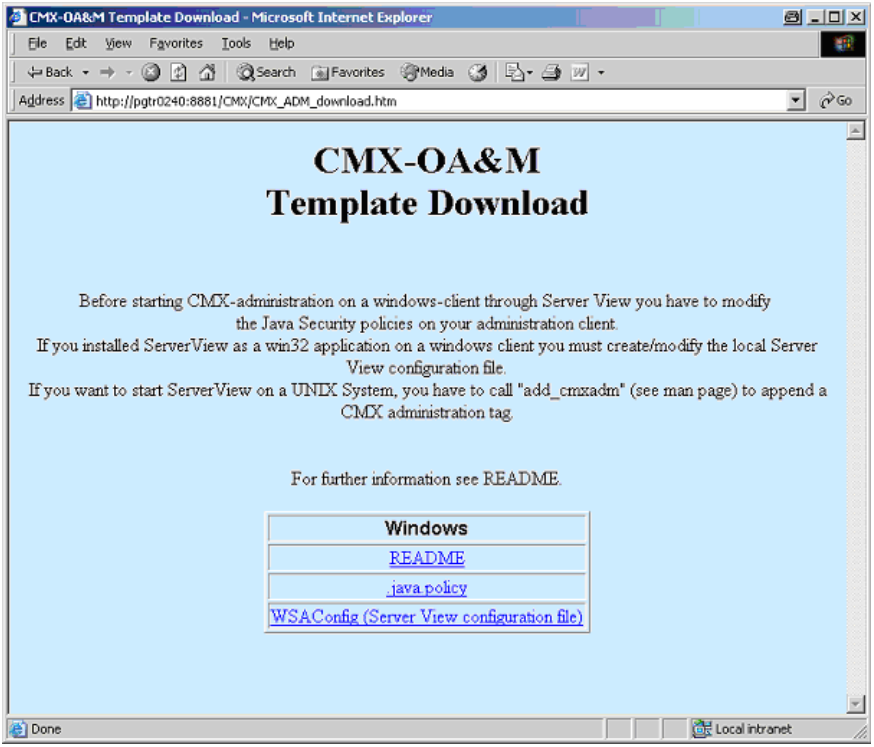


Figure 19: CMX_ADM download site

8.2.1 Java security settings

If you start ServerView using Java Web Start, ServerView has only restricted rights; for example, Java Web Start may not access a remote jnlp file. It is therefore necessary to extend the Java security settings on every administration client.

The Java security settings are defined with the help of policy files. Before you can use web-based CMX administration, you must therefore generate or add to the user-specific file *.java.policy*. You can download template files with the new/modified security settings from the CMX_ADM download site (see above).

Changing security settings

You can change the security settings in one of the following three ways.

- Copy the *.java.policy* file to *%USERPROFILE%/.java.policy* if it does not exist already, or
- Add the entries in the template file *.java.policy* to the existing private policy file *%USERPROFILE%/.java.policy*, or
- Change the security settings using the *policytool* tool (MS-DOS window).

%USERPROFILE% contains the value of the USERPROFILE variable, see below.

Preparations

In all three cases above you must first determine the following data.

1. Value of the *USERPROFILE* variable

   To do this, use the *set* command or the *echo %USERPROFILE%* command (in the MS-DOS window).

2. Value of the *USERNAME* variable

   To do this, use the *set* command or the *echo %USERNAME%* command (in the MS-DOS window).

3. Path of the Java Web Start directory (cache file)

   This path depends on the installed JRE version. For JRE 1.5.0, the path is usually as follows.

   file:c:/Profiles/%USERNAME%/Application Data/Sun/Java/Deployment/cache/javaws/http/-

   This is the default path in the Java policy template. You need only replace *%USERNAME%* with your user name and adapt the device name if necessary. Check whether this path matches your installation. You can find the path used in your Java installation in the Java Web settings by calling Java Web Start (javaws) from the Windows Start menu, *run* command. You get the following dialog box:

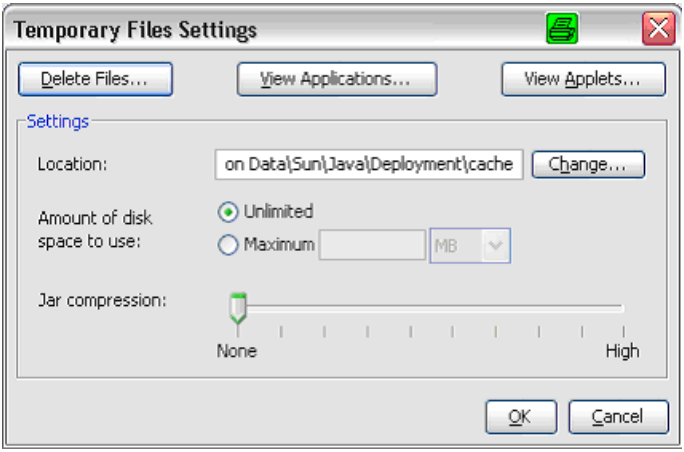


Figure 20: Setting of Java Web Start

For the entry in the policy file replace „\“ with „/“ and add *javaws/http/-* to the name of the applications folder.

Downloading the .java.policy file from the CMX_ADM download site

- On the download page select the file *.java.policy* with the right mouse button, see figure 19 on page 142.
- Choose *Save as*.
- Select the file name *USERPROFILE* and set *type* to *all files*.
- Choose *Save*.
- Check the entry for the Java Web Start cache file. You must always replace *USERNAME* with your own user name using any text editor.

Adding to an existing .java.policy file

The *.java.policy* file is an ASCII file and can therefore be edited using any ASCII editor.

- Copy the entries from the template file *.java.policy* into the private file *%USERPROFILE%/.java.policy*.
- Check the entry for the Java Web Start cache file. You must always replace *USERNAME* with your own user name using any text editor.

Using the policytool tool on the client

- Start the program *JREHOME\bin\policytool*, e.g. in an MS-DOS window or by selecting *Start* -> *Run* and entering *JREHOME*`/bin/policytool`.

  Default for *JREHOME* is *C:/Program Files/Java\j**.

- Select *File* -> *Open* to open the corresponding file *.java.policy*.
- Choose *Add Policy Entry*.
  - Define the code base, i.e. the link to the Java Web Start application cache, see above.
  - Click on the *Add permission* button.
  - Select *File Permission*.
  - For *Target Name* choose the value *<<ALL FILES>>*.
  - Specify execute as *action*.
  - Click on *OK* and *Done* to confirm your input.
- Save the configuration using *File* -> *Save*.
- Close the *policytool*.

8.2.2 WSAConfig file

To enable ServerView to be started as a win32 application on the administration client, you must download the local ServerView configuration file *WSAConfig* from the CMX_ADM download site.

The cofiguration file provided by the *SMWAwca* package is an extension of the file delivered with ServerView. It contains an entry for CMX (in addition to the entries for AlarmService and ARMTech). If no other foreign application is to be supported you can directly copy the *WSAConfig* file into the *%Program-Files%/Fujitsu SiemensComputers/WebSysAdmin* directory. Otherwise extend the the existing file with the CMXADM tag, see loaded *WSAConfig* file: [*CMXADM*].....#*CMXADM_END*.

WSAConfig includes a link to the Java Web Start application; the default is the pathname of the English JRE installation. If the JRE is installed in a different directory, you must modify *WSAConfig* as follows.

- On the CMX_ADM download site select the *WSAConfig* file by clicking on it with the right mouse button, see figure 19 on page 142.
- Choose *Save as*.
- Select the pathname SYSTEM(C:) and set *type=all files*.
- Choose *Save*.
- Open the *WSAConfig* file in any text editor, check and, if necessary, change the entry for *javaws.exe*.

8.3 Configuring the communication server

To enable ServerView to access CMX administration, you must make additions to the ServerView configuration file *WSAConfig* on the communication server. You do this using the *add_cmxadm* command as described on page 195.

The *del_cmxadm* command lets you remove individual entries, see page 196.

Caution
If *SMAWwca* is deinstalled, all CMX-specific entries are deleted from the ServerView configuration file *WSAConfig*.

8.4 Activating and deactivating SMAWwca

Once *SMAWwca* and, if necessary, *SMAWstunnel* have been installed, activate the administration software on the communication server as follows.

- Call the *wca_init start* command (see page 200). This makes available all configuration files required by *SMAWwca*. *Stunnel* is also started if it has already been installed and the required certificates have been generated. Otherwise you can start *Stunnel* at any time later using the *wca_stunnel start* command (see page 200).
- Enter the *add_cmxadm* command (see page 195). This permits access to CMX administration via ServerView.

Deactivating SMAWwca

You should deactivate *SMAWwca* using the *wca_init stop* command before deinstalling the *SMAWwca* package. This cancels the changes made in the system when web-based CMX administration was started.

8.5 Starting ServerView

How you start ServerView depends on the installation type and platform.

- ServerView in Windows as a remote application:

  You have two start options.

  – Start by entering the URL *http://comm-server:8881*

  – Start by entering the URL *http://comm-server:8883* and then selecting the PRIMEPOWER ServerView Suite (remote application using Java Web Start), see figure 21 on page 148.

  comm-server is the name of the communication server on which ServerView is installed.

- ServerView in Windows as a local win32 application:

  Call ServerView using the corresponding entry in the Windows Start menu.

- ServerView as a Solaris application:
  (gaphic console on GP7000F, PRIMEPOWER or PRIMESTATIION)

Start the ServerView Management GUI using the following shell script.

`/opt/SMAW/bin/wsa` [`sync`] [*hostname*]

The ServerView start site is displayed, see figure 22 on page 149. Note that the DISPLAY variable of the Xwindows system must be set, e.g. *DISPLAY=sytstem-address:0.0*.

Further information on ServerView is provided in the manual entitled „PRIME POWER ServerView Suite2.2“.

ServerView download site

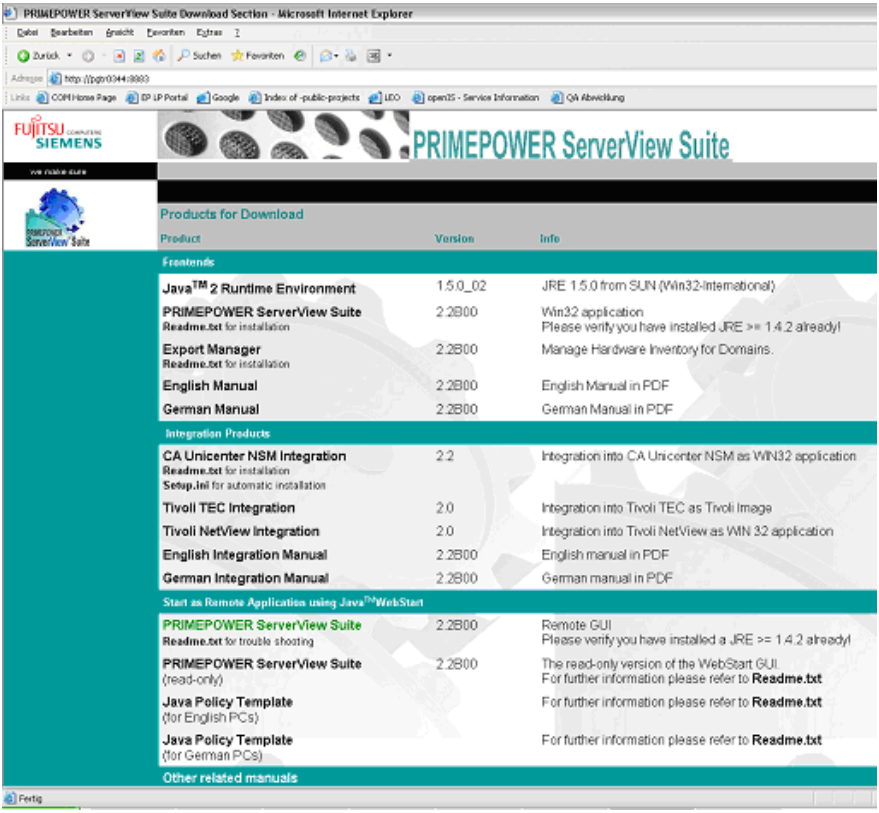


Figure 21: ServerView download site

ServerView start site

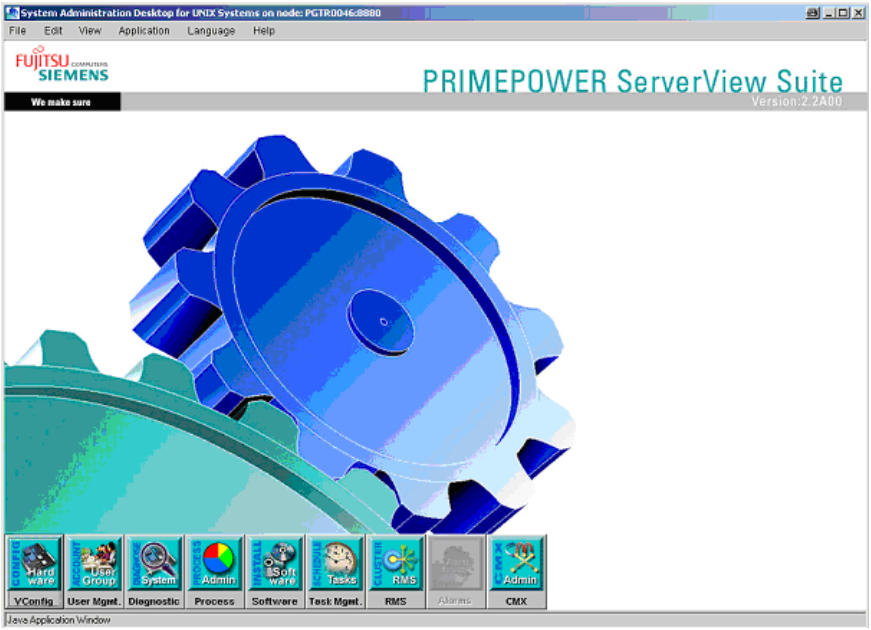


Figure 22: ServerView start site

8.6 Starting the CMX administration interface

After installing ServerView and *SMAWwca* on the communication server, you can start CMX administration (CLI or CUI) on the administration client in one of the following ways.

- Start ServerView and click on the *CMXADM* button on the start site, or
- Start ServerView and choose *Application* -> *CMXADM*, or
- Double-click on the CMXADM icon (if available).

> The Telnet application that starts CMX administration is implemented as a Java Web Start application. Java Web Start therefore automatically generates shortcuts on the Windows desktop and in the Start menu. By default, Java Web Start prompts you to create a shortcut when the application is started for the second time. You can change this in the control panel. CMX administration can then be started directly by double-clicking on the application icon.

The CMXADM start site is displayed in the browser.

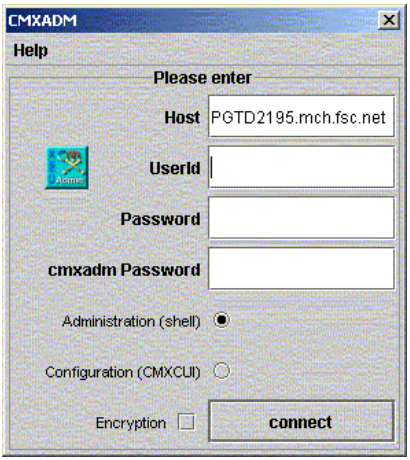


Figure 23: CMXADM start site

- Enter the name of the communication server in the *Host* field. *SMAWcmx* must be installed on this server.
  Default: Name of the server on which *SMAWwca* is installed.

- Enter the user ID, associated password and administrator password (*cmxadm* role) of the communication server. The password for the *cmxadm* role is optional and is meaningful only if administration is to take place from a user ID to which the *cmxadm* role is assigned.
- On the start site select either *Administration (shell)* or *Configuration (CMXCUI)*.
- If you want to encrypt data using SSL/TLS, select the *Encryption* checkbox.
- Click on the *Connect* button.

A shell session or CMXCUI is started.

Entry of the user ID *root* and associated password is accepted for reasons of compatibility.

8.7 Security

CMX administration is operated over a Telnet connection between Windows client and the server to be administered. Telnet is a TCP/IP protocol lacking any security, i.e. data is transmitted over the network without encryption. This applies especially to user IDs and the related passwords.

To improve security, we strongly recommend the use of SSL/TLS or IPSec. This is especially important where the administration client and the communication server are not located in a private LAN.

8.7.1 Encryption using SSL/TLS

Versions SSLv2 and SSLv3 of the Secure Sockets Layer (SSL) protocol were published by Netscape Communications Corporation in 1994 and 1995. On this basis, the Internet Engineering Task Force (IETF) defined the Internet standard Transport Layer Security protocol Version 1 (TLS 1.0) that was published in January 1999 as RFC2246 and was supplemented with RFC3546 in June 2003.

In the World Wide Web, SSL/TLS is most frequently used as a protocol layer between TCP and HTTP to **encrypt** data traffic between web servers and web browsers and to **authenticate** these two communication partners.

In CMX administration SSL/TLS is used to secure communication via a Telnet connection. SSL/TLS is implemented by means of the *OpenSSL* and *Stunnel* components because the Telnet application does not support the SSL/TLS protocol.

– *OpenSSL* is a freely available implementation of the SSL/TLS protocol. The SSL library (*libssl*) implements all protocol versions of SSLv2, SSLv3 and TLS 1.0. The cryptographic library (*libcrypto*) provides common algorithms for cryptography and also supports certificate and key management. *SMAWswca* uses the corresponding *OpenSSL* functions to create certificates.

– *Stunnel* is, like *OpenSSL*, freely available. *Stunnel* is used to encrypt network communication of services that do not support any cryptography functions, such as Telnet for example. *Stunnel* functions as an SSL wrapper.

More information on *OpenSSL* and *Stunnel* is available at the following URLs.

http://www.openssl.org
http://www.stunnel.org
http://www.tldp.org/HOWTO/SSL-Certificates-HOWTO/index.html

8.7.1.1 Mode of operation of SSL/TLS

The figure below illustrates the flow of data between administration client and communication server when SSL/TLS is used for encryption purposes.

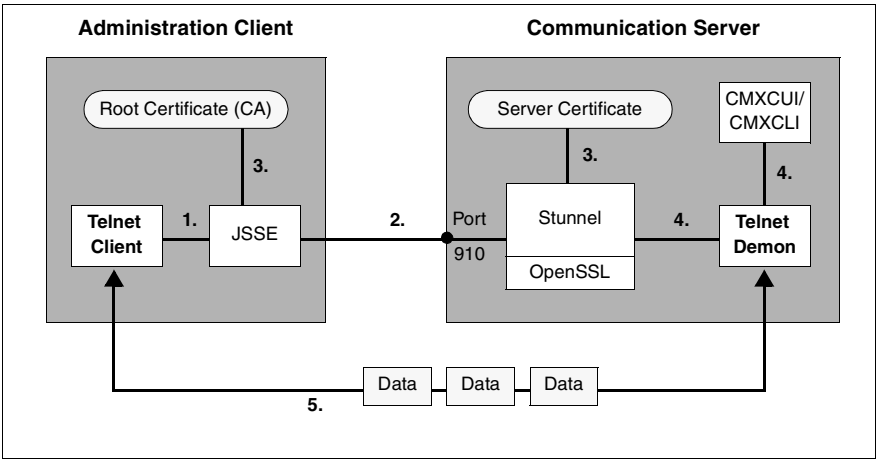


Figure 24: Connection setup and data flow when using SSL/TLS

Explanation:

1. When *Encryption* is selected on the CMXADM start site, the Telnet client initiates a connection to the communication server via JSSE. JSSE (Java Secure Socket Extension) is included in Java JRE 1.4 or higher and makes a Java version of the SSL/TLS protocol available. JSSE includes, for example, authentication and encryption functions.
2. First a connection is set up to the SSL wrapper *Stunnel* via port 910 (default). This port number is defined in the *Stunnel* configuration files, see the section “Using Stunnel” on page 163. The port number is set by default to 910 for CMX administration.

   SSL version and certificates are exchanged via this connection using the SSL handshake protocol.

3. The certificates are checked and an SSL/TLS connection is set up. The protocols are then able to operate securely on this connection.
4. *Stunnel* establishes a connection to the Telnet daemon that in turn starts CMXCUI or CMXCLI.
5. Data is now exchanged between the administration client and CMXCUI or CMXCLI on the communication server via a secure SSL/TLS connection.

8.7.1.2 Requirements for the use of SSL/TLS

If you want to exchange information between clients and communication servers via a secure SSL connection, you must perform the following steps.

- On the administration server:
  - Install *SMAWPbase* and *SMAWPossl* from the Control DVD.
  - Install *SMAWswca*.
  - Create a server certificate for use with *Stunnel*.
- On all communication servers:
  - Install *SMAWPbase* and *SMAWPstun* from the Control DVD.
  - Copy the server certificate to */opt/SMAW/SMAWcmc/wca/stunnel/certs*
  - Start *Stunnel* for CMX administration.
- On all administration clients:
  - Import the public key.

8.7.1.3 Generating certificates with Stunnel

To use the functions of *Stunnel* in conjunction with *SMAWwca*, you must generate a certificate/key pair in a precisely defined format. You must ensure that the private key is not encrypted because *Stunnel* is not able to prompt the user to enter the password for the key.

You can obtain a certificate from any Certificate Authority (**CA**), e.g. VeriSign, TC TrustCenter or any other official CA. To do this, you must send a Certificate Signing Request (CSR) to the certificate authority. The CSR contains the public key and the data of the requester. The CSR is signed by the CA after the data of the requester has been checked, and a suitable certificate is generated. The public key of the CA is publicly available.

CA certificates are preinstalled in some JRE versions. This ensures that the certificates issued and signed by the preinstalled CAs can be verified. For test purposes or if you are using your communication server in an intranet only, you can also generate a private CA and use it to create a certificate.

To simplify the use of certificates, scripts are installed with *SMAWswca*. These enable you to generate self-signed certificates, a certificate signing request, and the server certificate expected by *Stunnel*. You do this as follows.

1. Generate a demo CA with *manage_cert -newca*
2. Generate a private key with *manage_cert -newkey*
3. Generate a certificate signing request with *manage_cert -newreq*
4. Sign the certificate request using a private CA (*manage_cert -sign*)
5. Create the server certificate with *manage_cert -finish*

Steps 1 and 4 are not needed if your certificate is issued by an official CA.

The individual steps are explained in more detail below. Details on the *manage_cert* command are provided in the section “manage_cert - Manage certificates” on page 197.

Step 1: Generate a demo CA

You generate a private certificate authority named **Snakeoil-CA** using the *manage_cert -newca* command. You can have your certificate signing request signed by this CA. Note that a certificate signed in this way is not trustworthy and should be used for test purposes only.

After calling the *manage_cert -newca* command you are prompted to answer a series of questions. You are shown the default answers to these questions as they are set in the configuration file. Press ENTER to accept these defaults.

Several directories and files are then created; for example, the files for the public and the private key:
/opt/SMAW/SMAWswca/PrivateCA/certs/wca_cacert.pem (public key)
/opt/SMAW/SMAWswca/PrivateCA/private/wca_cakey.pem (private key)

You are advised to save the files generated as otherwise you cannot extend a certificate that has already been issued.

Example:

```
# manage_cert -newca
CA certificate filename (or enter to create)  <enter>

Making CA certificate ...
Generating a 1024 bit RSA private key
....++++++
.....................................................++++++
writing new private key to
'/opt/SMAW/SMAWswca/PrivateCA/private/wca_cakey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XY]:                                )  <enter>
State or Province Name (full name) [Snake Desert]:                )  <enter>
Locality Name (eg, city) [Snake Town]:                            )  <enter>
Organization Name (eg, company) [Snake Oil, Ltd]:                 )  <enter>
Organizational Unit Name (eg, section) [Certificate Authority]:   )  <enter>
Common Name (eg, CA name) [Snake Oil CA]:                         )  <enter>
Email Address [ca@snakeoil.dom]:                                  )  <enter>
```

By default, the certificate is valid for 365 days. If you want to specify a different validity period, you can do so with the additional option *-days number*, see section “manage_cert - Manage certificates” on page 197 .

Step 2: Generate a private key

You generate a private key with the *manage_cert -newkey* command. You are advised to create a backup copy of the key because the certificate loses its validity if you lose the key.

A private key is generated in */opt/SMAW/SMAWswca/ssl/private/wca_privkey.pem*

Example:

```
# manage_cert -newkey
Create private key
Generating RSA private key, 2048 bit long modulus
.....+++

.........................+++
e is 65537 (0x10001)
Private key is in /opt/SMAW/SMAWswca/ssl/private/wca_privkey.pem
```

Step 3: Generate certificate request

You generate a certificate request for the communication server *comm-server* using the *manage_cert -newreq comm-server* command. You can send this request to an official certificate authority or sign it using your private CA.

After calling the *manage_cert -newreq* command you are prompted to answer a series of questions. You are shown the default answers to these questions as they are set in the configuration file. Press ENTER to accept these defaults. *Common Name* is the name of the system on which *Stunnel* is running. For *Common Name* you must always specify the full system name (see output of the *nslookup system-name* command).

A certificate request is generated in
/opt/SMAW/SMAWswca/ssl/req/comm-server.newreq.pem

Example:

```
# manage_cert -newreq myHost
Create a certificate request
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XY]:
State or Province Name (full name) [Snake Desert]:
Locality Name (eg, city) [Snake Town]:
Organization Name (eg, company) [Snake Oil, Ltd]:
Organizational Unit Name (eg, section) [Development]:
Common Name (hostname of the machine) [local host]:myHost.aaa.bbb.ccc
Email Address [ca@snakeoil.dom]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password ****
Request is in /opt/SMAW/SMAWswca/ssl/req/myHost.newreq.pem
```

Step 4: Sign certificate signing request using a private CA

You sign your certificate signing request with the private key of your CA using the *manage_cert -sign comm-server* command (*comm-server* = name of the communication server). The command implicitly uses the file */opt/SMAW/SMAWswca/ssl/req/comm-server.newreq.pem* as input.

The signed certificate is issued in */opt/SMAW/SMAWswca/ssl/certs/comm-server.newcert.pem*.

Example:

```
# manage_cert -sign myHost
Using configuration from /opt/SMAW/SMAWswca/conf/openssl_ca.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Jul 19 10:15:12 2004 GMT
            Not After : Jul 19 10:15:12 2005 GMT
        Subject:
            countryName               = XY
            stateOrProvinceName       = Snake Desert
            localityName              = Snake Town
            organizationName          = Snake Oil, Ltd
            organizationalUnitName    = Development
            commonName                = myHost
            emailAddress              = ca@snakeoil.dom
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                4D:B1:EC:9D:53:C7:EC:3E:A2:1A:41:1B:DC:A6:2F:04:9A:8E:B5:54
            X509v3 Authority Key Identifier:

keyid:E3:75:EE:63:28:B7:9A:1B:FB:77:6B:94:B4:4E:FC:6D:E9:97:21:62
                DirName:/C=XY/ST=Snake Desert/L=Snake Town/O=Snake Oil,
Ltd/OU=Certificate Authority/CN=Snake Oil CA/emailAddress=ca@snakeoil.dom
                serial:00

Certificate is to be certified until Jul 19 10:15:12 2005 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y

Certificate:
……


-----END CERTIFICATE-----
Signed certificate is in /opt/SMAW/SMAWswca/ssl/certs/myHost.newcert.pem
```

Step 5: Create server certificate for Stunnel

You can create the server certificate on any administration server on which the *SMAWswca*, *SMAWPbase* and *SMAWPossl* packages have been installed.

Stunnel expects the data in a precisely defined format. The final certificate must contain the private key, the certificate signing request, and the signed certificate. You create the associated file using *manage_cert -finish comm-server* (*comm-server* = name of the communication server).

The input file for *Stunnel* is created in */opt/SMAW/SMAWswca/ssl/certs/comm-server.pem*.

Example

```
# manage_cert –finish myHost
Final certificate is in /opt/SMAW/SMAWswca/ssl/certs/myHost.pem
```

Steps 1 and 4 are not needed if you sign your request using an official CA. In step 5 you must copy the certificate signed by the CA and save it with the following name:
/opt/SMAW/SMAWswca/ssl/certs/comm-server.newcert.pem.

8.7.1.4 Copy server certificate onto communication server

In step 5 you must copy the certificate generated onto the communication server and store it in the following directory:
/opt/SMAW/SMAWcmx/wca/stunnel/certs

8.7.1.5 Import root certificate to administration client

How you proceed depends on whether you are using a private root certificate or an official root certificate.

Importing a private root certificate

If you have signed your certificate signing request with your own CA, you must import the private root certificate on each administration client.

- To do this, copy the certificate */opt/SMAW/SMAWswca/PrivateCA/certs/wca_cacert.pem* from your administration server to the client.
- Import the certificate using the *keytool -import* command, see syntax description below.

  Specify the option *-keystore cacerts*. The default password for the keystore file *cacerts* is *changeit*. You are prompted to enter this password each time this file is accessed. You must also specify the option *-alias* ... to uniquely identify the entry.

  Syntax of the *keytool -import* command according to JSSE.

  keytool ␣**-import** {␣**-alias**␣alias} {␣**-file**␣cert_file} [␣**-keypass**␣keypass]
  {␣**-noprompt**} {␣**-trustcacerts**} {␣**-storetype**␣storetype}
  {␣**-keystore**␣keystore} [␣**-storepass**␣storepass]
  [␣**-provider**␣provider_class_name] {␣**-v**} {␣**-J**javaoption}

  Syntax notation in accordance with JSSE differs from the notation in the rest of this manual.

  {...} means that a default value is assumed if the option is not specified.

  [...] means that the value is prompted for if it is not enetered and is not defined in the security properties file.

The *keytool* command is not usually located in the path indicated by the path variable. To simplify entry, add the JRE path to the path variable *C:\Program Files\Java\jre1.5.0_02\bin*, e.g. via the Start menu: *Start -> Settings -> Control Panel -> System -> Advanced -> Environment Variables*.

Example 1: Importing a CA certificate

You import the CA certificate *wca_cacert.pem* using the *keytool -import* command, see figure 25. The certificate was copied beforehand to *P:\rg\wca_cacert.pem*. You must select *C:\Program Files\Java\j2re1.4.2_04\lib\security\cacerts* as the keystore file. If necessary, you must switch to the directory *C:\Program Files\Jaca\j2re1.4.2_04\lib\security* because the keystore file is always generated/expected in the current directory.

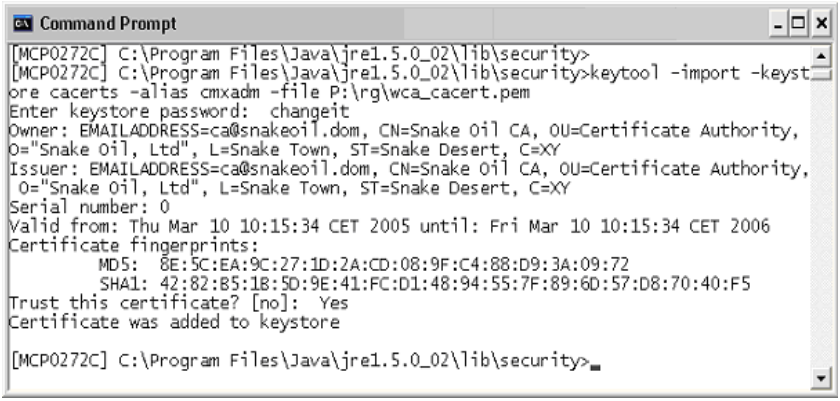


Figure 25: Importing a certificate using keytool -import

Example 2: Deleting the root certificate

If you no longer need the root certificate, you can delete it from the keystore file using the *keytool -delete* file.

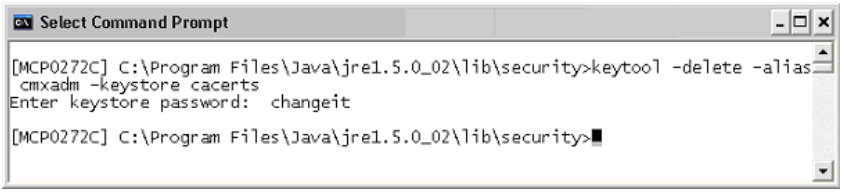


Figure 26: Deleting a certificate using keytool -delete

Importing an official root certificate

A series of certificates of well-known certificate authorities are installed with the Java Runtime Environment. If your certificate signing request was signed by one of these CAs (official root certificate), no changes are needed on the client side. You can obtain a list of well-known root certificates using the *keytool -list* command.

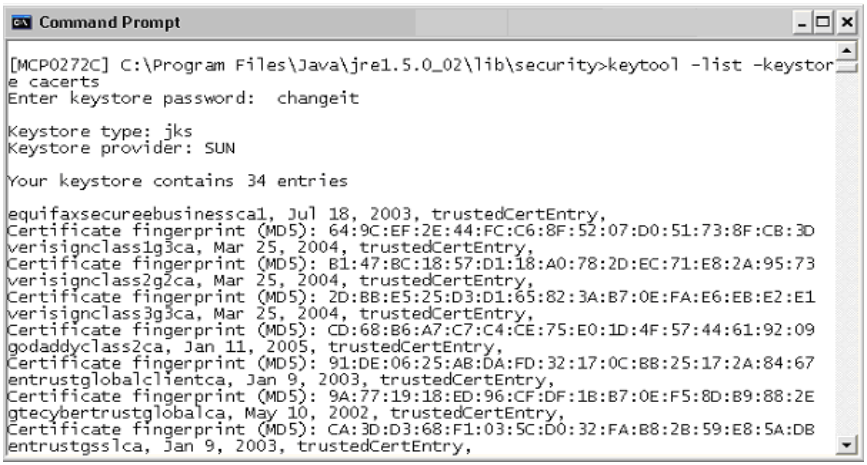


Figure 27: Outputting certification information using keytool -list

If you have your certificate signing request signed by a CA that is not known in the Java environment on your administration client, you must ask the certificate authority you have selected for the public key. You then import this key as described in step 1 on page 155.

8.7.1.6 Using Stunnel

Stunnel must be configured and started on the communication server to ensure that data can be exchanged between client and communication server via a secure SSL connection. In addition, a signed certificate must also be present in */opt/SMAW/SMAWcmx/wca/stunnel/certs/comm-server.pem*, see “Step 5: Create server certificate for Stunnel” on page 159.

Configuring Stunnel

When web-based CMX administration is activated using *wca_init start*, a configuration file with the following name is generated:
/opt/SMAW/SMAWcmx/wca/stunnel/conf/stunnel_wca.conf

This file contains all relevant information for starting *Stunnel*:
– directory and name of the server certificate file
– port number used
– name and arguments of the program that is started

It may be necessary to modify *stunnel_wca.conf*. *Stunnel* expects the server certificate in */opt/SMAW/SMAWcmx/wca/stunnel/certs/comm-server.pem*. *Stunnel* also listens to port 910 by default. If this port is already reserved in your system environment, you must select a free port and enter its port number in all configuration files using the *set_port* command, see page 199.

Starting and stopping Stunnel

After successful start of *SMAWwca* and booting of the communication server, *Stunnel* is automatically started if the server certificate specified in the configuration file is present and if port 910 is not reserved by another application.

You can also start *Stunnel* later using the *wca_stunnel start* command, and stop it using the *wca_stunnel stop* command, see page 200.

Note that *Stunnel* must be restarted after each certificate change.

8.7.2 Encryption with IPSec

You can use the IPSec protocols „Encapsulating Security Payload“ (ESP) and „Authentication Header“ (AH) to protect IP datagrams. AH features data authentication, data integrity and protection against repeated sending of packets. The ESP protocol also offers trustworthy traffic flow.

If you want to use IPSec to ensure secure communications over the Telnet connection, the IPSec settings on server and client must be agree with each other.

IPSec operates in transport mode and uses the AH (authentication algorithm MD5) as well as ESP (encryption algorithm DES) protocol, i.e. the entire IP datagram is authenticated and the data is encrypted.

The security service provided by IPSec requires the use of common keys to carry out the authentication procedure and/or trust mechanisms. The Internet Key Exchange protocol (IKE) is used to manage these keys (Solaris Version 9 and higher).

The method for authorization is "preshared keys". Keys on either side must match.

The example below illustrates an IPSec configuration for data transfer between an administration server under Solaris V9 and an administration client running under Windows >2000. In this example, data traffic via the Telnet port (port 23) is protected using IPSec.

8.7.2.1 Server configuration (Solaris V9)

The configuration of IPSec under Solaris V9 is divided into the following steps:

- Generating or extending a Security Policy Database (SPD)
- Generating or extending IKE security settings
  - configuration file (IKE Policy File)
  - key for IKE authentication (ike.preshared file)
- loading SPD
- starting IKE daemon

Security Policy Database (SPD) – /etc/inet/ipsecinit.conf

The file */etc/inet/ipsecinit.conf* contains all IPSec security settings used to determine the way data transfer is to be monitored.

If the file */etc/inet/ipsecinit.conf* does not exist on your system it must be created.

If it does exist, you must add the following entry:

```
# telnet traffic,  AH authentification: md5, ESP encryption: des,
# ESP authentification: md5, shared association
#
{lport 23} ipsec {auth_algs md5 encr_algs des encr_auth_algs md5 sa
shared}
```

Example: /etc/inet/ipsecinit.conf

```
#
#ident"@(#)ipsecinit.sample1.601/10/29 SMI"
#
# Copyright (c) 1999,2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# This file should be copied to /etc/inet/ipsecinit.conf to enable IPsec
# systemwide policy (and as a side-effect, load IPsec kernel modules).
# Even if this file has no entries, IPsec will be loaded if
# /etc/inet/ipsecinit.conf exists.
#
# Add entries to protect the traffic using IPSEC. The entries in this
# file are currently configured using ipsecconf from inetinit script
# after /usr is mounted.
#
# For example,
#
# {rport 23} ipsec {encr_algs des encr_auth_algs md5}
#
# Or, in the older (but still usable) syntax
#
#       {dport 23} apply {encr_algs des encr_auth_algs md5 sa shared}
#       {sport 23} permit {encr_algs des encr_auth_algs md5}
#
# will protect the telnet traffic originating from the host with ESP using
# DES and MD5.  Also:
#
# {raddr 10.5.5.0/24} ipsec {auth_algs any}
#
# Or, in the older (but still usable) syntax
#
#       {daddr 10.5.5.0/24} apply {auth_algs any sa shared}
#       {saddr 10.5.5.0/24} permit {auth_algs any}
#
# will protect traffic to/from the 10.5.5.0 subnet with AH using any
available
# algorithm.
#
# To do basic filtering, a drop rule may be used. For example:
#
# {lport 23 dir in} drop {}
```

```
# {lport 23 dir out} drop {}
#
# will disallow any remote system from telnetting in.
#
# WARNING:This file is read before default routes are established, and
#before any naming services have been started.  The
#ipsecconf(1M) command attempts to resolve names, but it will
#fail unless the machine uses files, or DNS and the DNS server
#is reachable via routing information before ipsecconf(1m)
#invocation.  (E.g. the DNS server is on-subnet, or DHCP
#has loaded up the default router already.)
#
#It is suggested that for this file, use hostnames only if
#they are in /etc/hosts, or use numeric IP addresses.
#
#If DNS gets used, the DNS server is implicitly trusted, which
#could lead to compromise of this machine if the DNS server
#has been compromised.
#####################################################################
#
# telnet traffic,  AH authentification: md5, ESP encryption: des,
# ESP authentification: md5, shared association
#

{lport 23} ipsec {auth_algs md5 encr_algs des encr_auth_algs md5 sa shared}
```

IKE policy file – /etc/inet/ike/config

The file */etc/inet/ike/config* is used to define the rules for IKE requests.

The following entries must be included:

```
p1_lifetime_secs 28800
p1_nonce_len 20

## Values for p1_xform parameter must conform with the entries
## in /etc/inet/ipsecinit.conf !!!
p1_xform { auth_method preshared oakley_group 2 auth_alg md5 encr_alg des }
p2_pfs 2

### Rules (for every administration client a
### particular rule must be generated):

{
   label "<string>"
   local_id_type ip
   local_addr <eigene IP-Adresse>
   remote_addr <IP address of an administration client>
}
```

Example: /etc/inet/ike/config

```
#
#ident"@(#)config.sample1.201/12/06 SMI"
#
# Copyright (c) 2001 by Sun Microsystems, Inc.
# All rights reserved.
```

```
##
## This file should be copied into /etc/inet/ike/config to enable the
## launch of the IKE daemon, in.iked(1m), at boot time.  You can also
## launch the IKE daemon after creating this file without rebooting by
## invoking /usr/lib/inet/in.iked with a root shell.
##

# Consult the ike.config(4) man page for further details.  Here is a small
# example from the man page.

### BEGINNING OF FILE

### First some global parameters...

## certificate parameters...

# Root certificates.  I SHOULD use a full Distinguished Name.
# I MUST have this certificate in my local filesystem, see ikecert(1m).
#cert_root    "C=US, O=Sun Microsystems\\, Inc., CN=Sun CA"

# Explicitly trusted certs that need no signatures, or perhaps self-signed
# ones.  Like root certificates, use full DNs for them for now.
#cert_trust    "EMAIL=root@domain.org"

# Where do I send LDAP requests?
#ldap_server   "ldap1.domain.org,ldap2.domain.org:389"

# Some PKI-specific tweaks...
# If you wish to ignore CRLs, uncomment this:
#ignore_crls
# If you wish to use HTTP (with name resolution) for URLs inside certs,
# uncomment this:
#use_http
# HTTP proxy and socks URLs should also be indicated if needed...
#socks "socks://socks-relay.domain.org"
#proxy "http://http-proxy.domain.org:8080"

## Phase 1 transform defaults...

p1_lifetime_secs 28800
p1_nonce_len 20

## Parameters that may also show up in rules.

p1_xform { auth_method preshared oakley_group 2 auth_alg md5 encr_alg des }
p2_pfs 2

### Now some rules...

{
   label "Client1"
   local_id_type ip
   local_addr 172.25.124.140
   remote_addr 172.25.123.64
}
{
   label "Client2"
   local_id_type ip
   local_addr 172.25.124.140
```

```
      remote_addr 172.25.123.153
   }
```

IKE preshared file – /etc/inet/secret/ike.preshared

The file */etc/inet/secret/ike.preshared* contains the key for IKE authentication.

An entry in the following format is required for every administration client:

```
{
          localidtype IP
          localid <local IP address>
          remoteidtype IP
          remoteid <IP address of the administration client>
          key <hexstring, 16 characters in alignment with partner configuration>
}
```

Here you can choose an individual kea for each administration client.

Example: /etc/inet/secret/ike.preshared

```
   #
   #ident"@(#)ike.preshared1.101/09/28 SMI"
   #
   # Copyright (c) 2001 by Sun Microsystems, Inc.
   # All rights reserved.
   #

   # ike.preshared - Pre-shared secrets for IKE authentication.
   #
   # Entries are of the form:
   #
   # {
   #<attribute> <value>
   #...
   # }
   #
   # Consult the man page for ike.preshared(4) for details.
   {
             localidtype IP
             localid 172.25.124.140
             remoteidtype IP
             remoteid 172.25.123.64
             key 31313131313131313131313131313131
   }
   {
             localidtype IP
             localid 172.25.124.140
             remoteidtype IP
             remoteid 172.25.123.153
             key 31323334313233343132333431323334
   }
```

Loading SPD

If the file */etc/inet/ipsecinit.conf* exists, the Security Policy Database will be loaded automatically when the system is booted.

To load the database on a running system, use the command *ipsecconf*. Initially, call the command with the option *-f* and subsequently with the option *-a* :

- Flush Policies:

```
/usr/sbin/ipsecconf -f
```

- Loading SPD:

```
/usr/sbin/ipsecconf -a /etc/inet/ipsecinit.conf
```

To check the entries, use the command *ipsecconf* without any options.The system will display all policy entries:

Example

```
# /usr/sbin/ipsecconf -f
# /usr/sbin/ipsecconf
# /usr/sbin/ipsecconf -a /etc/inet/ipsecinit.conf
        WARNING : New policy entries that are being added may
        affect the existing connections. Existing connections
        that are not subject to policy constraints, may be
        subject to policy constraints because of the new
        policy. This can disrupt the communication of the
        existing connections.
# /usr/sbin/ipsecconf
#INDEX 74
{lport 23} ipsec {auth_algs md5 encr_algs des encr_auth_algs md5 sa shared}
```

IKE daemon – in.iked

If the file */etc/inet/ike/config* exists, the IKE daemon will start automatically when the system is booted.

If you want to activate the changes to the IKE policy file of a running system, stop the IKE daemon and restart it with */usr/lib/inet/in.iked*.

Example

```
# ps -ef |grep iked
    root 20866     1  0 08:44:15 ?        0:00 /usr/lib/inet/in.iked
    root 20868 27027  0 08:44:20 pts/4    0:00 grep iked
# kill -9 20866
# /usr/lib/inet/in.iked
```

8.7.2.2 Client configuration - (Windows 2000)

Configuring IPSec under Windows 2000 is divided into following steps:

- Configuring/generating IPSec policy:
  - Configuring/generating a security rule.
  - Configuring the IPSec authentication rule.
  - Configuring/generating an IPSec filter list.
  - Configuring/generating an IPSec filter action
- Defining the authentication and encryption algorithms used with IKE.
- Assigning new security policies.

Under Windows 2000, the security policies are configured using the *Local Security Policy* tool.

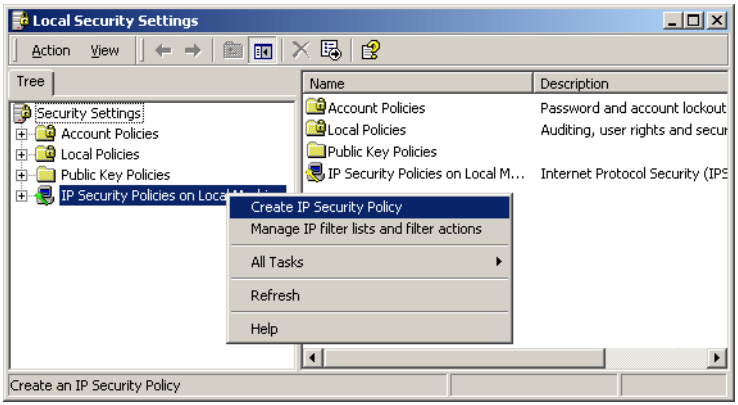
The individual steps are shown in the screenshots below.

Configuring/generating an IPSec policy

1. Setting a new IPSec policy
   - Open local security settings:

     Select the menu *Start* -> *Program files* -> *Administrative Tools* -> *Local Security Policy*.

► Create an IP security policy:

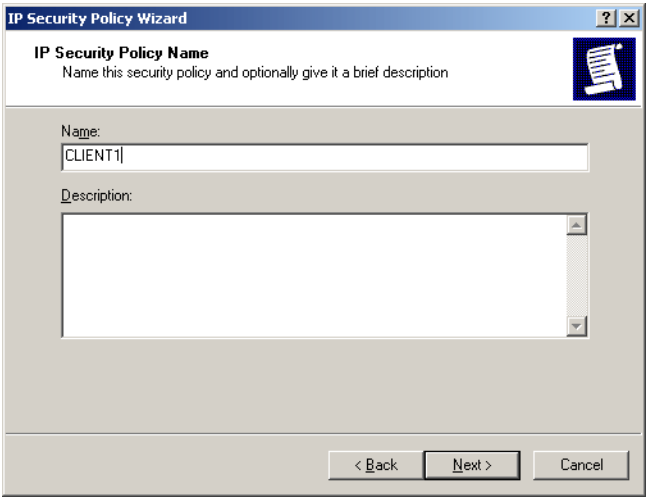


Use the right mouse button to click on *IP Security Policies on Local Machine* and select *Create IP Security Policy* from the context menu.

The IP security policies wizard is started.

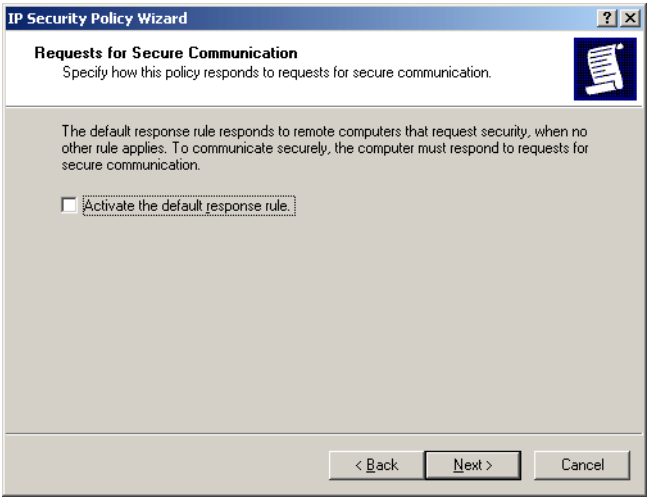
Click on the button *Next >*.

► Give the IP security policy a name:



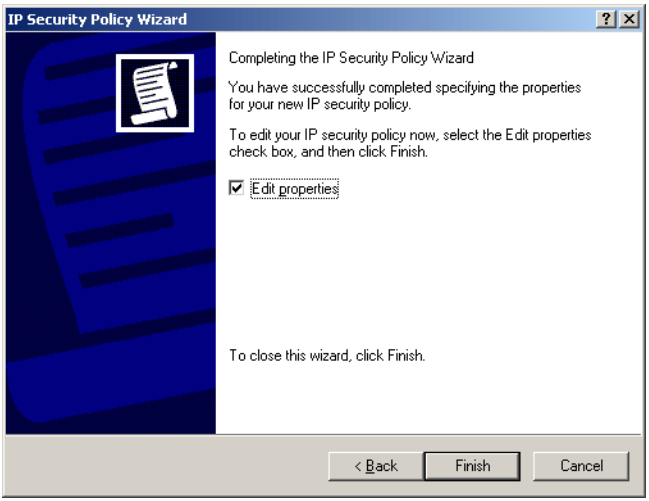
Enter the name in the field *Name* and click *Next >*.

► Deactivate the default response rule:



Deselect the option *Activate the default response rule* and click *Next >*.

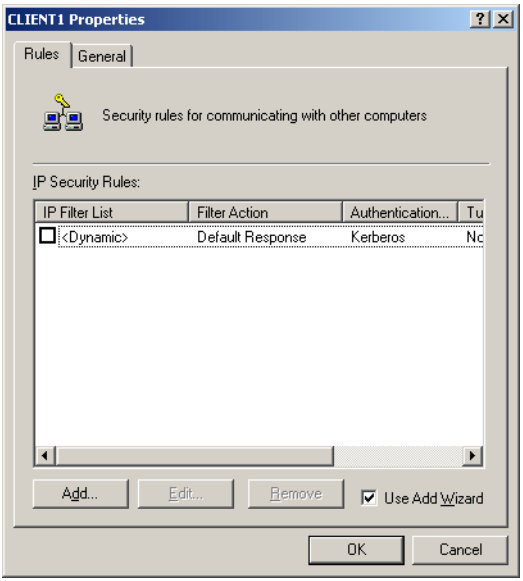
► Complete setting of the IP security policy:



Activate the option *Edit Properties* and click *Finish*.

2. Create a security rule for the new policy

   ► Create a security rule:

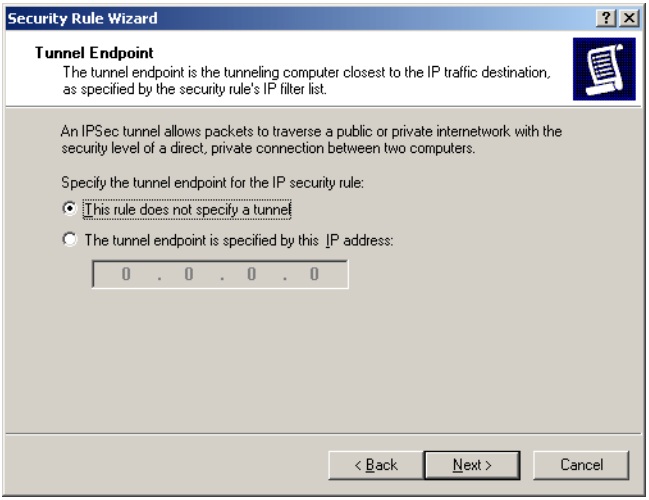


   In the dialog box activate the option Use Add Wizard then click *Add...*

   The Security Rules Wizard is opened.

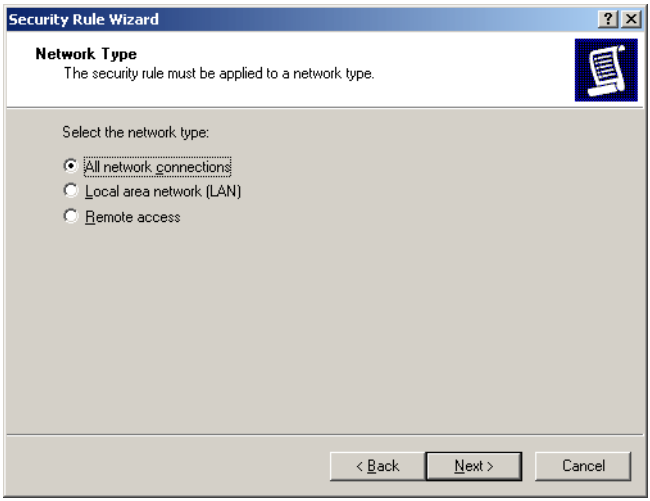
   Click *Next >*.

► Defining IPSec mode:



Select the option *This rule does not specify a tunnel* and click *Next* >.

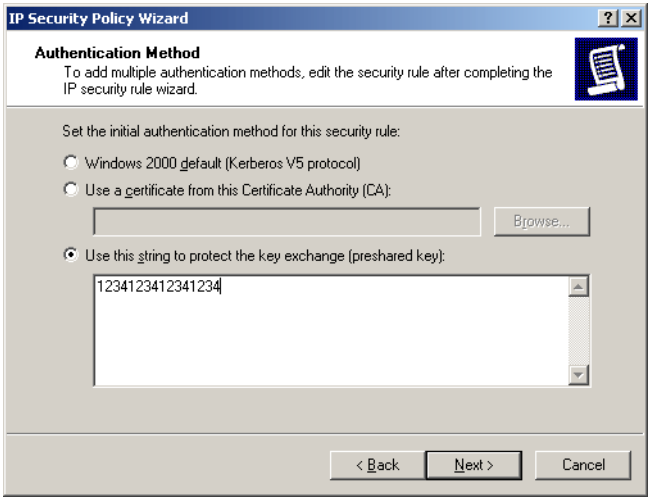
► Defining network type:



Select the option *All network connections* and click *Next >*.

3. Configure the IPSec authentication rule

– Define the authentication method preshared keys:



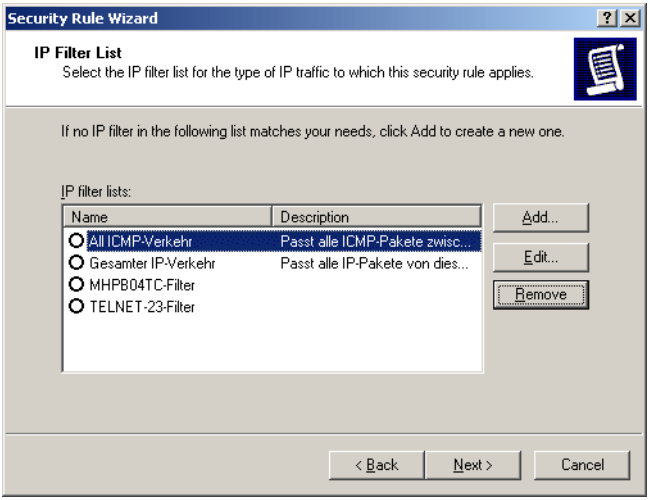
Select the option *Use this string to protect the key exchange...* and enter the preshared key (string, 16 characters).

Caution!

The same key must be entered on the administration server as a hexadecimal string.

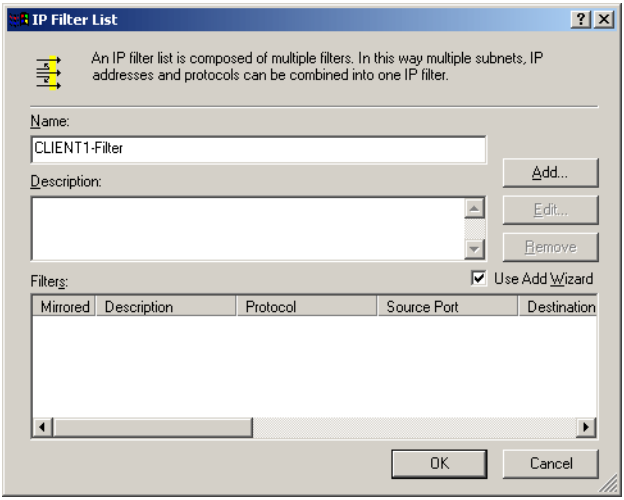
4. Configure the filter list

► Create a filter list for the policy rule:



Click *Add...*

► Define a name for the filter (list):

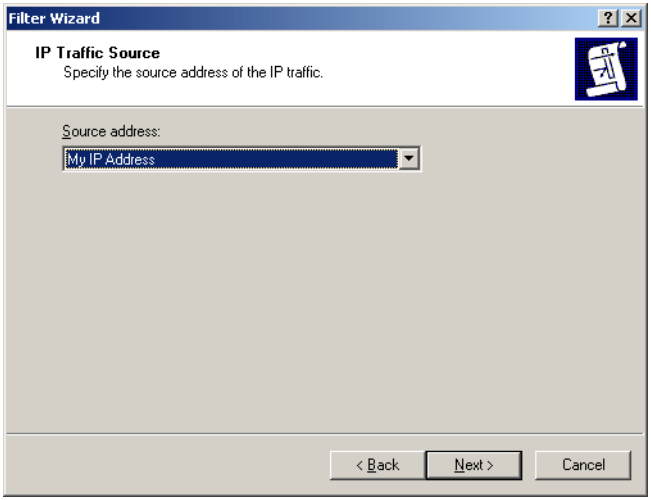


Enter the name in the field *Name* and activate the option to use the wizard and click *Add*.

The IP Filter Wizard is opened.

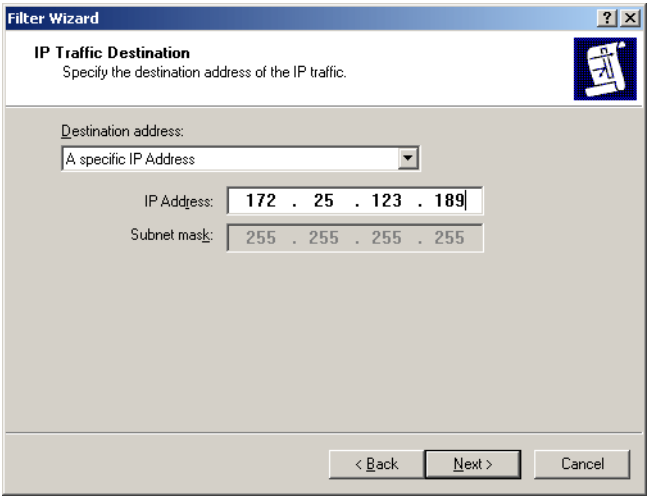
Click *Next >*.

► Define the IP traffic source:



In the list *Source address*, select the option *My IP Address* and click *Next >*.

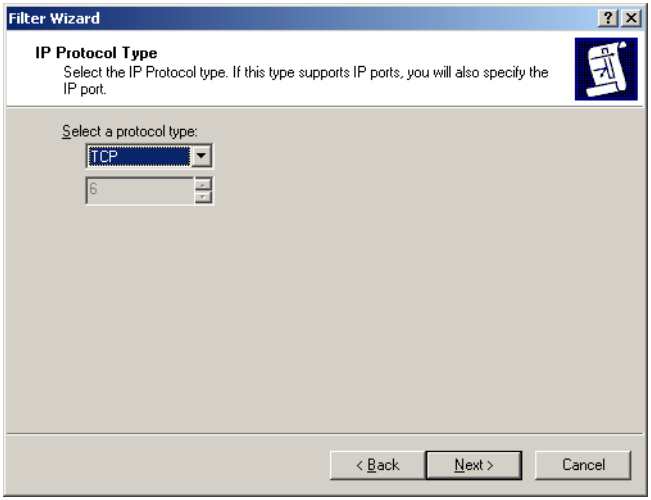
► Define the IP traffic destination:



From the list *Destination address,* select the option *A specific IP Address* and enter the address of the adimistration server in the field *IP address.*

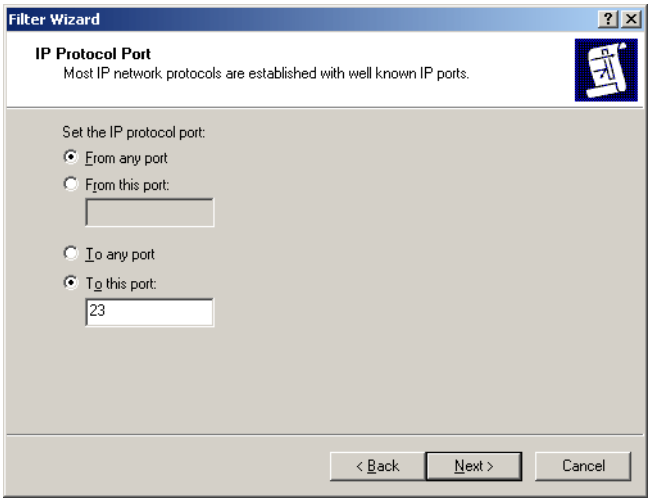
Click *Next >*.

► Define the IP protocol type:



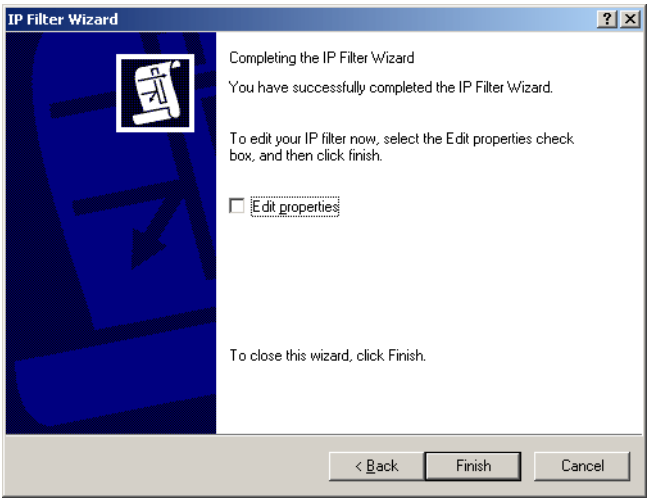
Select the protocol type *TCP* from the list and click *Next >*.

► Define the source and destination port:



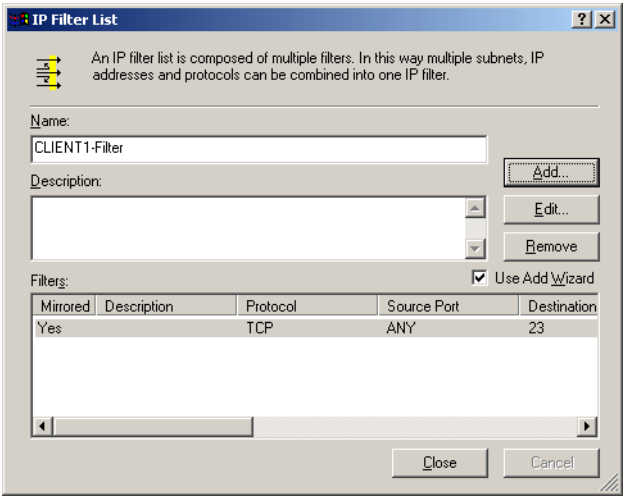
Select the option *From any port*, then select *To this port*, enter the port number 23 and click *Next >*.

► Complete the filter list:



Deselect the option *Edit properties* and click *Finish*.

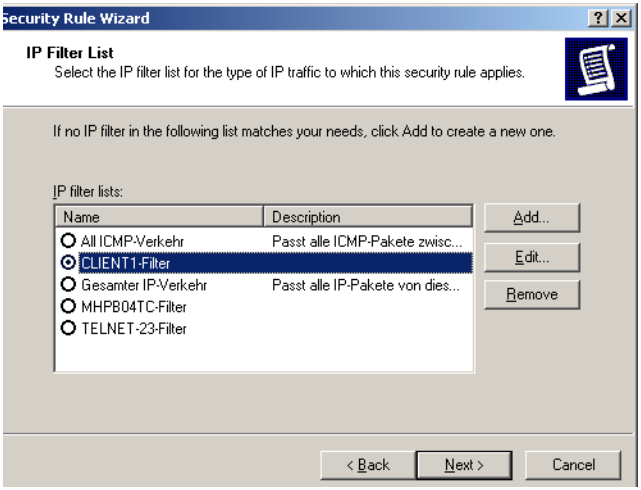
► Complete the IP filter list:



Click *Close*.

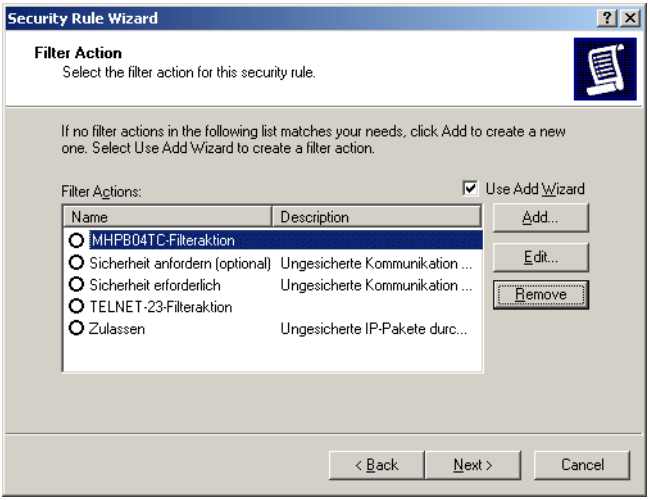
5. Configure/create the filter action

► Select the filter:



Select the newly created filter `CLIENT1-Filter` and click *Next >*.

► Define the filter action:

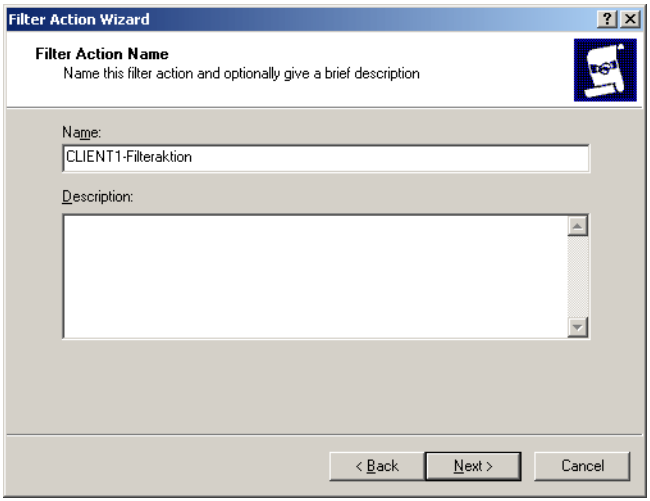


Activate the option *Use Add Wizard* and click the *Add...* button

The filter action wizard is opened.

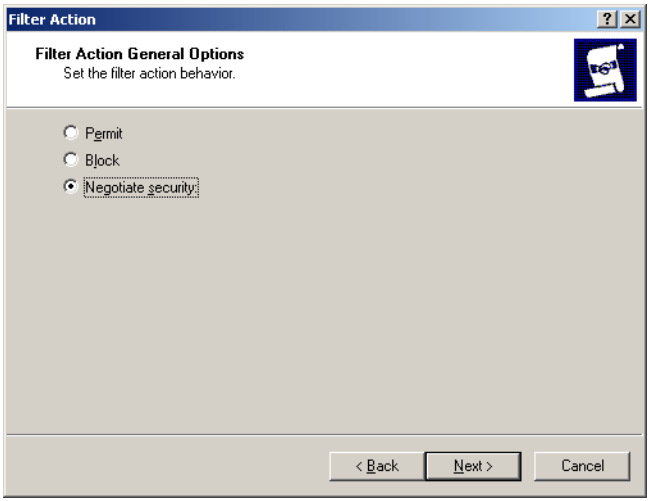
Click *Next >*.

► Define a name for the filter action:



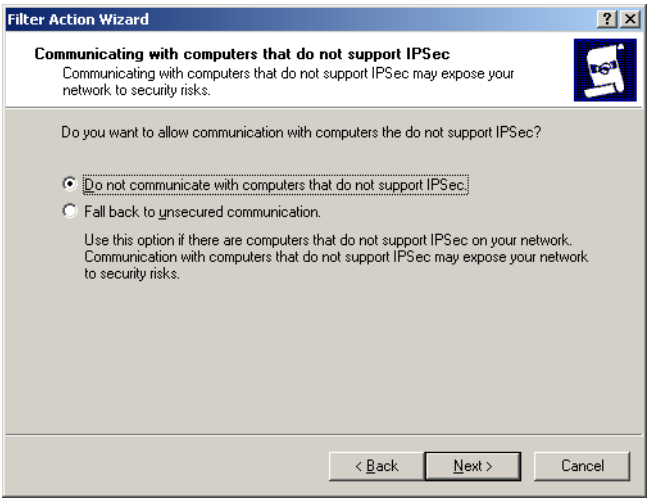
Enter the name in the field *Name* and click *Next >*.

► Define a filter action:



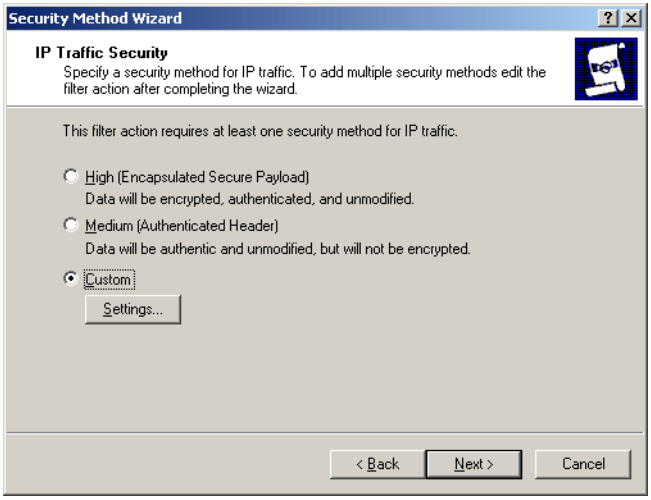
Select the option *Negotiate security* and click *Next >*.

► Define the communications for computers which do not support IPSec:



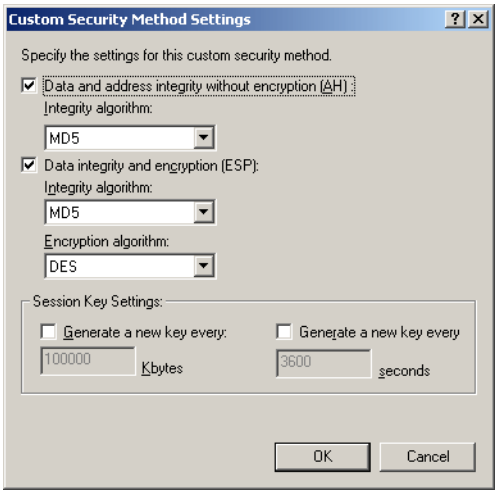
Select the option *Do not communicate with computers that do not support IPSec* and click *Next >*.

► Define the IP traffic security:



Select the option *Custom* and click *Settings...*

► Define the settings for the security method:



Select the following:

– in the list *Integrity algorithm (AH),* the option *MD5*
– in the list *Integrity algorithm (ESP),* the option *MD5*
– in the list *Encryption algorithm,* the option *DES*

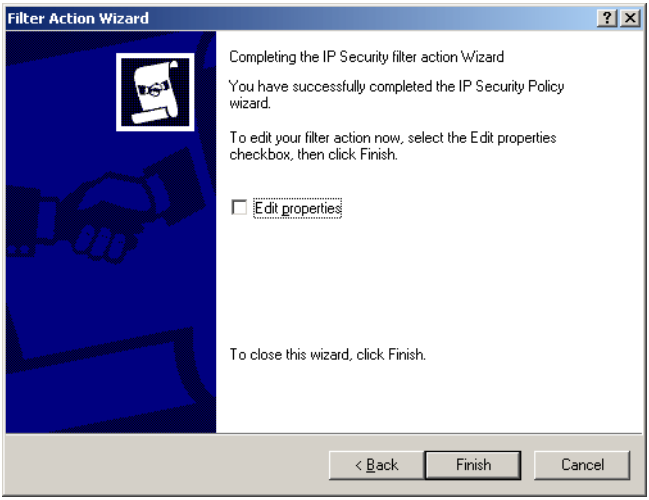
These values must conform with the configuration on the server.

Click *OK*.

The security method wizard is called again.

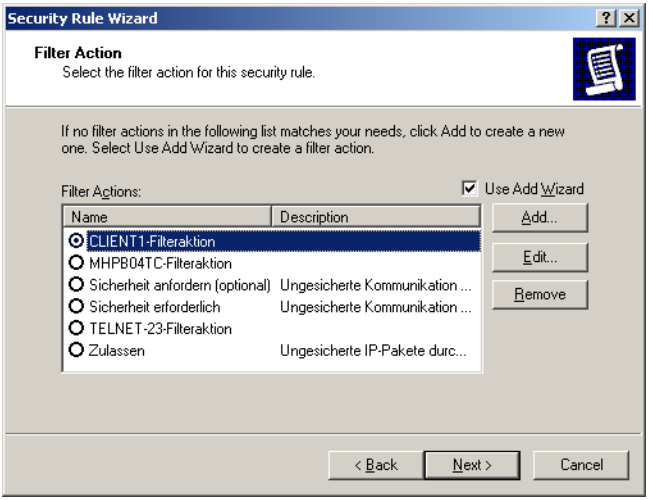
Click *Next >*.

► Complete filter action:



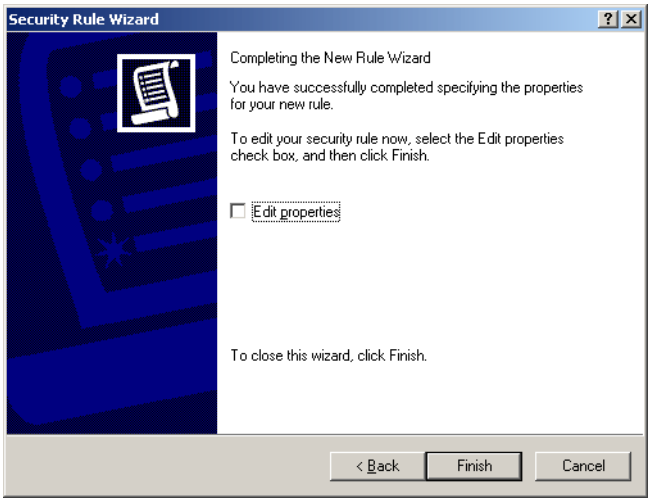
Deactivate the option *Edit Properties* and click *Finish*.

► Select the filter rule for the security rule:



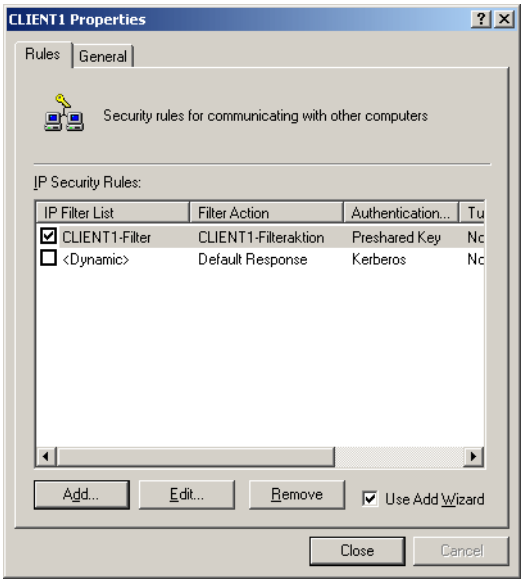
Select the newly created filter action `CLIENT1-Filte action` and click *Next >*.

► Complete the rule:



Deactivate the option *Edit Properties* and click *Finish*.

► Complete the security rule:



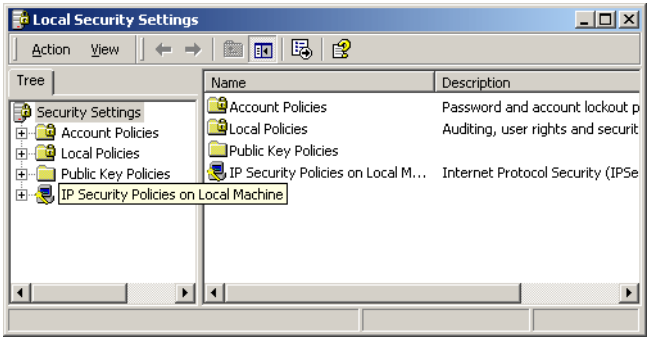
Click *Close*.

IKE settings

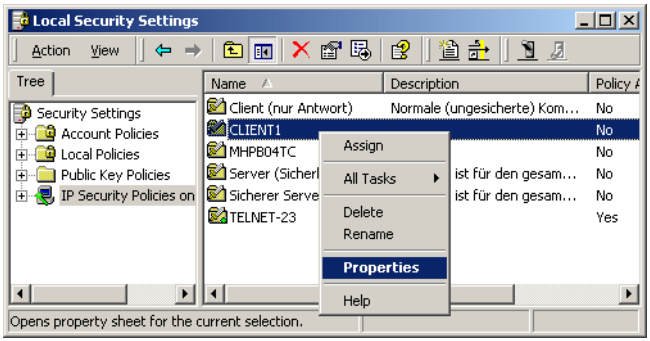
► Open local security settings:

Select menu *Start* -> *Program files* -> *Administrative Tools* -> *Local Security Policy*.

► Select IP Security Policies on Local Machine.

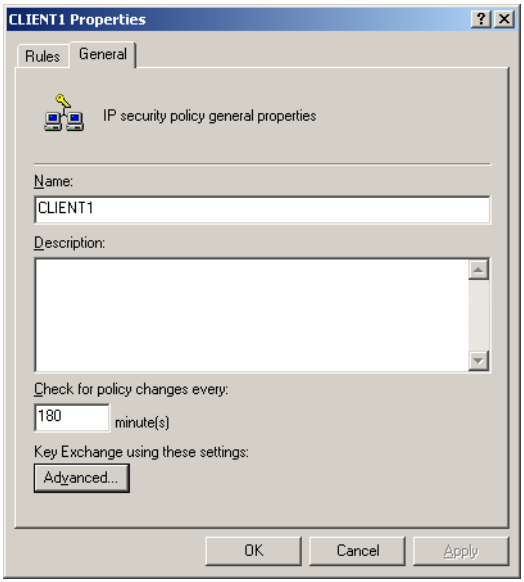


► Select the properties for security policy required:

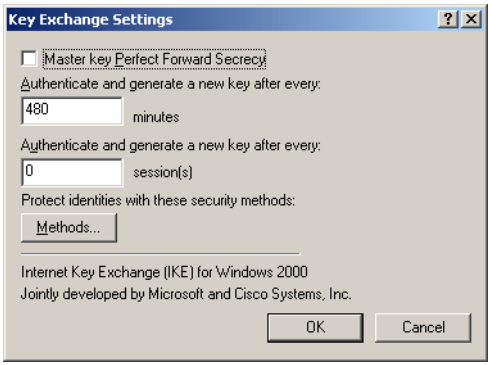


Right-click on the security policy `CLIENT1` and select the option *Properties* from the context menu.

► In the *Properties* window, select the tab *General* and click on the button *Advanced...*

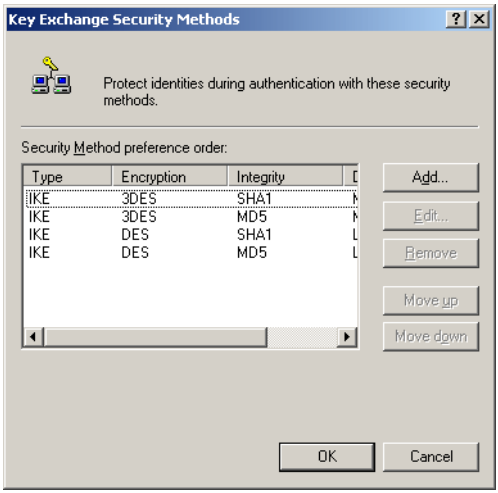


► In the window *Key Exchange Settings,* click on *Methods...*

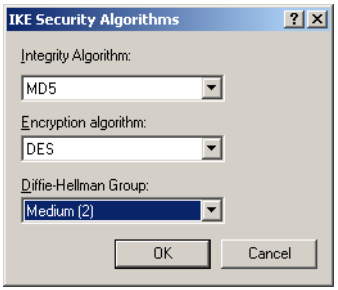


► In the window *Key Exchange Security Methods,* delete the default settings (Remove..) and click *Add...*

Choose the settings in the window *..Security Methods* which match those on the server.



► Add security methods:



In the *IKE Security Algorithms* window, select the following:

– in the list *Integrity algorithm,* the option *MD5*
– in the list *Encryption algorithm,* the option *DES*
– in the list *Diffie-Hellman Group,* the option *Medium (2)*

Click *OK*.

The Key Exchange Security Methods is displayed again.

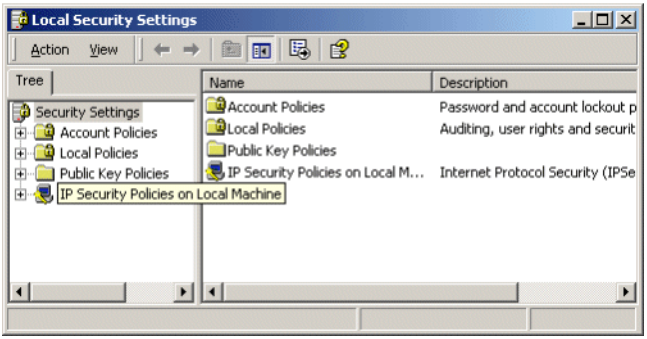
Click *OK*.

Assigning a new security policy

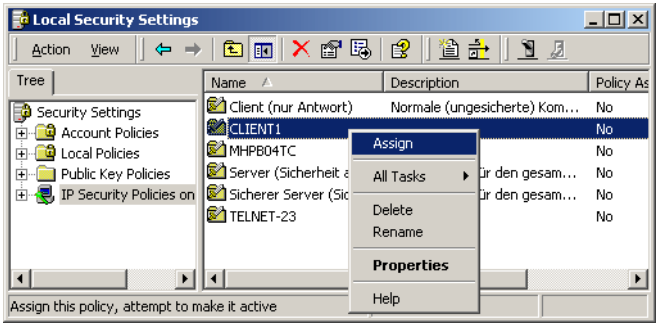
- Open local security settings:

  Select the menu *Start* -> *Program files* -> *Administrative Tools* -> *Local Security Policy*.

- In the window *Local Security Settings*, select the option IP Security Policies on Local Machine.



- Right-click on the policy `CLIENT1` and select the option *Assign* in the context menu.



8.8 Command interface

Web-based CMX administration makes the following commands available to you.

- **add_cmxadm**
  Add to ServerView configuration file
- **del_cmxadm**
  Delete an entry from the ServerView configuration file
- **manage_cert**
  Manage certificates on the administration server
- **set_port**
  Change the port number in configuration files
- **wca_init**
  Activate and deactivate *SMAWwca*
- **wca_stunnel**
  Start and stop *Stunnel*

The *manage_cert* command can be called on the administration server only, the other commands are available on the communication servers only.

The commands are described below in alphabetical order.

8.8.1 add_cmxadm - Add to ServerView configuration file

The *add_cmxadm* command lets you add all information needed by ServerView to start CMX administration to the ServerView configuration file. This information also causes a button for starting CMX administration to be created on the ServerView start site. ServerView must be rebooted so that the changes made take effect.

Syntax

/opt/SMAW/bin/add_cmxadm ␣applname␣{ pathname | **win** | **sol** }

applname
: Name of the application to be added. This name is shown below on the application button (maximum of 8 characters).

pathname

Full pathname of the Java Web Start application on the system on which *javaws* is started. You only need specify the pathname if it differs from the default pathname; otherwise you should specify the *win* or the *sol* option.

If the pathname for the Windows client contains special characters such as blanks, the string must be embedded in double quotes each followed by a single quote.

Example for JRE 1.4.2_04:
'" C:/Program Files/Java/j2re1.4.2_04/javaws/javaws.exe "'

Example for JRE 1.5.0_02:
'" c:/Program Files/Java/jre1.5.0_02/bin/javaws.exe "'

win

The default path for JRE 1.5.0_02 on Windows systems is used:
'" c:/Program Files/Java/jre1.5.0_02/bin/javaws.exe "'

sol

The path for JRE 1.5.0_02 on Solaris systems is used:
/opt/SMAW/SMAWj2rt/jre/solaris/jre1.5.0_02/javaws/javaws

8.8.2 del_cmxadm - Delete entry from ServerView configuration file

The *del_cmxadm* command deletes all information needed by ServerView to start CMX administration from the ServerView configuration file.

Syntax

/opt/SMAW/bin/del_cmxadm applname

applname

Name of the application to be deleted. This name is shown below on the application button (maximum of 8 characters).

8.8.3 manage_cert - Manage certificates

The *manage_cert* script makes a user-friendly interface to the *OpenSSL* certification programs available. It is limited to the arguments that are relevant in the context of CMX administration. *manage_cert* accesses the CMX administration-specific default values defined in the *OpenSSL* configuration files.

The script is called on the administration server.

Syntax

/opt/SMAW/SMAWcmx/bin/manage_cert
[␣**?** | ␣**-h** | ␣**-help**]
{ ␣**-newca** [␣**-days** number] |
␣**-newkey** |
␣**-newreq** [␣comm-server] |
␣**-sign** [␣comm-server] |
␣**-finish** [␣comm-server] |
␣**-verify** |
␣**-print**␣certificate }

? | **-h** | **-help**
Outputs the command syntax.

-newca [␣**-days** number]
Generates a new CA if one does not already exist. You are prompted for details on the CA when you press ENTER. You can accept the default values by pressing ENTER again. All relevant files are created in the */opt/SMAW/SMAWswca/PrivateCA* directory.

number specifies the period of validity of the certificate in days.
Default: 365

-newkey
Generates a new private key specifically for your server (*/opt/SMAW/SMAWswca/ssl/private/wca_privkey.pem*). Note that this private key is not encrypted because *Stunnel* is not able to prompt the user for the password for the key.

-newreq [␣comm-server]
Generates a certificate signing request (CSR). This is stored in */opt/SMAW/SMAWswca/ssl/req/comm-server.newreq.pem*.

comm-server is the name of the communication server for which the server certificate is to be created.
Default: *`uname -n`*

-sign [␣comm-server]
Calls the *OpenSSL* CA program to sign a certificate signing request. The program expects the request in */opt/SMAW/SMAWswca/ssl/req/comm-server.newreq.pem*.
The signed certificate is written to */opt/SMAW/SMAWswca/ssl/certs/comm-server.newcert.pem*.

comm-server is the name of the communication server for which the server certificate is to be created.
Default: *`uname -n`*

-finish [␣comm-server]
Generates the server certificate (*/opt/SMAW/SMAWswca/ssl/certs/comm-server.pem*) in the format required by *Stunnel*. *manage_cert* expects the signed certificate in */opt/SMAW/SMAWswca/ssl/certs/comm-server.newcert.pem*, the associated request in */opt/SMAW/SMAWswca/ssl/req/comm-server.newreq.pem* and the private key in */opt/SMAW/SMAWswca/ssl/private/wca_privkey.pem*.

comm-server is the name of the communication server for which the server certificate is to be created.
Default: *`uname -n`*

-verify
Verifies all certificates in the */opt/SMAW/SMAWswca/ssl/certs* directory.

-print␣certificate
Shows the content of the certificate named *certificate*.

End status

0 executed successfully

1 no arguments or incorrect arguments

2 output file already exists

3 input file missing

x denotes the *OpenSSL* return code

Files

/opt/SMAW/SMAWswca/conf/openssl_wca.cnf
OpenSSL configuration file with *Stunnel*-specific default values.

/opt/SMAW/SMAWswca/conf/openssl_ca.cnf
OpenSSL configuration file with CA-specific default values.

8.8.4 set_port - Change port number

The *set_port* command lets you change the port number in all configuration files that affect CMX administration. This is always necessary if the default port number 910 is already reserved by another application.

Syntax

/opt/SMAW/bin/setport ␣new-port [␣old-port]

new-port
New port number to be entered in the configuration files.

old-port
Previous port number
Default: 910.

Files

/opt/SMAW/SMAWcmx/wca/stunnel/conf/stunnel_wca.conf

/opt/SMAW/SMAWcmx/wca/telnet.cfg

/opt/SMAW/SMAWcmx/wca/telnet-windows.cfg

8.8.5 wca_init - Activate and deactivate SMAWwca

The *wca_init* command lets you activate and deactivate web-based CMX administration in an active boot environment. Relevant configuration files are generated during activation and deleted from the system during deactivation. If *Stunnel* is installed, it is started when *SMAWwca* is activated and stopped when *SMAWwca* is deactivated.

The *wca_init status* command lets you query the current status of the administration software.

Syntax

/opt/SMAW/bin/wca_init ␣{ **start** | **stop** | **status** }

start
Activates web-based CMX administration.

stop
Deactivates web-based CMX administration.

status
Outputs the current status.

Example

```
# wca_init status
SMAWwca started
stunnel(/opt/SMAW/SMAWcmx/wca/stunnel/conf/stunnel_wca.c
onf) is running
```

8.8.6 wca_stunnel - Start and stop Stunnel

The *wca_stunnel start* command lets you start the *Stunnel* daemon with a CMX administration-specific configuration file. In this case *Stunnel* acts as an SSL wrapper for the Telnet connection between administration client and communication server.

The server certificate is expected in *opt/SMAW/SMAWcmx/wca/stunnel/certs/comm-server.pem*. By default, *Stunnel* accepts connections via port 910.

The *wca_stunnel stop* command stops this daemon process.

Syntax

/opt/SMAW/SMAWcmx/bin/wca_stunnel ␣{ **start** | **stop** }

start
Starts the *Stunnel* daemon with a CMX administration-specific configuration file.

stop
Stops the *Stunnel* daemon for CMX administration.

End status

0 executed successfully

1 incorrect argument

2 *Stunnel* is already running

3 port 910 is reserved

4 *Stunnel* could not be started

5 specified certificate does not exist

6 *SMAWPstun* is not installed

7 *Stunnel* has not been started (*wca_stunnel stop*)

8 no authorization

9 kill wca_stunnel failed

File

/opt/SMAW/SMAWcmx/wca/stunnel/conf/stunnel_wca.conf
Stunnel configuration file

8.9 Solving problems

Various problems may arise when ServerView is started. Some are described below together with work-arounds/solutions.

1. Page was not found

   Display:

   ```
   This page cannot be displayed.

   The desired page is not available at the moment. There may
   be technical difficulties, or you should check your browser
   settings.
   ```

   Error recovery:

   - Use the *ps -ef | grep http* command to check whether the Apache server is running.
   - If the result is negative, start the Apache server with the following commands.

     1. cd /etc/rc2.d
     2. sh S97SIapache

2. JAVA Web Start download error

   Display in detail screen:

   ```
   An error occurred when starting/executing the application.
   Title: Serverview(MHPB04TC)
   Manufacturer: Fujitsu Siemens Computers
   Category: Download error

   Server returned incorrect MIME type when accessing resource:
   http://MHPB04TC:8881/wsa.jnlp - text/html
   ```

   Error recovery:

   Change the Web Start configuration as follows.

   - Choose *Start* -> *Programs* -> *Java Web Start* -> *File* -> *Settings*.
   - Select the option *None* for *Proxy Server*.

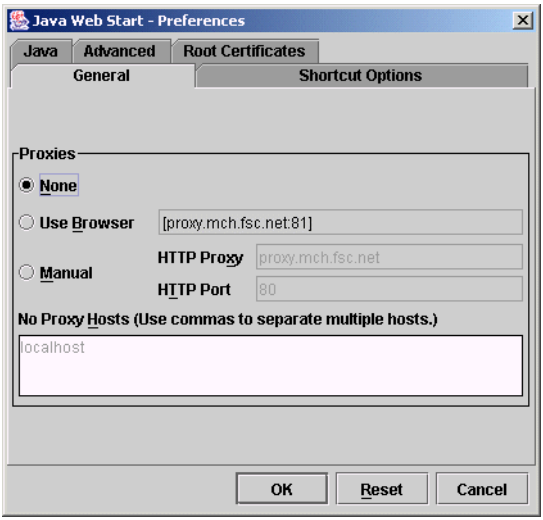


Figure 28: Java Web Start - Preferences

3. No reaction when ServerView is started (brief flickering)

   In this case Java Web Start is not started because incorrect folder options are set for jnlp files.

   Error recovery:

   - In Windows Explorer choose *Tools* -> *Folder Options* -> *File Options*.
   - *Opens with* must be followed by *javaws*.
     If necessary, click on the *Change...* button to change this entry.

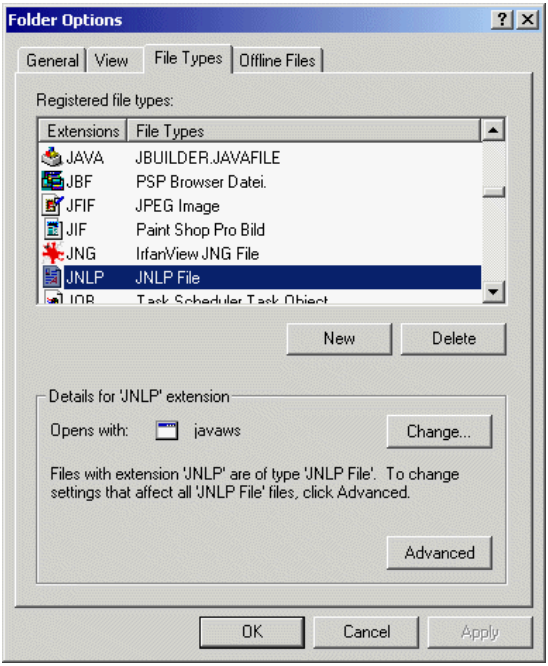


Figure 29: Folder options

4. No reaction when starting CMX administration by clicking on the CMXADM button

   In this case Java Web Start is not started. This may be due to an incorrect entry in the ServerView configuration file. Check the path to the Java Web Start application that you added to the configuration file */opt/SMAW/public_html/WSAConfig* using the *add_cmxadm* command.

5. Error starting CMX administration via SSL/TLS

   The two message boxes below may appear.

Figure 30: Error message when starting CMXADM (1)

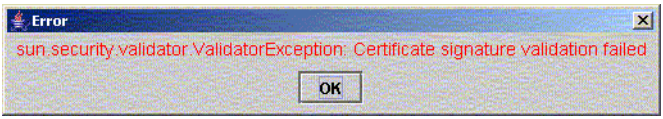


Figure 31: Error message when starting CMXADM (2)

In case (1), *Stunnel* was not started. Start *Stunnel* using *wca_stunnel start*.

In case (2), you have possibly changed your certificates and/or keys without restarting *Stunnel*. Stop *Stunnel* with *wca_stunnel stop* and restart with *wca_stunnel start*.

9 Configuring connections via RFC1006

The Internet *Request for Comments* RFC1006 was published by the Internet Architecture Board (IAB) to define a TCP/IP-based OSI transport service in accordance with ISO 8072.

The Transmission Control Protocol (TCP) is connection-oriented, i.e. a logical connection is set up before the actual data transfer. TCP ensures that data reaches the application in the correct order and that no data is lost or corrupted.

With the transport service based on ISO 8072, messages are separated from each other by end markers; the data flow is message-oriented. There are no such end markers with TCP, as a continuous data stream is transferred. A TCP application can only recognize from the file contents where one logical message ends and the next begins.

Another difference between TCP/IP and the ISO transport service can be seen during connection setup: TCP guarantees that all previously transmitted data reaches the receiver (orderly release). In accordance with ISO 8073, the responsibility for ensuring that connections are not closed until the data has been exchanged (abortive release) lies with the applications themselves and not with the transport system.

RFC1006 is implemented in CMX as the Transport Service Provider (TSP). The services of the RFC1006 TSP are offered via the Open Group standard Transport Provider Interface (TPI) from which the programming interfaces ICMX, XTI and TLI are mapped internally.

The RFC1006 TSP is started automatically with CMX. You can set tuning parameters and view the status and statistics.

9.1 Overview of configuration data

The following diagram provides an overview of the configuration data relevant to connection setup via RFC1006.

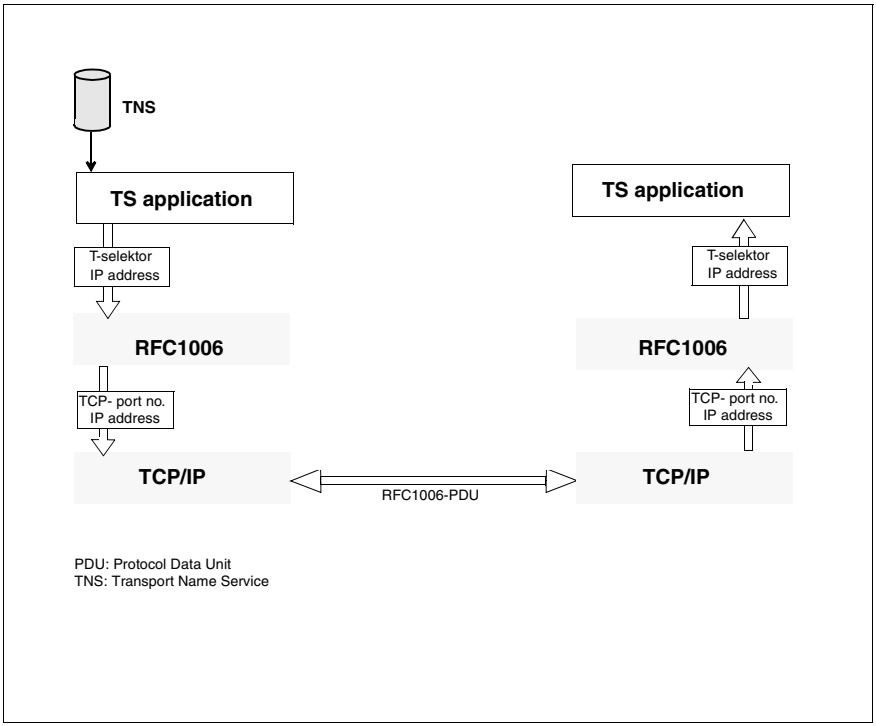


Figure 32: Connection setup via RFC1006 - overview

A transport connection via the RFC1006 TSP is assigned uniquely to a TCP connection. To establish this TCP connection, you need to know the IP address and port number. Each IP address/port number pair is referred to as a TCP address. A TCP address uniquely identifies a TCP connection endpoint. A TCP connection is thus uniquely identified by a pair of TCP addresses, i.e. those of the local and remote endpoints. A TS application attaches to the RFC1006 TSP using a transport selector, or T-selector for short. This T-selector is then used on the RFC1006 TSP protocol layer in order to address the TS application.

TCP signals incoming TCP connection requests to the RFC1006 TSP by means of a TCP listening stream, or TCP listener for short. When the RFC1006 TSP is started, it automatically creates a TCP listener with the TCP address '*.iso-tsap', where '*' stands for the IP address 0.0.0.0 which represents the entire collection of local network access points, and 'iso-tsap' stands for port number 102 which is reserved for the ISO transport service as per RFC1006. This means that all incoming TCP connection requests that arrive at any local network access point for port number 102 will be displayed to the RFC1006 TSP. In addition, TS applications can use special addresses and options to force the RFC10006 TSP to create further TCP listeners. These can then be used to forward connection requests which arrive at a specific local network access point or for a port number other than 102.

In the case of outgoing TCP connection requests, the exact TCP address of the remote endpoint is always specified, usually with port number 102. The local endpoint, however, is left open by specifying the TCP address '*.*' (all address bits set to 0). TCP then selects a free port number for the local endpoint, while IP selects a suitable network access point. As before, the TS application that initiated the connection request can use special addresses and options to force the RFC1006 TSP to use a port number other than 102 for the remote endpoint and to use a specific IP address for the local endpoint. The port number of the local endpoint, however, is left undefined. When establishing a connection actively, the TS application normally signs on with a T-selector only, and merely specifies an IP address and a T-selector for the partner address.

The addresses needed to establish the connection are taken from the TNS database.

9.1.1 Configuration data for local TS applications

To set up a transport connection via the RFC1006 TSP, you generally define a transport system application on your local system (referred to below as the TS application) and another transport system application for each partner application you want to reach (referred to below as remote TS applications).

To configure a local TS application in the TNS, you must normally specify the following data:

- A GLOBAL NAME with which the application can be identified in the TNS. For information on the structure and features of the GLOBAL NAME, see section “GLOBAL NAME” on page 77.

- The address format that identifies the TSP via which the connection is to be established. The RFC1006 TSP is identified by the format *RFC1006* or *LANINET*.
- In the case of the LANINET address format, a port number for which the RFC1006 TSP will create a TCP listener for passive connection setup if necessary.

  Example:

  ```
  TSEL LANINET A'1100'
  ```

  When the TS application attaches to the TSP for passive connection setup, the RFC1006 TSP will create a TCP listener with port number 1100.

  This specification is only required if a partner system does **not** address the RFC1006 standard TCP port number 102 as the endpoint of the TCP connection. This is the case, for example, for RFC1006 implementations in CMX versions prior to CMX V5.0 and in *open*FT applications that run on systems from other manufacturers.
- In the case of the RFC1006 address format:
  - the T-selector used by remote TS applications to address the local TS application which has attached to the TSP for passive connection setup and
  - the T-selector transferred to the partner system in the *calling TSAP-ID* parameter as address information in the RFC1006 protocol.

  For example, *open*FT applications use the T-selector *T'$FJAM'* for passive connection setup and the T-selectors *T'$FJAM001'*, *T'$FJAM002'*, etc. for active connection setup.

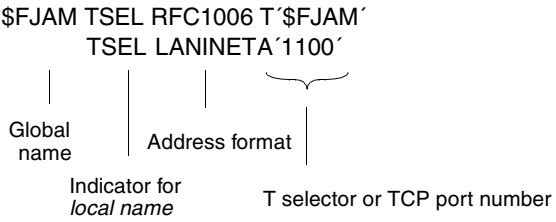
The LANINET and RFC1006 address formats can be specified on their own or both together.

If only the LANINET address format is specified, the TS application is a pure LANINET application. When attaching to the TSP for passive connection setup, it uses its own TCP listener, i.e. no other TS application can attempt passive connection setup with this port number. If a TCP listener with this port number already exists, the LANINET application will fail to attach to the TSP. In the case of active connection setup with a pure LANINET application, the specified port number will be transferred to the partner system in the *calling TSAP-ID* parameter as address information in the RFC1006 protocol.

If both address formats are specified, it is possible for several TS applications to share the same TCP listener and the associated port number. However, the T-selector specified with the RFC1006 address format must be unique and must be exclusive to one particular TS application which has attached to the TSP for passive connection setup. TS applications attached with the RFC1006 address format only and those attached for both address formats can receive connection requests via the TCP listener with the address '*.iso-tsap'.

With local TS applications, T-selectors, port numbers, and the corresponding address format specifications are contained in the LOCAL NAME identified by the TSEL indicator before each address format specification. The LOCAL NAME is assigned to a GLOBAL NAME.

Sample entry for a local TS application



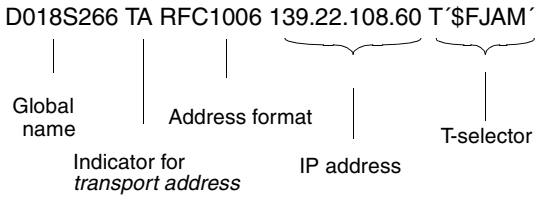
9.1.2 Configuration data for remote TS applications

To configure a remote TS application in the TNS, you must specify the following data:

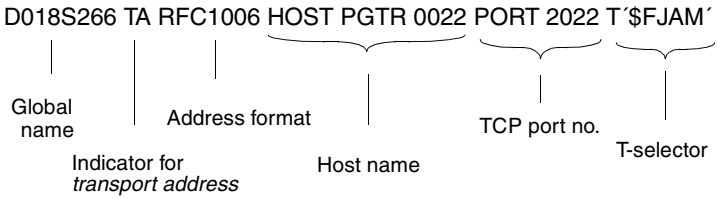
- The GLOBAL NAME with which the application can be identified in the TNS. For information on the structure and features of the GLOBAL NAME, see section “GLOBAL NAME” on page 77.
- The IP address of the remote system (or alternatively, the host name).
- The address format that identifies the Transport Service Provider via which the remote system is to be reached. The TSP RFC1006 is identified by the format *RFC1006* or *LANINET*.
- In the case of the RFC1006 address format:
  - the T-selector used to identify the TS application in the remote system and transferred to the partner system in the *called TSAP-ID* parameter as address information via the RFC1006 protocol.
  - a port number that identifies the remote endpoint of the TCP connection (optional). If this is not specified, port number 102 will be addressed.
- In the case of the LANINET address format, the port number used to identify the remote endpoint of the TCP connection and transferred in the *called TSAP-ID* parameter as address information in the RFC1006 protocol.

For remote TS applications, the IP address, the format, the TCP port number (if present), and the T-selector are contained in the TRANSPORT ADDRESS identified by the indicator TA. The TRANSPORT ADDRESS is assigned to a GLOBAL NAME.

Sample entry for a remote TS application



In the example above, the application is addressed via the standard port number 102.



9.2 Establishing a connection to a remote partner system

This section describes the procedures involved in connection setup and configuration with the help of an example. The two systems involved use CMX V5.0 or later. In a separate section, the procedures involved are described from the point of view of both the active system and the passive system.

9.2.1 Active connection setup

The CMX application begins by identifying its LOCAL NAME in the TNS and using this to attach to CMX for active connection setup. The CMX application then reads the TRANSPORT ADDRESS (IP address or host name, T-selector) of the partner application from the TNS database. If the TNS database contains the host name, this will be translated into the associated IP address dynamically when the address is requested.

The CMX application passes on this information to the RFC1006 TSP, which then sets up a TCP connection to the RFC1006 TCP in the partner system using the IP address and the standard port number 102. The RFC1006 TSP uses this connection to send the T-selector of the local CMX application (in the *calling TSAP-ID* parameter) and the T-selector of the remote CMX application (in the *called TSAP-ID* parameter) to the remote RFC1006 TSP in the form of an RFC1006 protocol element (CR TPDU). The rest of the procedure is described in the following section from the point of view of the passive system.

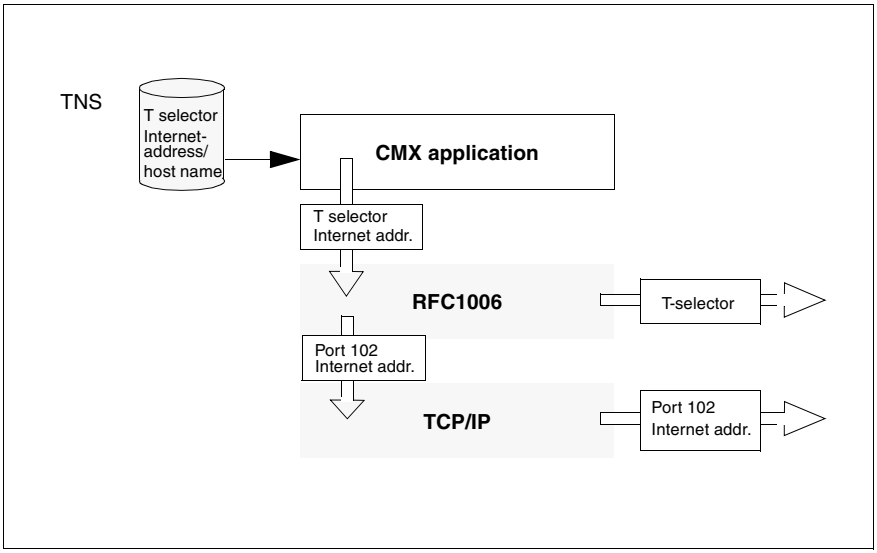


Figure 33: Active connection setup between systems with CMX V5

Supposing that the remote system with CMX V5 has the host name *PGRTV009* and the IP address *76.3.13.11*, one of the following entries would be necessary for the remote application:

```
rloginrem TA RFC1006 HOST PGRTV009 T'rlogin'
or
rloginrem TA RFC1006 76.3.13.11 T'rlogin'
```

9.2.2 Passive connection setup

The CMX application begins by identifying its LOCAL NAME in the TNS and using this to attach to CMX for passive connection setup. Irrespective of this, the RFC1006 TSP automatically creates the TCP listener '*.iso-tsap' and will be "listening" for any incoming TCP connection request at any local network access point for port number 102.

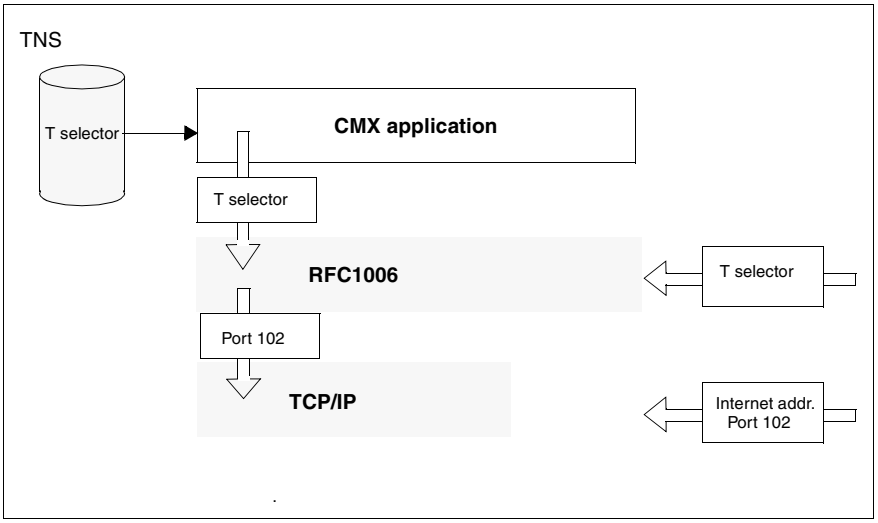


Figure 34: Passive connection setup to partner systems with CMX V5

At the initiative of the partner application, the RFC1006 TSP receives a connection request via the TCP listener with the source address (IP address and port number) of the remote TCP connection endpoint and the destination address '*.iso-tsap'. Once this TCP connection has been established, the RFC1006 TSP receives an RFC1006 protocol element (CR TPDU) containing the T-selector of the calling CMX application in the *calling TSAP-ID* parameter, and the T-selector of the called CMX application (i.e. the attached local CMX application) in the *called TSAP-ID* parameter. A connection request is then sent to this CMX application, together with the IP address of the remote TCP connection endpoint and the T-selector of the called CMX application in the form of the TRANSPORT ADDRESS. The local CMX application can use this TRANSPORT ADDRESS to determine the GLOBAL NAME of the partner application in the TNS.

Your CMX V5 system must be configured as follows in the TNS (local application, incoming connection setup from the CMX V5 partner application):

```
rloginlocal TSEL RFC1006 T´rlogin´
```

9.3 Querying the status/statistics of the RFC1006 TSP (rfc1006stat)

Use the following command to query the status and statistics of the RFC1006 TSP:

rfc1006stat␣[**-d**]␣[**-s**]

rfc1006stat␣[**-a**]␣[**-c**]␣[**-d**]␣[**-l**]␣[**-n**]

rfc1006stat␣**-z**

rfc1006stat returns the contents of various data structures of the RFC1006 TSP which provides the ISO 8072 transport service via TCP/IP as defined in RFC1006.

The command has three possible modes (see above):

- In the first mode – without the *-a*, *-c*, and *-z* options – it returns the following information:
  - TSP status
  - maximum and current number of transport connections
  - maximum and current number of TSAPs attached for passive connection setup (referred to below as listener TSAPs)
  - maximum TIDU length
  - preferred TPDU length
  - maximum TIDU length for TCP
  - the most important statistics for transport service users

  The *-s* option outputs additional information on special settings, counters and statistics relating to the pool of free TCP streams, and stream resource bottlenecks.

  The *-d* option outputs additional internal statistics for debugging purposes.
- In the second mode – *-a* and/or *-c* option – it returns detailed information on the TCEPs and/or listener TSAPs currently available. The output format is described in the section “TSAP/TCEP output format” on page 218.

The *-d* option outputs all opened RFC1006 streams and their associated TCP streams, irrespective of their TPI status. As in the first mode, the output begins with basic information on the TSP status, the permitted and maximum number of TSAPs and TCEPs etc., but does not include any statistics.

- In the third mode, it resets all statistics counters to zero without restarting the RFC1006 TSP.

Options

-a access points

Outputs detailed information on listener TSAPs and TCEPs that are not in the TPI status TS_DATA_XFER. As long as the RFC1006 TSP is operational, you will have an active listener TSAP bound to the control stream (UPPER = CONTROL).

-c connection end points

Outputs detailed information on TCEPs in the TPI status TS_DATA_XFER.

-d internal details

Outputs additional information on internal statistics for debugging purposes.

-l long line format

Joins the two lines of the TSAP/TCEP output format together.

-n number

Outputs IP addresses and port numbers in numeric format instead of as symbolic names.

-s special

Outputs special additional information such as settings, counters and statistics relating to the requesting, release, and availability of TCP steams, and stream resource bottlenecks.

-z zero

Resets all statistics counters to zero.

TSAP/TCEP output format

TSAP and TCEP details are output in exactly the same format, consisting of 13 fields arranged over two lines (provided the *-l* option is not specified).

Line 1 refers to the upper TPI between the ISO 8072 transport service user and the RFC1006 TSP.
Line 2 refers to the lower TPI between the RFC1006 TSP as the transport service user and TCP as the transport service provider.

Fields in line 1:

UPPER
: Upper stream ID, i.e. number identifying the stream between the TPI adapter and the RFC1006 TSP.

  The control stream */dev/SMAWcmx/rfc1006lm* is assigned the number 0 and the symbolic name CONTROL.
  The administration stream */dev/SMAWcmx/rfc1006adm* is assigned the number 1 and the symbolic name ADMIN.
  All other streams (clones of */dev/SMAWcmx/rfc1006*) are assigned numbers between 2 and *r6-max-conns-and-saps*. *r6-max-conns-and-saps* is defined in the file */usr/kernel/drv/SMAWr6.conf*.

TP0-STATE
: TPI status in relation to the ISO 8072 transport service user.

CIND(MAX)
: Current and maximum number of outstanding connection indications. In the case of listener streams, this is always set to “-”.

TREF
: Transport reference of the CMX application.

PV
: RFC1006 protocol variant:
  ’3.0’ in the case of the CMX address format LANINET
  ’-’ in the case of the CMX address format RFC1006

TPDUSZ
: Maximum TPDU length for this connection.

OWN-TSEL
: Coding scheme:
'A:' ASCII, 'E:' EBCDIC, 'T:' TRANSDATA, 'V:' void, 'X:' hexadecimal
and value of the T-selector of the application attached locally with the RFC1006 address format.

PARTNER-TSEL
: Coding scheme:
'A:' ASCII, 'E:' EBCDIC, 'T:' TRANSDATA, 'V:' void, 'X:' hexadecimal
and value of the T-selector of the partner application.

Fields in line 2:

LOWER
: Lower stream ID, i.e. number identifying the stream between the RFC1006 TSP and TCP.

TCP-STATE
: TPI status in relation to TCP as the transport service provider.

CIND(MAX)
: Current and maximum number of outstanding connection indications. In the case of listener streams, this is always set to "-".

OWN-TCP-ADDR
: TCP address in the format *host.port* or **.port*, where * stands for the IPv4 address 0.0.0.0 or the IPv6 adress ::, which represents the entire collection of local network access points.

PARTNER-TCP-ADDR
: TCP address in the format *host.port*, where *host* is the host name without the default domain. If the *-n* option is specified, *host* is replaced by the IP address in the canonical IPv4 or IPv6 representation format.

Special features in the output format for listener TSAPs

CMX applications attached with T_PASSIVE and the RFC1006 address format share the TCP listener '*.iso-tsap' or, if T_OPTA7 has been specified with the host name *host*, the TCP listener '*host*.iso-tsap'. The same applies for CMX applications that are also attached with the LANINET address format and the same port number *port* (i.e. the LOCAL NAME contains both `TSEL RFC1006` *T-selector* and `TSEL LANINET` *port*). In this case, the TCP listener '**.port*' or '*host.port*' may also be shared.

All shared TCP listeners appear only once in the listener TSAP output:

- '*.iso-tsap' is always assigned to the control stream (UPPER = CONTROL).
- '**.port*' and '*host.port*' are assigned to any suitable listener TSAP.
- '*host*.iso-tsap' is also assigned to a suitable listener TSAP, but is displayed in a separate two-line entry, where the UPPER field is preset with '*' (asterisk).

Listener TSAPs attached both for 'iso-tsap' (port number 102) and for another port number (i.e. the LOCAL NAME contains both `TSEL RFC1006` *T-selector* and `TSEL LANINET` *portnumber*) appear with the other port number in the OWN-TCP-ADDR field. Nevertheless, the attachment also applies for 'iso-tsap'.

In contrast, pure LANINET listeners (i.e. the LOCAL NAME contains `TSEL LANINET` *portnumber* only) are never attached for 'iso-tsap' and do not share their TCP listener with other listener TSAPs. They are identified by '`3.0`' in the PV field.

See also

rfc1006tune

9.4 Setting operating parameters for the RFC1006 TSP (rfc1006tune)

Use the *rfc1006tune* command to set the operating parameters for the RFC1006 TSP.

Caution!

Tuning measures should be restricted to experienced system administrators who have an in-depth knowledge of UNIX, networks, and transmission protocols.
If they are carried out incorrectly or left incomplete, this will impair system behavior and may even render the system inoperable.

rfc1006tune [␣**-a**␣maxsaps] [␣**-c**␣maxconns] [␣**-e**␣secs] [␣**-h**␣high] [␣**-l**␣low] [␣**-p**␣port] [␣**-r**␣response] [␣**-t**␣tidusize] [␣**-u**␣unrelated]

rfc1006tune␣**-d**

rfc1006tune allows you to tune various settings of the RFC1006 TSP.

Parameter values that deviate from the default values are automatically saved to the configuration file */opt/SMAW/SMAWcmx/lib/rfc1006/rfc1006.conf*.
This configuration file is then read out by the daemon */opt/SMAW/SMAWcmx/etc/rfc1006d* each time the RFC1006 TSP is started.

The command has two possible modes:

- In the first mode, it sets the specified parameters to the specified values. Parameters can be set to their default value by means of a blank string. Any parameters not specified retain their current values.
- In the second mode, it resets all parameters to their default values.

Each successful *rfc1006tune* command returns the following information:

– the parameter values currently in use
– parameter values that will come into effect following the next restart
– default parameter values
– value ranges

If you simply require information on current parameter values, use the first command mode without any options.

Options

-a maxsaps

Sets the permitted number of simultaneous listener TSAPs to the value specified in *maxsaps*.
The *maxsaps* parameter must be set to a decimal number or a blank string, in which case the default value applies. The associated keyword in the configuration file is *MAXSAPS*.

Note that a listener TSAP will be used implicitly by the control stream (see entry containing UPPER = CONTROL in the output for the command *rfc1006stat -a*).

1 <= maxsaps <= r6-max-conns-and-saps - maxconns
where *r6-max-conns-and-saps* is defined in the file */usr/kernel/drv/SMAWr6.conf*.

The default value for *maxsaps* is *(r6-max-conns-and-saps +1) / 2*.

-c maxconns

Sets the maximum permitted number of simultaneously open transport connections to the value specified in *maxconns*.
The *maxconns* parameter must be set to a decimal value or a blank string, in which case the default value applies. The associated keyword in the configuration file is *MAXCONNS*.

0 <= maxconns <= r6-max-conns-and-saps - maxsaps
where *r6-max-conns-and-saps* is defined in the file */usr/kernel/drv/SMAWr6.conf*.

The default value for *maxconns* is *r6-max-conns-and-saps / 2*.

-d default

Resets all parameters to their default values.

-e secs

Sets the time period (i.e. the number of seconds between two alarm ticks) to the value specified in *secs*.

With every alarm tick the RFC 1006 TSP decreases and checks

- the remaining time for timers with respect to unrelated TCP connections
  (unrelated TCP connections are passively established connections where no CR TPDU has been received yet, see option *-u*.)
- the timer with respect to responses sent to outgoing connection requests (waiting on CC TPDU, see option *-r*)

The *secs* parameter must be set to a decimal value or a blank string, in which case the default value applies. The new value takes effect immediately. The associated keyword in the configuration file is *PERIOD*.

6 <= secs <= 50000

The default value for *secs* is 30.

-h high

Sets the maximum number of free TCP streams in the pool to the value specified in *high*.

The *high* parameter must be set to a decimal value or a blank string, in which case the default value applies. The associated keyword in the configuration file is *HIGHPOOL*.

low <= high <= r6-max-conns-and-saps /4
where *r6-max-conns-and-saps* is defined in the file */usr/kernel/drv/SMAWr6.conf*.

The default value for *high* is 64.

-l low

Sets the minimum number of free TCP streams in the pool to the value specified in *low*.

The *low* parameter must be set to a decimal number or a blank string, in which case the default value applies. The associated keyword in the configuration file is *LOWPOOL*.

0 <= low <= min(high, r6-max-conns-and-saps / 4)
where *r6-max-conns-and-saps* is defined in the file */usr/kernel/drv/SMAWr6.conf*.

The default value for *low* is 32.

-p port

Sets the port number for the TCP listener assigned to the control stream to the value specified in *port*.

The *port* parameter must be set to a decimal number or a blank string, in which case the default value applies. The new port number comes into effect the next time the RFC1006 TSP is restarted. The associated keyword in the configuration file is *LISTEN-PORT*.

1 <= port <= 32767

The default value for *port* is 102.

-r response

Sets the maximum waiting time for responses to outgoing connection requests (CR TPDU) to the value specified in *response*. *response* represents the number of alarm ticks, see option *-e*. If, within the specified period, no confirmation (CC TPDU) or disconnection request (DR TPDU) arrives from the partner system, the connection is terminated by the RFC1006 TSP. The local CMX application receives disconnection notice including cause for disconnection and T_RLNORESP.

The *response* parameter must be set to a decimal number or a blank string, in which case the default value applies. The new value for such connection requests takes effect only for those connection requests sent after the time of change. The associated keyword in the configuration file is *RESPONSEPRD*.

2 <= response <= 50000

The default value for *response* is 6.

-t tidusize

Sets the maximum TIDU length to the value specified in *tidusize*.

The *tidusize* parameter must be set to a decimal value or a blank string, in which case the default value applies. The new TIDU length comes into effect the next time the RFC1006 TSP is restarted, and is the value returned to CMX applications by the *t_info()* function. The maximum TIDU length set here affects the TPDU length negotiated by the RFC1006 protocol. The associated keyword in the configuration file is *TIDUSIZE*.

4089 <= port <= 65273

The default value for *tidusize* is 4089.

-u unrelated

Sets the maximum life time of unrelatad TCP connections to the value specified in *unrelated*. *unrelated* is the number of alarm ticks, see option *-e*. If no RFC 1006 connection request (CR TPDU) is received within the specified period, a TCP connection initiated by the partner system is terminated by the RFC 1006 TSP. The *unrelated* parameter must be set to a decimal number or a blank string, in which case the default value applies. This new value takes effect only for passive connections established after the time of change.

The associated keyword in the configuration file is *UNRELTCPPRD*.

2 <= unrelated <= 50000

The default value for *unrelated* is 2.

Files

Optional configuration file
/opt/SMAW/SMAWcmx/lib/rfc1006/rfc1006.conf

Template for the configuration file
/opt/SMAW/SMAWcmx/lib/rf1006/rfc1006.template

See also

rfc1006stat

10 Administration and maintenance

This chapter describes the maintenance and administration functions, as they can be activated via the command line interface. The CMX commands are listed in alphabetical order.

Man pages

The CMX product is accompanied by *man pages*. These deal with both the commands described in this chapter and additional commands which in CMX V5.1 are used only by experts. All man pages are in English.

To display the man page for a specific command, enter:

`man` *command*

The following list provides an overview of the available commands, arranged according to component.

10.1 Overview of commands

Commands marked with * are only described in the man pages.

CMX-specific commands

cmxconf
: output system-local configuration information on a GLOBAL NAME of a CMX application (providing this is possible for all levels of the transport system). For further information see section “Checking the configuration of a CMX application (cmxconf)” on page 231.

cmxdec
: decode CMX messages, see section “Decoding CMX messages (cmxdec)” on page 233.

cmxdiag
: prepare diagnostic documents, see section “Collection and preparation of diagnostic information (cmxdiag)” on page 237.

cmxinfo
: information on CMX configuration, see section “Information on CMX configuration (cmxinfo)” on page 239.

cmxm
: control CMX monitor, see section “CMX monitor (cmxm)” on page 266.

cmxmd
: control CMX monitor daemon, see section “CMX monitor daemon (cmxmd)” on page 279.

cmxprod
: query installed communication products, see section “Querying installed communication products (cmxprod)” on page 281.

cmxstat
: output currently reserved TSP-specific resources (applications, connections, subnet connections and internal statistics), see section “TSP-specific status information (cmxstat)” on page 283.

cmxtrc
: switch traces on/off in a transport system, i.e. in protocol layers 1 to 4. By specifying a GLOBAL NAME, you can restrict traces to specific CMX applications. For further information see section “Traces for the transport system (cmxtrc)” on page 287.

cmxtune
: modify limit values for CMX automatons, see section “Changing limits for the CMX automaton (cmxtune)” on page 290.

StartStop
: start and stop CMX and CCPs, see section “Starting and stopping CMX and TSPs (StartStop)” on page 302.

TSP-specific commands

rfc1006
: start and stop the RFC1006 TSP, see section “Starting and stopping CMX and TSPs (StartStop)” on page 302.

rfc1006stat
: query the status and statistics of the RFC1006 TSP, see section “Querying the status/statistics of the RFC1006 TSP (rfc1006stat)” on page 216.

rfc1006tune
: set operating parameters for the RFC1006 TSP, see section “Setting operating parameters for the RFC1006 TSP (rfc1006tune)” on page 221.

ntp
: start and stop TSP NTP, see section “Starting and stopping CMX and TSPs (StartStop)” on page 302.

tp02
: start and stop TSP TP0/2, see section “Starting and stopping CMX and TSPs (StartStop)” on page 302.

nea
: start and stop TSP NEA, see section “Starting and stopping CMX and TSPs (StartStop)” on page 302.

The last two commands can only be used if the appropriate product is installed on your system.

TNS-specific commands

tnsxchk
: check status of a TS directory, see section “Checking the TS directory (tnsxchk)” on page 305.

tnsxcom
: add, modify, and delete TS directory entries, see section “TS directory: create, update, output (tnsxcom)” on page 307.

tnsxd*
: start TNS daemon.

tnsxdel
: delete entries from a TS directory, see section “Deleting TNS entries (tnsxdel)” on page 311.

tnsxinfo
: display information on a TS directory, see section “Displaying information on the TS directory (tnsxinfo)” on page 314.

tnsxlock
: lock and unlock access to TNS daemons, see section “Locking access to the TNS daemon (tnsxlock)” on page 320.

tnsxprop
: display entries of a TS directory, see section “Outputting properties of TS applications in a TS directory (tnsxprop)” on page 321.

Trace commands

cmxl
: control and edit CMX library trace, see section “Controlling and editing the CMX library trace (cmxl)” on page 255.

comtr
: monitor traces for CMX-dependent drivers, see section “Traces for CMX drivers (comtr)” on page 291.

neal
: control and edit NEA library trace, see section „Controlling and editing NEABX library trace (neal)“ on page 298.

tnsxt
: save and edit TNS trace information, see section “Starting and stopping TNS trace (tnsxt)” on page 324.

The individual commands are described below in alphabetical order.

10.2 Checking the configuration of a CMX application (cmxconf)

The *cmxconf* command checks the consistency of the system-local configuration for a GLOBAL NAME and also the readiness for operation of all components involved. As far as possible, the configuration is viewed in its entirely in the transport system, i.e. from the transport layer to the line connection.

cmxconf can be specified in two formats: for a local application (option *-o*), or for a partner application (option *-p*).

Syntax

cmxconf ␣**-o**␣own-appl [␣**-t**␣ts-prov]

cmxconf ␣**-p**␣part-appl

-o␣own-appl
: (own) The local name parts of the application *own-appl* are checked, i.e. output indicates whether the local TSAPs are attached to the appropriate transport providers. *own-appl* is the GLOBAL NAME of the local application. The check can be restricted to a specific transport provider using *-t ts-prov*.

-t␣ts-prov
: If an application has several LOCAL NAMES, the check is restricted to one transport system provider (*ts-prov*).

  Possible values for the transport provider:

```
nea
ntp
tp02
rfc1006
```

-p␣part-appl
: (peer) The entire local configuration (TNS and FSS) is output for the specified partner address with the GLOBAL NAME *part-appl*. In the case of WAN transport providers, an additional check is made to determine whether the components of the transport system via which a connection is to be set up are active. This check covers both the transport address and the route information. If the partner address is of the type RFC1006, the only check made is whether the transport system is active.

Example

The following command lists the configuration information for the partner address *nearem*:

```
cmxconf -p nearem

Configuration information for the remote partner: nearem
================================================================================
NEA-NSAP:       71/255  Remote T-Sel:   A'nearemot'

Configured routes for this remote NSAP:
SNPAROUTES  name=nearoute short-id=1 subnet=X25-1 type=X25 dte-addr=123 SNPAROUTES
name=nearoute1 short-id=2 subnet=X25-2 type=X25 dte-addr=123

Active local SNPA's for this partner:
CC  IF#  STATE TYPE  Bit/s LINK    LINKS NETW. SUBNET SUBNET-ID SUBNET-ADDR
W3    1  NETC  X.21    64k -         0/1 -     LEASED X25_1     1
W3    2  NETC  X.21    64k -         0/1 -     LEASED X25_2     2
```

The *cmxconf* command therefore returns information on the accessibility of a local or remote application from the local viewpoint. It can be used to check the consistency of the local configuration of all possible communication partners of a TNS directory.

– This command is TNS-dependent and does not function for other directories. If the customer does not use a directory, one option is to simulate the entry in TNS in the event of communication faults.

– A route to a partner address need not necessarily be defined. If no route is defined in SNPA format, the transport providers (NTP and TP02) choose a subnet connection that supports the address format.

10.3 Decoding CMX messages (cmxdec)

The *cmxdec* command enables decoding of ICMX and XTI messages. These are messages of the following types:

– Error messages defined in the include files <*cmx.h*>, <*neabx.h*> and <*tnsx.h*> of the ICMX or in the include file <*xti.h*> of XTI. These messages are generated in the program interfaces ICMX(L), ICMX(NEA) and XTI.

– Messages resulting from faulty ICMX or XTI system calls, i.e. error messages defined in the <*errno.h*> file.

– Messages containing reasons transferred by the CMX automaton or CCP for a disconnection by either of these.

ICMX and XTI error messages, system messages, and reasons for disconnection are output in the form of a numeric code, decimal number, or hexadecimal number with leading “0x”. An error message generated at the program interface to the TNS can only be interpreted correctly if a value is given for the error type, the error class, and the error value. An error code is therefore output in the form of three decimal numbers. The values in this code may be negative.

An error code or the code for a disconnection (reason) is decoded by *cmxdec* if you transfer the message type (XTI error message, ICMX(L) error message, etc.) and the specified error code to *cmxdec*. The symbolic value defined in the corresponding include file is then output by *cmxdec* to standard error output.

Depending on the caller’s environment (LANG variable), *cmxdec* returns explanatory texts for messages. The texts are language-dependent and optional. German and English versions of the message catalogs are always supplied with CMX.

Syntax

cmxdec [␣**-c**] [␣**-d**] [␣**-n**] [␣**-s**] [␣**-t**] [␣**-x**] ␣code ...

The options identify the type of message specified in *code*. The default value is *-c*. *cmxdec* recognizes the options described below. The options are mutually exclusive.

-c

The value specified in *code* is an ICMX error message, as returned by *t_error()* at the ICMX(L) interface.

Output format: ␣

```
ICMX(L) error decoding (5.1)
CODE 0x%x = %d (TYPE %d CLASS %d VALUE %d)
      Symbolic value and explanation for TYPE
      Symbolic value and explanation for CLASS
      Symbolic value and explanation for VALUE
```

If VALUE indicates a system error message, the numeric value is usually entered in the last line instead of a symbolic value.

-d

The value specified in *code* is a reason for disconnection, as specified by *t_disin()* (ICMX(L)), *x_disin()* (ICMX(NEA)) or *t_rcvdis()* (XTI). Please note that for XTI/Internet (XTI via TCP/IP), the reason for disconnection is decoded as a system error, defined in <*errno.h*>.

Output format:

```
CMX reason decoding (5.1)
REASON 0x%x = %d:
ICMX(L):
      Symbolic value and explanation
XTI/Internet:
      Symbolic value and explanation
XTI/ISO:
      Symbolic value and explanation
```

-n

The value specified in *code* is an NEABX error message, as returned by *x_error()* at the ICMX(NEA) interface.

Output format:

```
ICMX(NEA) error decoding (5.1)
CODE 0x%x = %d (TYPE %d CLASS %d VALUE %d)
      Symbolic value and explanation for TYPE
      Symbolic value and explanation for CLASS
      Symbolic value and explanation for VALUE
```

-s

The value specified in *code* is a system error message as returned by system calls.

Output format:

```
CMX system error decoding (5.1)
CODE 0x%x = %d
      Explanation in English
```

-t

The three numeric values specified in *code* are a TNS error message, as returned by the TNS calls in the standard header.

Output format:

```
ICMX(TNS) error decoding
TYPE %x=%d CLASS =x%x=%d VALUE 0x%x=%d)
      Symbolic value and explanation for TYPE
      Symbolic value and explanation for CLASS
      Symbolic value and explanation for VALUE
```

-x

The value specified in *code* is an XTI error message, as returned by *t_error()*.

Output format:

```
XTI error decoding
CODE 0x%x = %d
      Symbolic value and explanation
```

code

For *code*, specify the numeric error code returned by ICMX, NEABX, TNS, or XTI, the code of a system message or the code for a reason for disconnection, which *cmxdec* is to decode. With options *-c, -d, -n, -s,* and *-x*, you must specify a decimal number or a hexadecimal number with leading “0x” or “0X” for *code*. With option *-t*, you must specify three signed decimal numbers or hexadecimal numbers with leading “0x” or “0X” for *code*.

A preset numerical value has different meanings for ICMX(L) and XTI.

TYPE, CLASS and VALUE correspond to the classifications of the error codes used in ICMX, NEABX and TNS; REASON is the reason for a disconnection. The numeric specification is followed by the symbolic definition according to the include files <*cmx.h*>, <*neabx.h*>, <*xti.h*> and <*tnsx.h*>. The explanatory text for this message attaches itself to this definition.

If an invalid or undefined value is specified for *code*, *cmxdec* decodes this value to the extent possible in terms of the symbolic representation of the reason for disconnection or the type, class and value of the error message. As explanatory text for the value of *code*, *cmxdec* outputs the message *Cannot decode*.

10.4 Collection and preparation of diagnostic information (cmxdiag)

The command *cmxdiag* collects diagnostic information and stores this in the file */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar* for fault analysis later. The diagnostic documents comprise configuration files, status information, log files (system, CMX) and traces. The range of the diagnostic documents is controlled by command parameters.

Syntax

cmxdiag ␣[**all** | **cctraces** | **konfig** | **ktraces** | **log** | **status** | **traces** | **cmxsnap**]

The command syntax is displayed if a command is entered without parameters.

all
: reads out all the information relevant to fault diagnostics and copies this data into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*.

cctraces
: copies all existing trace lists of the communication controllers (see manuals “CMX/CCP, ISDN Communication“ [3] and „CMX/CCP, WAN Communication“ [4]) into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*.

cmxsnap
: reads out the process table and copies this data into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*.

konfig
: reads out all configuration files relevant to fault diagnostics and copies this data into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*.

ktraces
: prepares traces for CMX drivers and copies this information into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*.

log
: This has the same function as *konfig*. Also copies the cron logfiles and logfiles of the CMX components into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar.*

status

This has the same function as *log*. Also detects and copies the current status of the communication controller, the status of the configured interfaces and information for the TSPs and the CMX configuration into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*

traces

This has the same function as *ktraces* and *cctraces*. Also copies the logfile of the cronjob and logfiles of the CMX components into the diagnostic packet */opt/SMAW/SMAWcmx/lib/ccp/diagfiles/collect/diag.tar*.

10.5 Information on CMX configuration (cmxinfo)

You can use the *cmxinfo* command to query information on the CMX configuration, and on the type and number of CCs served and the TSP access points. The command provides information on the possible and actual load on CMX and on the CCs/TSP access points. *cmxinfo* outputs the information to *stdout*.

The following diagram provides an example of the implementation of TSPs and subnetwork profiles. See also the information on the CMX architecture in the chapter “Architecture of Solaris communication” on page 11.

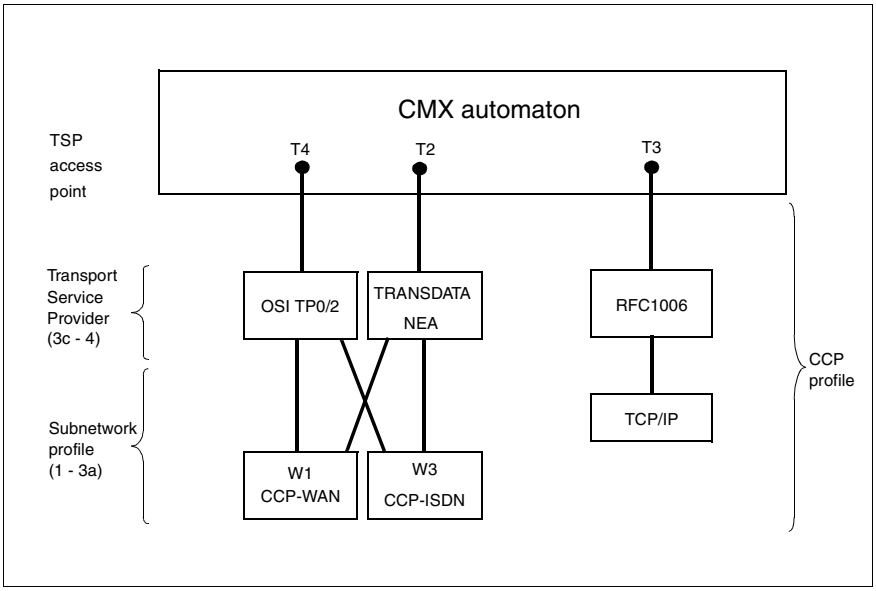


Figure 35: TSPs, subnetwork profiles, and CCP profiles

The following information is output by *cmxinfo*:

– the limits of the CMX automaton specifying how many TS applications (ICMX and XTI applications), transport connections and CCs can be served at the same time

– the number of CCs/TSP access points currently served

– the type of CCs/TSP access points served

– information on the CCs/TSP access points served, e.g. storage capacity of the CC, current status (operational or not)

– properties of the operational TSPs, e.g. the length of the data unit (TIDU), the number of transport connections that can be sustained simultaneously via the TSP access point

– information on all currently existing TSAPs or further information on a specific TSAP

The options can be used to limit the output to certain values.

Syntax

cmxinfo [␣**-t**]
[␣**-a**]
[␣**-b**␣{ id | **all** } [␣**-l**]]
[␣**-c**␣id]
[␣**-s**]
[␣**-C**␣{ id | **all** }]
[␣**-S**␣{ id | **all** }]
[␣**-i**] [␣**-M**] [␣**-v**]

If you call *cmxinfo* without arguments, the limits of the CMX automaton and the type and number of the CCs and TSP access points served are output.

-t

Output of execution trace information. Specifying *-t* is appropriate only in cases of error.

-a

Only the limits for CMX are output.

-b␣id | **all**

Only hardware information on the CC/TSP access point specified in *id* is output, assuming the CC/TSP access point is present in your computer. The individual CCs/TSP access points are specified as described with the *-c* option. Specifying *all* provides you with hardware information on all CCs/TSP access points currently available.

-l

Output of information on the WAN interface of the controller (only in conjunction with *-b id*).

-c␣id

Only information on the CC/TSP access point specified in *id* is output, assuming it is present in your computer. The individual CCs/TSP access points are specified as follows in *id*:

W[1-32]

For CC-WAN (X.21, V.24, V.35, ISDN). TSP access point which exists on a programmable CC. See section “Architecture of CCP profiles” on page 23.

T[1-6]

For a TSP access point (see section “Architecture of CCP profiles” on page 23).

-s

Output of a list of all currently available TSAPs together with the respective PIDs, the number of TCEPs (and the TEP status if the application uses the XTI interface).

-S␣id | **all**

Output of information for a currently available TSAP with the identification *id*. You can display available TSAPs using the *cmxinfo -s* command. The input format for *id* is x.y (e.g. 12.0) or hexadecimal beginning with 0x or 0X. If you specify *all*, the information for all currently available TSAPs is displayed.

-**C**␣id | **all**

Output of information for a currently available TCEP with the identification *id*. You can display available TCEPs using the *cmxinfo -S* command. The input format for *id* is x.y (e.g. 12.0) or hexadecimal beginning with 0x or 0X. If you specify *all*, the information for all currently available TCEPs is displayed.

-i

Output of information on LAN interfaces via which CMX applications can communicate.

-M

With *-b* or *-C* options, information is output in a shorter format: header and footer lines will be suppressed.

-v

Output of the thread Id (only in connection with the option *-s* or *-S*).

Output format of cmxinfo

The output of *cmxinfo* always begins with the header line. It contains the following entry:

```
CMX INFORMATION (6.0)
```

The extent of the information that follows depends on the components on which you have queried information, i.e. on the options you specified when you entered the *cmxinfo* command.

Below is a description of the output of *cmxinfo* regarding the individual components (CMX automaton, CCs/TSP access points).

Information on the CMX automaton (option -a)

You are given information on the limits of the CMX automaton and its current load.The output is in the following format:

```
CMX AUTOMAT VERSION G-5.1 x CCs/TSP access points
TEP a (g) ATT a (g) TSAP a (g) TCEP a (g) TSP a (g)
```

The specified values are explained below. The limits (g) are in parentheses, the current values (a) without parentheses.

CMX AUTOMATON VERSION

The version of the CMX automaton is entered here. The CMX automaton is the central component of CMX in the operating system kernel. It is the link between the user level (TS applications, ICMX and XTI library) and the TSPs (see section “Performance range of CMX and CCPs” on page 11).

x CCs/TSP access points

x is the number of CCs/TSP access points served by the CMX automaton.

TEP

Value *a* gives the number of currently supported transport endpoints (TEPs, see “X/Open Transport Interface” User Guide [2]). The number of TEPs is the sum of all XTI transport endpoints plus the number of all processes or threads attached via ICMX. More than one TS application process or thread can be attached to CMX.

The limit *g* specifies the maximum possible number of TEPs.

ATT

Value *a* specifies the number of currently existing attachments to the CMX automaton.

a is the sum of attachments of processes via all active TS applications. The value may be different from TEP, as a process can attach for more than one TS application.

An attachment can take place via the interface ICMX or XTI.

Value *g* states how many attachments can exist at the same time.

TSAP

Value *a* states the number of TS applications currently attached to the CMX automaton.

The TS application is assigned a transport service access point (TSAP) via the LOCAL NAME. All the processes that attach to the CMX automaton for this TS application are linked to this TSAP. Thus *a* is the number of currently existing TSAPs.

Value *g* states the maximum number of TS applications (ICMX and XTI applications) that can be attached to CMX at the same time.

TCEP

Value *a* states how many transport connections exist (TCEP =Transport Connection EndPoint).

Value *g* states the maximum number of transport connections that can be sustained via the CMX automaton at the same time.

TSP

Value *a* specifies the number of transport service providers (TSPs) served by the CMX automaton. Value *g* specifies the maximum number of TSPs that CMX can serve.

Information on the CCs/TSP access points (option -b all)

The information on the CCs/TSP access points is presented in the form of a table. The meanings of the output values is explained in the example below.

```
ID   TYPE         VERS  STATE    I1 I2  IO_ADDR    MEM  HW/FW_VERS CAB/BUS/SLOT
=============================================================================
T2 TPI-NEA       G-6.0 READY    - -      -         -      -        -/-/-
T3 TPI-RFC1006   G-6.0 READY    - -      -         -      -        -/-/-
T4 TPI-TPO/2     G-6.0 READY    - -      -         -       -       -/-/-
T5 TPI-NULL-TP   G-6.0 READY    - -      -          -      -       -/-/-
T6 TPI-LOOP      G-6.0 READY    - -      -          -      -       -/-/-
W1 PWS2          G-6.0 READY    1 -  0x0164a000 8192K  02/01.14   -/PCI/3
W2PWS0           G-6.0 READY    1 -  0x01694000 4096K  03/01.17   -/PCI/4
-----------------------------------------------------------------------------
```

In the table, some spaces and leading zeroes have been omitted. ‘-’ means there is no suitable value.

The columns in the table have the following meanings:

ID

Symbolic designation and type of CCs/TSP access points served. The following values are possible:

W[1-32]

For CC-WAN (X.21, V.24, V.35, ISDN), see section “Architecture of CCP profiles” on page 23.

T[1-6]

For a TSP access point which exists in the kernel (see section “Architecture of CCP profiles” on page 23).

TYPE

Type of the TSP access point (e.g. TPI-NEA, see example above) or of the CC. The familiar CC types are also output in the table below. If the CC type is unknown, only the hexadecimal value is output.

The following values are possible for CCs:

PCI bus CCs
PWS0
PWS2
PWXV-2
PWXV-4

Table 27: Possible values for CC type

VERS
: Version of the CC adapter/TSP. The CC adapter is the device driver for the CC in the operating system kernel.

STATE
: Status of the CC/TSP access point. The following values are possible:

  NONEX
  : The CC exists but cannot be managed by CMX.

  EXIST
  : The CC exists but is not attached to CMX, or the TSP is installed but not active.

  ATTACH
  : The CC is attached to CMX but is not operational.

  READY
  : The CC is loaded and regarded by the CC adapter as operational / the TSP access point is active.

  0xabcd
  : Hexadecimal representation of the status.

The following values are only relevant for CCs and not for TSPs:

I1
: Interrupt of the CC. Represented by:

  dd
  : decimal number (dd) from the range 0-99.

  xxxx
  : 4-digit hexadecimal representation.

I2
: Second interrupt of the CC. For representation, see I1.

MEM
: CC memory capacity in Kbytes.

HW/FW_VERS
: Hardware and firmware version of the CC.

CAB/BUS/SLOT

If possible, the position of the CC is displayed with the number of the cabinet, bus, and slot.

Information on the TSP access points

The information on the transport services offered by the TSP access points is divided into two tables. The meanings of the values output are explained using the following sample output.

```
ID    CCP       VERS  TSAP  TCEP  TIDU  TSP  SUBRV
================================================
T2 0x00a0    G-5.10  2000  2000  2183    1      0
T3 0x00a0    G-5.10  2048  2048  4089    1      0
T4 0x00a0    G-5.10  2048  2048  2043    1      0
T5 0x00a0    G-5.10  2048  2048  1409    1      0
T6 0x00a0    G-5.10  4096  4096 65321    1      0
W5 no info
------------------------------------------------
ID    TSPSEL  ETSDU UDCRQ UDCRS UDDRQ ADDRFORM
================================================
T2      -       12    92    92     1  WANNEA
T3      -       16    32    32    64  RFC1006
T3      -       16    32    32    64  LANINET
T4      -       16    32    32    64  WANSBKA
T5      -       32   256   256   129  WAN3SBKA
T5      -       32   256   256   129  SDLCSBKA
T6      -       16    32    32    64  LOOPSBKA
T6      -       16    32    32    64  TRSNASBKA
------------------------------------------------
```

In the table, some spaces and leading zeroes have been omitted. ‘-’ means there is no suitable value.

The columns have the following meaning:

ID

Symbolic designation and type of CCs/TSP access points served (see section „Information on the CCs/TSP access points (option -b␣all) on page 244).

CCP

If the TSP access point (ID=T[1-6]) is not active, or if no transport service is produced by the CCP (other IDs), “no info” is output here and all other fields are blank.

VERS

Version of the TSP access point in hexadecimal representation.

TSAP

Number of TS applications that the TSP can support simultaneously.

TCEP

Number of transport connections (TCEPs) that the TSP can sustain at one time.

TIDU

Maximum length of a TIDU (Transport Interface Data Unit) supported by TSP. A TIDU is the data unit that the application transfers to CMX in a single call to send data, or receives from CMX in a single call to collect data.

TSP

Maximum number of transport service providers (TSPs) which the CCP can serve simultaneously. With TSP access points (ID=T[1-9]) this value is always 1.

SUBRV

SUBRV contains the revision status of the CCP. This value has no meaning for TSP access points.

TSPSEL

For TSPSEL the TSP selector is output in binary form.

ETSDU

Length of the ETSDU (expedited transport service data unit) supported by the CCP/TSP. ETSDU gives the size of the expedited data unit that can be transferred by the CCP/TSP in a single transmit request. Expedited data is data transferred by the CCP/TSP with priority over normal data. The value 0 means that expedited data is not supported.

UDCRQ, UDCRS, UDDRQ

TS applications can pass information in the form of user data to the communication partner on connection setup and disconnection. The permitted length of this user data depends on the CCP/TSP and is defined as follows:

UDCRQ

Maximum length of user data during connection request (ICMX(L) call *t_conrq()*), XTI call *t_connect()*) from a local to a remote TS application.

UDCRS

Maximum length of user data when a local TS application replies to a connection request by a remote TS application (ICMX(L) call *t_conrs()*, XTI call *t_accept()*).

UDDRQ

Maximum length of user data during disconnection by a local TS application (ICMX(L) call *t_disrq()*, XTI call *t_snddis()*).

ADDRFORM

The address format or formats supported by the TSP access point. The meaning of the address formats is described in section “Address formats” on page 84. The address format is output in hexadecimal representation if no plain text string exists for the supported address format.

Information on all active TSAPs (option -s -v)

```
    TSAP       PID    THREAD-ID    TSTAT   #TCEP
==============================================
     0.0      9337            -     N/A       0
     1.0     10319            -     N/A       0
     2.0     10320            -     N/A       0
  5622.0      7942            1     N/A      75
  5622.0      7942            4     N/A      75
  5622.0      7942            5     N/A      75
  5622.0      7942            6     N/A      75
----------------------------------------------
```

The THREAD-ID column will not be shown if this is called without the option *-v*.

The columns in the table have the following meanings:

TSAP

Identification of the TSAP.

PID
: Process ID.

THREAD-ID
: identification of the thread.

TSTAT
: status of the TEP in the case of an XTI application. The following values are possible:

  UNBND
  : unbound

  IDLE
  : no connection has been set up

  OUTCON
  : outgoing connection setup request not yet answered

  INCON
  : incoming connection setup request not yet answered

  DATA
  : data transfer

  OUTREL
  : outgoing connection closedown request not yet answered

  INREL
  : incoming connection closedown request not yet answered

  N/A
  : not available (for ICMX(L) and ICMX(NEA) application)

#TCEP
: number of connections

Information on a particular TSAP (option -S *id* -v)

```
  TSAP       PID THREAD-ID     TCEP  L/R CSTAT  SDAT  RDAT ID  OWN_GLOBAL_NAME
=============================================================================
5624.0      8874         3  11593.0    R  DATA   17M   17M  T6  DEMO_PD01
    -       8874         4  11595.0    R  DATA   18M   18M  T6  DEMO_PD01
    -       8874         6  11609.0    R  DATA   15M   15M  T6  DEMO_PD01
    -          -         6  11611.0    R  DATA   16M   16M  T6  DEMO_PD01
    -          -         6  11613.0    R  DATA   17M   17M  T6  DEMO_PD01
    -          -         6  11615.0    R  DATA   16M   16M  T6  DEMO_PD01
    -          -         6  11617.0    R  DATA   15M   15M  T6  DEMO_PD01
    -       8874         7  11619.0    R  DATA   16M   16M  T6  DEMO_PD01
    -          -         7  11621.0    R  DATA   16M   16M  T6  DEMO_PD01
    -          -         7  11623.0    R  DATA   15M   15M  T6  DEMO_PD01
    -          -         7  11625.0    R  DATA   14M   14M  T6  DEMO_PD01
    -          -         7  11628.0    R  DATA   14M   14M  T6  DEMO_PD01
    -          -         7  11632.0    R  DATA   13M   13M  T6  DEMO_PD01
    -       8874         5  11601.0    R  DATA   15M   15M  T6  DEMO_PD01
    -          -         5  11603.0    R  DATA   16M   16M  T6  DEMO_PD01
    -          -         5  11605.0    R  DATA   16M   16M  T6  DEMO_PD01
    -          -         5  11607.0    R  DATA   17M   17M  T6  DEMO_PD01
-----------------------------------------------------------------------------
```

The THREAD-ID column will not be shown if this is called without the option *-v*.

Information on all currently available TSAPs (option -S all -v)

```
  TSAP       PID THREAD-ID     TCEP  L/R CSTAT  SDAT  RDAT ID OWN_GLOBAL_NAME
=============================================================================
   0.0      9337         -        -    -     -     -     - -  <CMX-Daemon>
   1.0     10319         -        -    -     -     -     - -  N/A
   2.0     10320         -        -    -     -     -     - -  $FJAM
5622.0      7942         1  11551.0    R   DATA    0B    0B T6  DEMO_PD01
    -          -         1  11552.0    R   DATA    0B    0B T6  DEMO_PD01

    -       7942         8  11482.0    R   DATA   18M   18M T6  DEMO_PD01
5623.0      8001        12  11480.0    L   DATA   18M   18M T6  DEMO_AD01
    -       8001        11  11481.0    L   DATA   15M   15M T6  DEMO_AD01
    -       8001         6  11484.0    L   DATA   18M   18M T6  DEMO_AD01
    -       8001        13  11485.0    L   DATA   17M   17M T6  DEMO_AD01
-----------------------------------------------------------------------------
```

The THREAD-ID column will not be shown if this is called without the option *-v*.

Information on the TCEP *id* (option -C *id*)

```
TCEP    L/R  CSTAT  SDAT  RDAT ID  OWN_GLOBAL_NAME  REMOTE_GLOBAL_NAME
=============================================================================
4712.0   L    DATA  202B  33B  T1  AD04             AD04
-----------------------------------------------------------------------------
```

Information on all currently available TCEPs (option -C all)

```
TCEP    L/R CSTAT  SDAT  RDAT ID   OWN_GLOBAL_NAME     REMOTE_GLOBAL_NAME
=========================================================================
4709.0   L  DATA  202B  110B  T1  Mueller.Egon.Mch-P  Meier.Egon.Pad
4710.0   L  DATA    0F    0F  T2  Mueller.Egon.Mch-P  Meier.Franz.Pad
4711.0   L  DATA   22K  121K  W1  Meier.Otto.Mch-P    Mueller.Egon.Pad
4712.0   L  DATA  202B   12B  T1  Meier.Otto.Mch-P    Mueller.Emil.Pad
4713.0   L  DATEX   0F    0F  T2  Meier.Otto.Mch-P    N/A
4714.0   L  DATA  102K   13K  T6  Meier.Otto.Mch-P    N/A
4715.0   R  DATST  44M   56M  W1  Meier.Otto.Mch-P    Mueller.Fritz.Pad
```

Output with option -C id -M

If the TS directory does not contain all the GLOBAL NAMES of TCEPs (output “N/A”), you may use the option *-M*. It provides an alternative output for a specific TCEP (or for all TCEPs if you specify the option *all* instead of *id*).

```
TCEP    LOCAL_NAME                    TRANSPORT_ADDRESS
=========================================================================
4712.0  TSEL LOOPSBKA A'Otto1'
     -  TSEL LANINET A'4711'
     -  TSEL RFC1006 A'D018V008'
     -   -                             TA LOOPSBKA A'Emil1'
```

The columns have the following meanings:

TSAP
: Identification of the TSAP. If this TSAP is not active, “no info” is output.

PID
: Process ID.

THREAD-ID
: Identifcation of the thread.

TCEP
: Identification of the TCEP.

L/R
: Specification as to how the transport connection was initiated.

  L
  : via the local application

  R
  : via the remote application

CSTAT
: Status of TCEP. The following values are possible:

  EXIST
  : TCEP exists

  DATA
  : data transfer

  DATEX
  : stop for normal data

  DATST
  : stop for normal and expedited data

  REDIN
  : redirection of connection announced

  DISIN
  : connection closedown indicator announced

  CONIN
  : connection setup indicator announced

  CONRQ
  : connection setup request transmitted

SDAT
: Amount of data sent (T = terabyte, G = gigabyte, M = megabyte, K = Kbyte, B = byte). Maximum 9999T; otherwise an overflow occurs (indicated by “OF”).

RDAT
: Amount of data received (cf. SDAT).

ID
: CC/TSP access point via which the connection was set up.

OWN_/REMOTE_GLOBAL_NAME
: GLOBAL NAME that was allocated to the TSAP in the TS directory (OWN = local, REMOTE = remote), or “N/A” if GLOBAL NAME was not found in the TS directory.

LOCAL_NAME
: One or several TSELs (transport selectors) of the local application.

TRANSPORT_ADDRESS
: Transport address of the remote application.

Information on LAN interfaces (option -i)

With the *-i* option you obtain information on all LAN interfaces available to TS applications.

```
INTERFACE   TYPE       STATE ADDRESS                    MACADDR
=====================================================================
hme0        ETHN(PCI) UP    172.25.236.26               0:80:17:28:7b:8
qfe0        ETHN(PCI) UP    10.99.218.44                8:0:20:e4:c4:34
qfe0        ETHN(PCI) UP    fe80::a00:20ff:fee4:c434    8:0:20:e4:c4:34
---------------------------------------------------------------------
```

The columns have the following meanings:

INTERFACE
: Name of the network interface.

TYPE
: Type of subnetwork reached via the interface.

STATE
: Status of the LAN interface. The following values are possible:

  UP
  : The interface is operational.

  DOWN
  : The interface is not operational.

ADDRESS
: IP address of the LAN interface.

MACADDR
: MAC address of the LAN interface, if available.

Information on WAN interfaces (option -b *id* -l)

In combination with options *-b␣-id* the option *-l* provides information on all WAN interfaces of a controller specified by *id*.

```
ID   TYPE    INTERFACE_1  INTERFACE_2  INTERFACE_3  INTERFACE_4
================================================================
W14  PWXV-4      V.24         UNUSED       X.21         X.21
----------------------------------------------------------------
```

Option *cmxinfo␣-b all␣-l* returns the following output (RM600):

```
ID   TYPE    INTERFACE_1  INTERFACE_2  INTERFACE_3  INTERFACE_4
================================================================
W3
W12  PWS2
W14  PWXV-4      V.24         UNUSED       X.21         X.21
----------------------------------------------------------------
```

The columns have the following meanings:

ID
: Symbolic identifier of the WAN controller with values W[1-32].

TYPE
: Type of controller.

INTERFACE_[1-4]
: Interface type. The type is variable for the following PCI controllers:

  PWXV-[2|4] with 2 or 4 lines: UNUSED, V.24, X.21, V.35V, V.35P, V.10, V.36, LOOP.

10.6 Controlling and editing the CMX library trace (cmxl)

The trace mechanism of the CMX library is activated and controlled via the CMXTRACE environment variable. The trace entries of a process are collected in compressed, binary format in a dynamically created buffer and are periodically written to temporary files. These files are edited in a separate step using *cmxl*.

In the case ofr multi-threading (MT) there are some differences. These will be drawn to your attention in the text.

Controlling the trace mechanism - CMXTRACE

Every CMX *t_attach()* call issued by a process evaluates the CMXTRACE environment variable and, when appropriate, activates the trace mechanism. CMXTRACE must have been set before the application is started, i.e. prior to the first *t_attach* of the process to be monitored. Following activation of the trace mechanism, the temporary file *CMXLapid* with process ID *pid* is opened, if it has not already been opened. The files are granted the access permissions `rw-------` (0600). Memory is then dynamically reserved for buffering the trace entries.

In multi-threading applications, all trace entries effected by process threads are written into a temporary file *CMXLapid*.

Memory and files are reserved for the lifetime of the process.

The options specified in CMXTRACE control the trace mechanism. Options *-s*, *-S*, *-D* and *-G* determine the extent of logging. Options *-p* and *-r* control buffering and cyclical overwriting of the file.

Syntax of the environment variable

CMXTRACE="[␣**-s**] [␣**-S**] [␣**-D**] [␣**-p**␣fac] [␣**-r**␣wrap] [␣**-f**␣directory] [␣**-G**␣dirx]";

export␣CMXTRACE

The options *-s, -S* and *-D* determine the type of the trace mechanism. To activate the trace mechanism, a value must be specified.

-s

An ordinary log is kept of the ICMX(L) calls, their arguments, the options and user data.

-S

A detailed log is kept of the calls, their arguments, the contents of any options and the user data in its full length.

The options *-s* and *-S* are mutually exclusive.

-D

The calls are then logged in detail together with additional information on system calls. This option can only be specified in addition to *-s* or *-S*.

-p␣fac

The decimal digit *fac* determines the buffering factor. The amount of buffering is *fac* * 1024. If 0 is specified for *fac*, every trace entry is written immediately to the file (unbuffered).

fac=0...8.
-p␣fac not specified: *fac=1* is assumed.

-r␣wrap

The decimal number *wrap* specifies that logging is to be directed to the second temporary file *directory/CMXMapid* after *wrap* * 1024 bytes. Here, *pid* means process ID.

The second file *CMXMapid* is handled by the trace mechanism in the same way as *CMXLapid*.

After every *wrap* * 1024 bytes, the trace mechanism switches between *CMXLapid* and *CMXMapid*. When this is done the old contents of the second file are lost.

-r wrap not specified: *wrap* = 512 is assumed.

-f␣directory

The trace files are written to the specified directory.
-f directory not specified: */var/opt/SMAWcmx/tmp* is assumed for *directory*.

-G␣dirx

A detailed log is made with additional information on DIR.X calls. The log file is stored in the current directory under the name *logfile.pid*.

dirx can have the following values:

0x0
trace TNS names

0x02
trace DIR.X names

0x04
trace configuration

0x08
trace internal calls

0x10
trace “alarms”

These values can be combined in binary form.

The option is ignored in the MT library.

Editing the trace information - cmxl

cmxl reads the entries generated by the trace mechanism from the temporary file *file*, processes them in accordance with the specified options and outputs the results to *stdout*.

Syntax

cmxl [␣**-c**] [␣**-d**] [␣**-e**] [␣**-t**] [␣**-v**] [␣**-x**] [␣**-D**] ␣file ...

The options specify which trace entries from *file* are to be edited. More than one of the values described in the following may be specified per *cmxl* call. Only options *-v* and *-x* are mutually exclusive.

If no option is specified, *cdex* is assumed.

-c

Editing is performed for the ICMX(L) calls:

– for attaching/detaching the application to/from CMX,
– for connection setup and disconnection, and
– for connection redirection.

-d

Editing is performed for the ICMX(L) calls:

– for data exchange, and
– for flow control.

-e

Editing is performed for the ICMX(L) calls for event handling.

-t

In addition to the logging of error messages, explicit editing is performed for the *t_error()* calls.

Error messages are always logged, even if this option is not specified.

-v

Detailed editing is performed for the ICMX(L) calls, their arguments, the options and the user data. The extent of editing of the data depends on the options specified in CMXTRACE.

-x

Limited editing is performed for the calls and their arguments, *excluding* options and user data.

-D

Detailed editing is performed with additional information on system calls.

file ...

Name of one or more files with trace entries to be edited.

Output formats (CMXTRACE, NEATRACE, cmxl, neal)

To understand the description of the trace information in this section, you must be familiar with the ICMX(L) and ICMX(NEA) program interfaces of CMX. These interfaces are described in the “Programming Applications” manual [1].

The format of the trace information edited by *cmxl* and *neal* is the same. Only the output format of *cmxl* will therefore be described here. This description can also be used to interpret the *neal* output. Just replace *t_*... with *x_*... except for calls *t_vdatarq()* and *t_vdatain()*.

The trace information edited by *cmxl* is output in the following format:

```
Header  ICMX(L) TRACE (G-V6.0) date hh:mm:ss
lines   OPTIONS ‚cdex‘ TRACE FILE ‚trace file‘ ICMX(L) V5.1E

1st line        time stamp t_xxxxx(args in %d, 0x%x, %s)
2nd line        [Options and user data in %d, %x, %s]
3rd line                    [TRANSPORT ADDRESS, LOCAL NAME]
4th line        [Results, events in %d, %x, %s]
1st line        time stamp ... next call ...
```

The two header lines are output once at the start of the output of the trace information. They contain:

– version and CMX version (here V6.0)
– date and time (hh:mm:ss) tracing was started
– editing options selected (here the default options *-c, -d, -e, -x*)
– name of the edited trace file
– version of the program interface

Trace information for the individual command calls is output over several lines, the different lines having different formats. The formats for these lines are described in the following (the lines are designated 1st line through 4th line).

1st line

A time stamp is placed at the start of the 1st line for each logged command call. In *cmxl* the time stamp has the format hh:mm:ss:msc, where hh = hours, mm = minutes, ss = seconds and msc = milliseconds. In *neal* the time stamp has the format hh:mm:ss.

This is followed by the logged command call (*t_xxxxx*) and, enclosed in parentheses, the values of the arguments (*args*), in the order required by ICMX(L). The arguments are shown in decimal form (%d), hexadecimal form (0x%x) or symbolic form (%s).

When interpreting the logged values, note the following:

– For the arguments *datap, fromaddr, name, opt* and *toaddr* the address transferred is shown (0x%x).

– For the arguments *chain, flags, cmode, pid* and *reason* the corresponding values are shown (0x%x, %d, or %s), even when the arguments are actually the addresses of these values.

– For the *tref* argument the corresponding value is shown (0x%x). The calls *t_conrq()* and *t_event()* are exceptions; here the address transferred is shown for *tref*.

– When data lengths are logged, e.g. *datal*, the value in effect at the time of the call is shown (%d). For the calls

  t_concf(), t_conin(), t_datain(), t_vdatain(), t_xdatin(), t_redin() and *t_disin()*

  any modified value in effect upon return is also shown (%d), the two values being separated by "><".

The 2nd and 3rd lines are output only when option *-v* is specified for *cmxl* and the trace mechanism has collected appropriate information (option *-S*).

2nd line

In the 2nd line the options are logged together with option numbers and option fields. They appear in the order declared in the option structure as defined in the include file *<cmx.h>*.

When interpreting the logged values, note the following:

– For the option fields *t_maxl, t_optnr, t_timeout* and *t_xdata* the value transferred is shown (0x%x, %d or %s). In the case of *t_udatap*, the address transferred is shown.

– For the argument t_udatal, the value in effect at the time of the call is shown (%d). For the calls *t_conin()*, *t_concf()*, *t_disin()* and *t_redin()*, any modified value in effect upon return is also shown (%d), the two values being separated by "><".

3rd line

Lines having "3rd line" format record the TRANSPORT ADDRESS, the LOCAL NAME and user data, if logged by the trace mechanism and edited by *cmxl*. This is followed by the data, shown in hexadecimal and printable form.

Example of output in the 3rd line:

```
Offset   Data, shown in hexadecimal form   .printable form.
 0     4c4f4b41 4c455220 4e414d45 20242424 .LOCAL NAME $$$.
```

4th line

The 4th line of an entry is used to log the result of a call. In the event of an error, T_ERROR is entered. If the call was successfully executed, the result is logged only if it deviates from T_OK. If this is the case, the result is logged together with the information returned by the call,

i.e. for

- *t_attach()*, the result T_NOTFIRST,
- *t_conrq()*, the transport reference (tref) supplied,
- *t_event()*, the event reported and the corresponding transport reference (tref),
- *t_datain()*, *t_vdatain()*, *t_xdatin()*, the length of the data still to be read,
- *t_datarq()*, *t_vdatarq()*, *t_xdatrq()*, the results T_DATASTOP, T_XDATSTOP.

As a rule, logged call results are shown in the following forms:

- decimal (%d), for lengths or values
- symbolic (%s), if a corresponding definition exists in the file *<cmx.h>*
- hexadecimal (0x%x) in all other cases

If symbols are defined for an argument or option field, but the value does not correspond to any of the permitted symbols, output will be displayed in hexadecimal form (0x%x), preceded by a question mark (?). T_ERROR results are marked with several # characters to make them stand out.

Example of editing with cmxl

Options *c, d, e* and *x* were selected for editing the trace information. This means that all calls for attaching to and detaching from CMX, connection setup, disconnection, connection redirection, data exchange and flow control are edited with their arguments but without options or user data.

ICMX(L) Trace Expansion *Expanded: Jan 1 13:12:41*

```
Trace collected with data length 32 starting Jan 1 13:10:52.
Application was running in 32-bit mode.
Trace expanded by CMX 5.1 from file „CMXLa01180“ with options
c      expand ICMX connection handling calls (set by default)
d      expand ICMX data and flow control calls (set by default)
e      expand ICMX event handling calls (set by default)
v      verbose mode showing args, options, user data
using T_MSG_SIZE=256, FD_SETSIZE=1024.

Traced System and CMX: SunOS PGTR0046 5.7 106541-04 sun4us; CMX
5.1E00 14

hh:mm:ss.msc ICMX Call t_*****or (intermediate) results
```

```
13:10:52.000 t_getloc(0xffbefe00, NULL)
     glob:
     0 52315f32 315f3030 30302e50 322e7266 |R1_21_0000.P2.rf|
    10 63313030 362e5049 54                |c1006.PIT       |
     loc 0xff3870cc: RFC1006
     0 01000018 000e0000 00000004 0008d7c9 |                |
     10 e3f2f0f0 f0f00000                  |                |
13:10:52.000 t_attach(0x3e034, 0x3c454)
     opt:
     t_optnr T_OPTA1 (1) t_apmode T_PASSIVE (2)
     t_conlim 8
     name: RFC1006
     0 01000018 000e0000 00000004 0008d7c9 |                |
    10 e3f2f0f0 f0f00000                   |                |
14:24:16.000 returns T_OK (0)
14:24:16.000 t_event (0x3e02c, T_WAIT, 0x4e4a0)
     opt:
     t_optnr T_OPTE1 (1) t_timeout T_NOLIMIT (-1)
14:24:16.000 returns T_CONIN (5) tref 0xa53
     t_attid 0x104 t_uattid 0x0
     t_ucepid 0xe t_evdat 0x0 (0)
14:24:16.000 t_conin (0xa53, 0xffbef9e8, 0xffbef8b0, 0xffbefb40)
     opt:
     t_optnr T_OPTC1 (1) t_udatap 0xffbef7a8 t_utada1 256><0
     t_xdata T_NO (0) t_timeout T_NO (0)
     toaddr:RFC1006
```

Files

CMXLapid, CMXMapid
: Files with compressed trace entries in binary format.

logfile.pid
: Contains DIR.X trace entries.

If not otherwise specified for CMXTRACE, the files *CMXLapid* and *CMXMapid* are created for the CMX library trace in the */var/opt/SMAWcmx/tmp* directory. *pid* is the process ID.

10.6.1 Notes about multi-threading

A CMX library trace generated in CMX V6.0 can be processed and prepared in a thread-specific way. The command *cmxl_mt* is provided for this purpose. It interprets the ASCII file *file* previously generated in binary format by *cmxl* (*cmxl ... file*). For diagnostic purposes the complete corresponding ASCII file should be made available.

The command has the following syntax:

cmxl_mt [␣**-h**] [␣**-t**␣{ tid | **all** }] [␣file ...]

Options and parameters

cmxl_mt (command without parameters)
The ASCII trace file is read from *stdin*; only the trace header and the thread IDs listed in the trace files are output to *stdout*.

-h
Displays a command syntax description.

-t tid
The ASCII trace file is read from stdin; all entries for thread ID *tid* are output to stdout.

-t all
The ASCII trace file is read from stdin; the entries for each thread are written to the file *file.tid*. The trace header and the thread ID list is stored in the *file.hdr* file.

file ...
The ASCII trace file *file* is read and only the trace header and trace ID are output to *stdout*.

-t tid file
The ASCII trace file *file* is read and entries for the thread <tid> are output to stdout.

-t all file
The ASCII trace file *file* is read; the entries to to each thread are written to the file *file.tid*. The trace header and the thread ID list is stored in the *file.hdr* file.

Display formats

For MT output, the display format differs only slightly compared to the previous display formats. In the ASCII file produced with *cmxl*, the thread ID in the form "[*tid*]" is also identified (see below). All subsequent trace entries are assigned to this thread up to the point where a different thread IDis is indicated. The meaning of the output lines remains the same.

Examples

Display format of the *tracefile* edited by cml:

```
ICMX(L) Trace Expansion                          Expanded: Oct  1 14:34:48

Trace collected with data length 0 starting Oct  1 14:30:58.
Application was running in 32-bit multithreaded mode.
Trace expanded by CMX 6.0 from file "CMXLa27504" with options
c       expand ICMX connection handling calls (set by default)
d       expand ICMX data and flow control calls (set by default)
D       expand system calls (sockets etc.; implies v)
e       expand ICMX event handling calls (set by default)
v       verbose mode showing args, options, user data
X       expand XTI system calls in case of ICMX over XTI
using T_MSG_SIZE=256, FD_SETSIZE=1024, openmax=256,
      fac=0, wrap=1024000000.

Traced System and CMX: SunOS PGTR0046 5.9 Generic sun4us; CMX 6.0E50
09hh:mm:ss.msc ICMX Call t_***** or (intermediate) results
[0001]
14:30:58.000 t_getloc(0xffbffb2d, NULL)
                glob:
               0  44454d4f 5f414430  31                 |DEMO_AD01       |
                loc 0x28d68: LOOPSBKA
               0  01000018 000e0000  00000400 00094445  |              DE|
              10  4d4f5f41 44303100                     |MO_AD01         |
14:30:58.000 t_getaddr(0xffbffb3a, NULL)
                glob:
               0  434d585f 504153                       |CMX_PAS         |
                addr 0x28e30: LOOPSBKA
               0  02000013 04000010  00098007 434d585f  |            CMX_|
              10  504153                                 |PAS             |
[0002]

14:30:58.000 t_attach(0x23df4, 0xfeffbf7c)
                opt:
                t_optnr T_OPTA6 (6)  t_apmode T_ACTIVE (1)
                t_conlim 1
                name: LOOPSBKA
               0  01000018 000e0000  00000400 00094445               DE|
               10  4d4f5f41 44303100                     |MO_AD01        |
```

Display format of the header file *tracefile*.hdr:

```
ICMX(L) Trace Expansion                          Expanded: Oct  1 14:34:48

Trace collected with data length 0 starting Oct  1 14:30:58.
Application was running in 32-bit multithreaded mode.
Trace expanded by CMX 6.0 from file "CMXLa27504" with options
c       expand ICMX connection handling calls (set by default)
d       expand ICMX data and flow control calls (set by default)
D       expand system calls (sockets etc.; implies v)
e       expand ICMX event handling calls (set by default)
v       verbose mode showing args, options, user data
X       expand XTI system calls in case of ICMX over XTI
using T_MSG_SIZE=256, FD_SETSIZE=1024, openmax=256,
      fac=0, wrap=1024000000.

Traced System and CMX: SunOS PGTR0046 5.9 Generic sun4us; CMX 6.0E50 09

hh:mm:ss.msc ICMX Call t_***** or (intermediate) results

Found Thread Id's
    4
    3
    5
    1
    2
```

Display format of the thread-specific prepared file *tracefile.tid*:

```
hh:mm:ss.msc ICMX Call t_***** or (intermediate) results
14:30:58.000 t_attach(0x23df4, 0xfeffbf7c)
                opt:
                t_optnr T_OPTA6 (6)  t_apmode T_ACTIVE (1)
                t_conlim 1
                name: LOOPSBKA
              0  01000018 000e0000  00000400 00094445  |         DE|
             10  4d4f5f41 44303100                     |MO_AD01    |
14:30:58.000    T_COM (0)
14:30:58.000    open("/dev/SMAWcmx/cxnet", 0) == 5
14:30:58.899    ioctl(5, STINIRQ, 0x330cc) = 0
14:30:58.899    T_OK (0)
                t_uattid 0x0 t_attid 0x2d06 t_sptypes 0xc00
                t_cclist (0x331c0): 1 504
14:30:58.899  returns T_OK (0)
```

10.7 CMX monitor (cmxm)

The CMX monitor observes the current activities of the CMX automaton. It outputs counter states and ascertains statistics about the load on the individual CMX components and the TSP access points in the operating system kernel.

The CMX monitor provides information on the following:

– the number of attached TS applications and the number of attached processes controlling these

– the number of existing transport connections to remote and local communication partners

– the activities of the TS applications, e.g. the number of calls from ICMX and XTI commands per second

– the amount of data sent and received per second

– the TSP access points in use, e.g. whether a TSP access point is operational or not, how many transport connections are active on the individual TSP access points, etc.

The statistics obtained by the CMX monitor are listed in detail under the descriptions of the output formats.

You can call the CMX monitor using the *cmxm* command. In the call you can choose from 3 operating modes for the CMX monitor. These differ in the type and extent of the information provided and in the output format:

– tabular output of statistics

– semi-graphical output of statistics

– summary output of counter states

Tabular output of statistics

The CMX monitor cyclically calculates the levels, rates of change (units per second) and the ratios of various counters. You can set the cycles in which the CMX monitor calculates and outputs these values when you call *cmxm*. The output is in the form of a table to standard output.

Semi-graphical output of statistics

The CMX monitor ascertains the same statistics as for the tabular output. The number of variables output is limited to the statistics for data exchange between the different components. The statistics are output in the form of diagrams to standard output. The meanings of the output values are given in the description of the tabular output in the next section.

Summary output of counter states

The CMX monitor simply outputs to standard output the counts run up for requests to the individual components since operating system startup or since last reset of statistics with option *-z* . The output values can be interpreted from the following description of the tabular output (see section “Format of tabular output of the CMX monitor” on page 270). Note that the values are absolute. For example, it is not the average number of data items transmitted per second that is output as in the tabular output, but the number of data items transmitted since system startup. Note that the counters are reset to zero if a threshold is exceeded, and counting recommences from there.

Example of summary output (cmxm -s command):

```
CMX MONITOR (5.1): AUTOMAT STATISTICS Jan 14 14:38:58
CMX AUTOMAT G-5.1  8 CC BOOT Jan 13 19:47 TIME 14:38:58
       5 TSP
   15878 APPLICATIONS etablished
   16009 ATTACH invokes
   29194 TCEP set ups(4% refusals, 3% aborts)
39057077 ICMX(S3) commands (0% nomem 0% bad 0% busy)
 2277821  ICMX(S3) events (0% sync)
12357613 ICMX(CC) commands sent (0% blocked)
 9275589  ICMX(CC) commands got (0% blocked)
 9190014  blocks sent (33% with flow control)
 9218321  blocks got (33% with flow control)
       8.003240 Gbyte sent
       8.048487 Gbyte got
```

In all three modes you can select whether the information gathered by the CMX monitor is to cover all the activities of the CMX automaton or only those relating to a specific TSP access point.

The CMX monitor uses the active system as the source for its statistics. This means that it reads the current counts, edits them in accordance with the mode selected, and outputs them.

As an alternative to interactive output, a background process (daemon) can be used to generate a statistics file, which the CMX monitor then edits. The daemon is started and terminated with *cmxmd*. It periodically collects statistics in a file which you can then have edited by the CMX monitor at any time desired after the daemon is terminated. The results can then be output in tabular or semi-graphical form to standard output.

cmxm is terminated at the keyboard with DEL or ENTER (and q for semi-graphical output), or by the signal SIGINT. If terminated by other means following semi-graphical output, the terminal will be in an undefined state.

The command has the following syntax:

cmxm [␣**-a**] [␣**-c**␣id] [␣**-f**[␣file]]
[␣**-l**␣{ ln | **all** }]
[␣**-i**␣sec] [␣**-b**␣hms] [␣**-e**␣hms]
[␣**-n**␣cnt] [␣**-s**] [␣**-v**] [␣**-z**]

The options *-f, -s* and *-v* are used to specify how the statistics are to be output and the source used by *cmxm* for them.

If *cmxm* is started with no options specified, statistics for the CMX automaton are output cyclically every second in tabular form using the default values of the options *-c␣id, -i␣sec, -n␣cnt*.

-a

The statistics for the CMX automaton are edited (default value).

-c␣id

The statistics for the TSP access points *id* are edited, assuming they are present in your computer. The individual TSP access points are specified in *id* as follows:

W[1-32]

For CC-WAN (X.21, V.24, V.35, ISDN).

T[1-6]

For TSP access point (see section “Architecture of CCP profiles” on page 23).

-f [␣file]

cmxm interprets the statistics from the *file* statistics file in which the *cmxm* daemon *cmxmd* has collected statistics.

If *file* is not specified, the file *cmxm*[*id*].*DD* is assumed. Here, *id* either identifies a CC/TSP access point as with the *-c* option or is blank, and *DD* is the day of the month (beginning with 1).

l␣ln | **all**

displays statistics (throughput rates) for *all* lines (max. 4) or for line *ln* of a specified WAN controller *(-c␣id)*.

-i␣sec

sec determines the seconds per interval for cyclical output of the statistics. For *sec*, specify a positive decimal number. Statistics will then be output every *sec* seconds to *stdout*.

-i␣sec not specified: *sec=1* is assumed.

In the case of line statistics (*-l* option), choose a value ≥5.

-b␣hms

For *hms*, specify the starting time for interpretation. This must be specified in the form hh[:mm[:ss]] (hh = hour, mm = minute, ss = second). Specifying *-b hms* is only appropriate if *-f* was specified.

-b hms not specified:

hms = 00:00:00 is assumed.

-e␣hms

For *hms* specify the stopping time for interpretation. This must be specified in the form hh[:mm[:ss]] (hh = hour, mm = minute, ss = second).

Specifying *-e hms* is only appropriate if *-f* was specified for *option*.

-e hms not specified: *hms* = 24:00:00 is assumed.

-n␣cnt

cnt determines the number of cycles for cyclical output. For *cnt*, specify a positive decimal number.

-n␣cnt not specified or negative: *cnt=continuing* is assumed.

-s

The statistics are output in summary form.

-v

The *-v* option causes the edited results to be output in semi-graphical form to *stdout*, which in this case must be connected to a terminal.

Together with *-c all*, the *-v* option produces a corresponding presentation of statistics for the CMX automaton and the available CCs.

-z

The -z option resets all statistic data for CMX automaton and CCs/TSPs to zero.

The following sections describe the format of the tabular and summary output of the CMX monitor.

Format of tabular output of the CMX monitor

The information gathered by the CMX monitor depends on whether you have requested statistics for the CMX automaton or statistics for a CC/TSP access point. The two types of statistics are described separately below.

Statistics for the CMX automaton

When output in tabular format, the statistics are arranged into lines. The output begins with three header lines. After every 20 lines the three header lines are output again.

```
 CMX MONITOR (5.1): AUTOMAT STATISTICS
 CMX AUTOMAT G-5.1 8CC BOOT Jan 1 08:09  TIME 13:29:40 IVAL 1
PG 1
TSAP   ATT     TEP      ICMX(S3)      ICMX(CC)     TCEP    DATA
SEND  DATA GET
 act   act   act bu  cmd n e  evt wt  snd  get   act rj ab bls
kbys blg  kbyg
  3    11    11  0   462 0 0  39  0   168  148    64  0  0 118
243.6 147 309.6
  3    11    11  0    9  0 0   3  0    1    3     64  0  0  0
 0.0   3   6.0
   ...
```

The first lines contain the following:

– the version of the CMX automaton

– the number of CCs/TSP access points supported by the CMX automaton

– the time of system startup (BOOT)

– the current time (TIME)

– the interval at which statistics are taken (IVAL)

– the page (PG); this corresponds to the number of times the header lines have been output

The information gathered by the CMX monitor is grouped together under headings. The next line contains the headings, and the line under this the associated statistical variables. The division into headings is by (internal) interfaces. In the following description of the statistics each heading is listed together with its meaning, below which appear the corresponding statistical variables and their meanings.

TSAP
: TS applications.

  act
  : Currently active TS applications (ICMX(L) and XTI applications).

ATT
: Attachments.

  act
  : Currently active attachments of processes within these TS applications.

TEP
: Transport endpoints.

  act
  : Currently active transport endpoints (TEPs). TEPs are the XTI transport endpoints and the processes attached in ICMX applications or threads.

  bu
  : Percentage of collisions in device file openings.

ICMX(S3)
: These statistics relate to the procedures at the system interface between the user process (TS application, ICMX(L) and XTI library) and the CMX automaton in the operating system kernel.

  cmd
  : Average number of calls issued by TS applications to the CMX automaton per second.

  n
  : Percentage of calls in *cmd* temporarily rejected due to a resource bottleneck.

e
: Percentage of calls rejected due to an error.

evt
: Average number of events per second.

wt
: Percentage of events awaited synchronously by TS applications.

ICMX(CC)
: These statistics relate to activities at the interface between the CMX automaton and the drivers in the kernel which control access to the TSPs (WAN adapter, TPI adapter). The following values refer to all TSPs in use.

snd
: Requests from the CMX automaton to the CCs/TSPs to send data. The average number of requests per second is given.

get
: Requests from the TSPs to the CMX automaton to fetch received data. The average number of requests per second is given.

TCEP
: These statistics relate to the activities of transport connections set up via the CMX automaton.

act
: Currently active transport connections.

rj
: Percentage of rejected connections. This value includes both the connections rejected by local TS applications and the connection requests made by local TS applications but rejected by their communication partner.

ab
: Percentage of transport connections forcibly closed down by the system.

DATA SEND

This statistic relates to the data sent, totaled up for all CCs/TSP access points.

bls

Average number of blocks sent per second. One block corresponds to one TIDU.

fcs

Number of data blocks sent divided by number of transmission credits in percent.

kbys

Average number of kilobytes of data sent per second.

DATA GET

This statistic relates to the data received, totaled up for all CCs/TSP access points.

blg

Average number of blocks received per second. One block corresponds to one TIDU.

fcg

Number of blocks received divided by number of given receive credits in percent.

kbyg

Average number of kilobytes of data received per second.

Statistics for a CC/TSP access point

When output in tabular format, CC/TSP access point statistics are arranged into lines. The output begins with three header lines. After every 20 lines the three header lines are output again.

```
CMX MONITOR (5.1): CC STATISTICS                      Jan 1 13:42:43
CC ADAPTER  G-5.10  CC T6 RDY  BOOT Jan  1 08:09    TIME 13:42:43
CCP VERSION 0x00a0 ADDRFORMS:  LOOPSBKA TRSNASBKA
     0 TSAP set ups
     0 TCEP set ups (0% refusals, 0% aborts)
      0 ICMX(S3) events (0% sync)
      0 ICMX(CC) commands sent (0% blocked)
      0 ICMX(CC) commands got (0% blocked)
      0 blocks sent (0% with flow control)
      0 blocks got (0% with flow control)
     0 TSP starts
     0 TSP interrupts
     0.000000 Gbyte sent
     0.000000 Gbyte got
     0 ADM&DIAG starts
     0 ADM&DIAG interrupts
     0 ADM&DIAG Kbyte sent (0 bytes per block)
     0 ADM&DIAG Kbyte got (0 bytes per block)
```

The first lines contain the following:

– the version of the driver which controls access to the CC/TSP access point

– the symbolic designation of the CC/TSP access point and the current status of the CC/TSP access point

  The symbolic designations for the CCs/TSP access points have the following meanings:

  W[1-32]
  : For CC-WAN (X.21, V.24, V.35, ISDN).

  T[1-6]
  : For TSP access point (see section “Architecture of CCP profiles” on page 23).

  The status of the CC/TSP access point can be one of the following:

  ATT
  : The CC/TSP access point is attached to CMX but is not operational.

  RDY
  : The CC/TSP access point is attached to CMX and is operational.

– the time at which the CC/TSP access point was last loaded (BOOT)
– the current time (TIME) or the time at which the CC/TSP was taken out of operation (DOWN)
– the interval at which statistics are taken (IVAL)
– the page (PG); PG corresponds to the number of times the header lines have been output

The information gathered by the CMX monitor is grouped together under headings. The next line contains the headings, and the line under this the associated statistical variables. The division into headings is by internal interfaces. In the following description of the statistics each heading is listed together with its meaning, below which appear the corresponding statistical variables and their meanings.

ICCP
: This statistic relates to the procedures at the interface between the CC device driver in the operating system (CC adapter) and the CC. Administration procedures are not included.

  cmd
  : Average number of calls for communication to the CC per second.

  int
  : Average number of interrupts from the CC per second.

ICMX(S3)
: This statistic relates to the activities at the interface between the user process and operating system kernel relating to this CC/TSP access point.

  evt
  : Average number of events per second.

  wt
  : Percentage of synchronously awaited events.

ICMX(CC)

This statistic relates to the procedures at the interface between the CMX automaton and the drivers in the kernel which control access to TSPs (WAN adapter, TPI adapter).

snd

Requests from CMX to the TSP access point to send data. The average number of requests per second is given.

get

Requests from the TSP access point to CMX to fetch received data. The average number of requests per second is given.

cw

Number of requests in *snd* and *get* that are in wait status.

pw

Percentage of all send and fetch requests that were placed in wait status.

TSAP

Activities of the TSAPs. TSAPs are the transport service access points to which the TS applications are linked and via which you access the services of the transport systems (TSPs).

act

Currently active TSAPs on the TSP access point.

TCEP, DATA SEND, DATA GET

The statistical variables under these headings have the same meanings as the corresponding variables in the statistics for the CMX automaton. The value output, however, only refers to one TSP access point.

Line statistics

The *-l␣ln|all* option provides line statistics for the specified WAN controller. These statistics are cyclically transferred from the controller to the host computer. Because formatting and editing by *cmxm* taking some time, it is a good idea to set a default interval for tabular line-by-line editing of 15 seconds *cmxm -i␣sec␣*15).

The following example shows the output of a command *cmxm -cW2 -l␣all -i␣20*. It provides line statistics for all lines of controller W2 with 20 sec. interval.

```
CMX MONITOR (6.0): LINE STATISTICS                          Oct 15 13:50:57
CC ADAPTER G-5.10 CC W1/RDY  BOOT Oct 15 13:19 TIME 13:50:57  IVAL 20 PG1
   INTERFACE_1   |    INTERFACE_2   |   INTERFACE_3    |   INTERFACE_4
 SEND  GET  FCS- |  SEND  GET  FCS- |  SEND  GET  FCS- |  SEND  GET  FCS-
kBy/s kBy/s errs | kBy/s kBy/s errs | kBy/s kBy/s errs | kBy/s kBy/s errs
  0.0   0.0    0 |   -     -      - |     -     -    - |  0.0   0.0   0.0
  6.8   6.9    0 |   -     -      - |     -     -    - |  0.0   0.0   0.0
  6.6   6.7    0 |   -     -      - |     -     -    - |  0.0   0.0   0.0
```

The cmxm *-cW2 -l1* command provides line statistics for line 1:

```
CMX MONITOR (6.0): LINE STATISTICS                          Oct 15 13:54:44
INTERFACE (1) of CC W1/RDY  BOOT Oct 15 13:19 TIME13:54:44 IVAL 15 PG1
   DATA-SEND      DATA_GET       HDLC       HDLC       Parity-/
blk/s kBy/s     blk/s kBy/s    abrt/ovr   lng-errs     FCS-errs
    0   0.0         0   0.0           0          0            0
    9   9.4         8   9.5           0          0            0
    6   6.2         5   6.2           0          0            0
```

The meaning of the two header lines is described in the previous output example. The next two lines are headings for components according to which the statistic values are grouped.

SEND/GET kby/s
: Average amount of data transmitted/received via this line in kilobytes per second.

Parity-/FCS-errs
: Number of FCS or parity errors per cycle via this line.

DATA-SEND/DATA-GET
: Statistics for data transfer via this line.

  blk/s
  : Average number of blocks transmitted/received per second.

  kBy/s
  : Average number of kilobytes transmitted/received per second.

HDLC
: Statistics for HDLC protocol per output interval.

  abrt/ovr
  : Number of received HDLC aborts/overruns.

  lng-errs
  : Number of HDLC length errors (too long/too short).

Note that the output format may slightly change in future versions.

Examples

cmxm -v -c W1 -i 10 -n 20
: This command outputs the activities of the controller W1 in semigraphical format every 10 seconds and twenty times.

cmxm -f -b 8:00 -e 16:30 -i 300 /var/opt/SMAWcmx/tmp/cmxm.21
: This command provides the following output: from the file containing daily statistics for the 21st of the month (created with cmxmd), the activities between 8 a.m and 4.30 p.m. (at 5 minutes intervals) are displayed in tabular form.

cmxm -s -c T5
: This command outputs all activities of the TSP TP5 up to the present in summarized form.

Files

cmxm[*id*].*DD*
: Statistics for the day of the month *DD* with *id* as per option *-c* or blank. Please see the Release Notice for the location of the file in your computer's file system.

See also

cmxinfo, cmxmd.

10.8 CMX monitor daemon (cmxmd)

The CMX monitor daemon *cmxmd* cyclically collects statistical data in the background on the current activities in CMX and logs it to a file for later interpretation for the day by *cmxm*. The statistics cover the activities at the interface as well as the statistics of ICMX/XTI applications, their connections and their data throughput. They may be called globally or just for one specific CC/TSP access point.

When called without arguments, *cmxmd* collects statistics on the CMX automaton from the running system periodically (every 10 seconds) until the end of the day (24:00:00 h) and records them in the file *cmxm*[id].DD (DD is the day of the month, beginning with 1). By specifying options and arguments the monitor daemon *cmxmd* can be controlled more precisely. *cmxmd* either terminates by itself at the end of the day or is terminated by a signal (preferably SIGINT) or by being called with the *-h* option.

cmxmd writes its process ID to the file *cmxmd*[*id*].*pid* and it writes self-explanatory information on its execution to the trace file *cmxmd*[*id*].*trc*.

Please see the Release Notice for the name of the directory in which you can start the CMX monitor daemon in your system.

Syntax

cmxmd [␣**-h**] [␣**-c**␣id] [␣**-i**␣sec] [␣**-n**␣cnt] [␣**-o**␣file]

-h

A *cmxmd* daemon previously started with *cmxmd* is terminated when the *-h* option is specified. If the *-c* option was specified when the *cmxmd* daemon was started, the *-c* option must also be specified in the same form at termination.

-c␣id

Id specifies the CC/TSP access point for which *cmxmd* is to collect statistics. If the specified CC/TSP access point is not present in your computer, the CMX automaton rejects the command and a corresponding error message is output to *stderr*.

The individual CCs/TSP access points are specified in *id* as follows:

W[1-32]
: For CC-WAN (X.21, V.24, V.35, ISDN).

T[1-6]
: For a TSP access point (see section “Architecture of CCP profiles” on page 23).

If the *-c* option is specified when *cmxmd* is started, it must also be specified in the same form when *cmxmd* is terminated (*-h* option).

-i␣sec
: *sec* specifies the number of seconds per interval for periodic collection of statistics. For *sec*, specify a positive decimal number.
-i␣sec not specified: *sec* = 10 is assumed.

-n␣cnt
: For *cnt*, specify how many times *cmxmd* is to take statistics. A positive decimal number should be specified.
-n cnt not specified: *cmxmd* collects statistics until the end of the current day (24:00:00h).

-o␣file
: The statistics are to be written to the *file* file. All users must have write permission for the directory in which *file* is to be created.

  If *-o file* is not specified, *cmxmd* writes the statistics to the file *cmxm*[id].DD. Here, *id* is either the value specified for the CC/TSP access point with option *-c* or is blank, and DD is the day of the month (beginning with 1). With each call this file is written anew.

Files

cmxmd[*id*].*pid*
: Process ID of the executing CMX daemon.

/opt/SMAW/SMAWcmx/lib/cmx/cmxmd[*id*].*trc*
: Trace file for the CMX daemon.

/var/opt/SMAWcmx/tmp/cmxm[*id*].*DD*
: Statistics for the day of the month *DD* with *id* as per *c* option or blank.

10.9 Querying installed communication products (cmxprod)

You can use the *.cmxprod* command to query which communication products are installed on your system and whether these are complete. The command has the following syntax:

cmxprod [␣**-a**] [␣**-n**␣ boot-env |␣**-r**␣rootdir] [␣product ...]

-a

Output without header line and with letter P, p or c in the first column. P is followed by the product designation, p is followed by the package designation and c is followed by additional parameters of a package.

-n␣boot-env

Outputs the products installed in the boot environment *boot-env*.

-r␣rootdir

Outputs the products installed in the root directory *rootdir*.

product

Specifies the product name. The following values are possible (the prefix *CCP-* is optional):

```
CMX
CCP-OSI/NEA
CCP-ISDN-LINK
CCP-WAN-LINK
CS-GATE
```

One or more product names can be specified. If no product name is specified, *cmxprod* displays the information on all products installed in the local system.

For the *cmxprod* command, a product is considered installed if the package which gives the product its name is installed.

Example

```
[PGTR0046:root] cmxprod
CMX/CCP 6.0 Products and Packages:
CMX Communications Manager UNIX
        SMAWcmx 6.0A0004 Apr 24 2003 06:00
        SMAWcxagt 6.0A0004 Apr 24 2003 06:04
        SMAWxti 6.0A0004 Apr 24 2003 06:03
        SMAWntp 6.0A0004 Apr 24 2003 06:04
        SMAWr6 6.0A0004 Apr 24 2003 06:04
        SMAWcsr 6.0A0004 Apr 24 2003 06:03
        SMAWwca is not installed.
        SMAWPbase 1.004 Apr 04 2003 16:31
        SMAWPglib 1.2.1002 Feb 04 2003 10:17
        SMAWPgtk+ 1.2.1002 Feb 04 2003 10:19
        SMAWPethe 0.9.1103 Apr 04 2003 16:33
CCP-OSI/NEA STREAMS-based NEA (NEATE/NEAN) and ISO (TP02) Protocols
        SMAWnea 6.0A0004 Apr 24 2003 06:04
        SMAWtp02 6.0A0004 Apr 24 2003 06:04
CCP-WAN-LINK Network Access to X.21, V.24, X.25 WANs
        SMAWwan 6.0A0004 Apr 17 2003 06:03
CCP-ISDN-LINK Network Access to ISDN/S2m and ISDN/S0
        SMAWisdn 6.0A0004 Apr 17 2003 06:03
CS-GATE STREAMS-based Transport Gateway TGW
        SMAWtgw 6.0A0004 Apr 24 2003 06:05
[PGTR0046:root]
```

Errors

Any errors that occur are logged to standard error output.

See also

cmxinfo.

10.10 TSP-specific status information (cmxstat)

The *cmxstat* command returns status information on the various transport service providers in a uniform format. Not only the attached TSAPs but also the subnet connections attached to the TSPs and the existing connections can be listed.

The command has five variants.

Syntax

cmxstat␣**-t**␣ts-prov␣**-a**

cmxstat␣**-t**␣ts-prov␣**-s**

cmxstat␣**-t**␣ts-prov␣**-c**{␣**n** |␣**t**}

cmxstat␣**-t**␣ts-prov␣**-e**

cmxstat [␣**-t**␣ts-prov] [␣**-o**␣own-appl [␣**-d**]] [␣**-p**␣part-appl]

-t␣ts-prov

Name of the TSP, possible values for ts-prov:

```
nea
ntp
tp02
rfc1006
```

-a

(attached applications) Outputs information on locally attached TSAPs.

-s

(subnet) Outputs information on subnet connections.

-c{␣**n** |␣**t**}

(connection) Outputs information on connections. Argument *n* outputs the current subnet connections, argument *t* outputs the current transport connections.

-e

(error) Outputs error statistics.

-o␣own-appl[␣**-d**]

(own) Outputs information on the local application *own-appl*. If the local application is configured in TNS, its GLOBAL NAME must be specified, otherwise the T-selector together with the additional option *-d* must be specified for *own-appl*.

-p␣part-appl
(partner) Outputs information on the partner application *part-appl*. The GLOBAL NAME configured in the TNS must be specified for *part-appl*.

Output attached TSAPs (option -a)

Depending on the options, this variant lists the TSAPS currently attached to a TSP. Not only the local TSAP name but also the total number of connection requests received and the number of accepted requests and rejected requests are output for each application attached locally to a TSP (*t_attach* with T_PASSIVE). These statistics include only connection requests received by the application from a partner. The connections established actively by the application are not included.

The following example lists the TSAPs set up at the NEA transport provider:

```
#cmxstat -t nea -a

NEA TP: Locally attached TSAPs
T-Selector      NSAP        ConnInd    ConAcc    ConRej
A:nealokal      70/255       20          19        1
```

Output active subnet connections (option -s)

This command variant lists the active subnet connections for WAN TSPs. The socket addresses via which incoming calls are received are listed for RFC1006-TSP.

In addition to the local address of the subnet connection, the WAN TSPs output the passive connection requests from the subnet, the connections accepted and the connections rejected.

In the view of the null transport provider, the following sample subnet connections are attached:

```
#cmxstat -t ntp -s
Null TP:        Locally attached SNPAs
Subnet-Id      Addr               CC  Ln    CurrCon ConAcc    ConRej
X25-1 (0x11)   123                W2  L1          0        0         0
X25-2 (0x12)   321                W2  L33         0        0         0
X25-1 (0x11)   1                  W3  L1          0        0         0
X25-2 (0x12)    2                  W3  L2           0        0         0
```

The OSI transport system using TCP/IP lists the TCP sockets at which the RFC1006-TSP receives incoming connection requests:

```
#cmxstat -t rfc1006 -s
OSI TP over TCP/IP:     Listening TCP sockets
Port IP address
102 ::
```

By default, RFC1006-TSP occupies TCP port 102 for all IP interfaces regardless of whether an application is attached or not.

The display of subnet connections gives no information on the current status of the lines. You can obtain detailed information on the lines using the *bstv linkstat -bWx* command.

Output current connections (option -c)

This command variant lists the current transport connections (*-c t*) of a transport system. In multiplexing transport systems (OSI-TP Cl.2 and NEA), several transport connections are operated via a network connection; you can list the existing subnet network connections (*-c n*) for these transport connections separately.

The following example outputs the transport connections of the NEA-TSP. In the NEA architecture the transport layer builds on a connectionless network layer. It is not possible to determine the subnet connection via which the packets are routed.

```
#cmxstat -t nea -c t
NEA TP:        Current connections
Tref  Ini TPI-State Lref  Rref  Ncon  Local TSel   LocAddr     RemoteTSel     RemAddr
16     1  DATA      1     2     -1      A:nearemot   70/255     A:nealokal     71/255
19     1  DATA      2     3     -1      A:nearemot   70/255     A:nealokal     71/255
```

Output error statistics (option -e)

The option *-e* outputs the TSP-specific error list. This list contains the error situations registered by the transport service provider. These include sent/received interface signals, protocol errors and bottleneck situations when requesting system services such as kernel memory.

The error statistics for an NEA-TSP could look like this:

```
cmxstat -t nea -e
Collecting statistic data starts at: Thu Apr 15 06:07:27
2004
              M_ERROR sent:    0
          M_ERROR received:    2
  Interface error received:    0
      Interface error sent:    0
           Protocol errors:    0
 Memory allocation failure:    0
              Mblk failure:    0
           No free streams:    0
```

Output TSP resources of a CMX application (options -o, -p)

This variant outputs the resources of the transport system(s) that belong to a LOCAL NAME or to a partner address.

The option *-o* outputs for the specified local CMX application all resources bound in the transport system. These include the TSAPs assigned to the application and the connections ending in these TSAPs. If the local application is not configured in the TNS, you must specify the T-selector together with option *-d*. The T-selector must be specified in the following format:
type:printable form, e.g. *A:nonx2902*.

The option *-p* outputs all transport connections that belong to the specified partner application. If options *-o* and *-p* are specified at the same time, all connections between the local application and the partner application are listed.

10.11 Traces for the transport system (cmxtrc)

The *cmxtrc* command switches traces in transport systems on and off. The protocol layers on which traces can be used and the events to be recorded are dependent on the specific transport system and the command input options, in particular the GLOBAL NAME of the applications. If, for example, the route determined by the GLOBAL NAME of a partner application is assigned to exactly one WAN interface (controller and line), a line trace on the controller is switched on in addition to the trace in the transport provider (NEA, NTP or TP02). The line trace can subsequently be edited using the *ethereal* protocol analyzer.

The *cmxtrc* command switches the traces in the transport systems on using the basic mechanisms of the *comtr* command; in WAN transport systems, with the additional help of the line-specific controller trace. The trace files are stored in the current directory with the default name *comtr*, see page 291f, option *-f*.

Please note the following:

– No more than one transport system filter may be active at any one time on each transport system.

– Trace filters may be set up simultaneously in different TSPs. No more than one line trace may be active at any one time.

– *cmxtrc* switches off a trace switched on using the *comtr* command and saves the current contents of the trace buffer. The same applies for line-specific traces.

The command has three formats: switch on, information on traces, switch off.

Syntax

cmxtrc [␣**-c** |␣**-d**] [␣**act** |␣**pas**] -o own-appl [␣**-h**␣hostname] [␣**-t**␣ts-prov] [␣**-n**␣num]

cmxtrc [␣**-c** |␣**-d**] [␣**act** |␣**pas**] -p part-appl [␣**-o**␣own-appl] [␣**-h**␣hostname] [␣**-n**␣num]

cmxtrc [␣**-c** |␣**-d**] [␣**act** |␣**pas**] ␣**-h**␣hostname [␣**-t** rfc1006] [␣**-n**␣num]

cmxtrc [-␣**-c** |␣**-d**] [␣**act** |␣**pas**] ␣**-t**␣ts-prov ␣**-n**␣num

cmxtrc␣**-i** [␣**-t**␣ts-prov]

cmxtrc␣**-r** [␣**-t**␣ts-prov]

The individual variants are described below.

Switch on trace (options -c, -d)

-c [␣**act** |␣**pas**]
(connection) Logs connection setup but not the data transfer phase, default setting. The optional arguments *act* and *pas* let you differentiate between active and passive connection setup; both are logged by default.

-d [␣**act** |␣**pas**]
(data) Logs connection setup and data transfer phase. The data primitives and their length are logged but not their contents.

-o␣own-appl
(own) Logs the actions of the local CMX application with the GLOBAL NAME *own-appl*. If the assigned LOCAL NAME involves several TSPs, you must specify the transport system using the option *-t*. If the TSAP already exists when the command is entered, all newly set up connection endpoints of the TSAP satisfy the conditions of the trace filter.

-h␣hostname
(host) This option is relevant for RFC1006 only. Only the events assigned to the local IP interface with *hostname* are logged.

-t ␣ts-prov
(TSP) Defines the TSP. This entry is only necessary if the TSP is not already identified by the other entries. Possible values: nea, ntp, tp02, rfc1006

-p␣part-appl
(partner) Logs the actions to the partner with the GLOBAL NAME *part-appl*. If, in a WAN TSP, the connection is set up via a uniquely configured subnet connection, the corresponding line-specific trace is switched on at the same time.

-n␣num
(number) Specifies the number of connections to be logged. The trace filter is switched off after the number of connections indicated by *num* have been logged.

-v
(verbose) Controls the trace form. The option *-v* causes the trace entries to be written directly to stdout. By default, the trace events are saved in a trace buffer and edited after the trace is switched off.

Notes on use

– A trace filter applies only to connections that are established **after** the filter has been set up. Consequently the filter does not apply to connections between communication partners which satisfy the filter criteria but which already existed before the *cmxtrc* command was issued.

– The options *-o* and *-p* define TSAP-specific trace filters. They are therefore suitable for analyzing the communication problems of individual CMX applications.

– Trace documents for reproducible problems can also be defined by logical specification of the connections to be logged, e.g. the next connections to be set up.

Information on traces switched on (option -i)

The option *-i* informs you whether a trace filter is switched on in a transport system. The filter attributes are also displayed.

Example

cmxtrc -t ntp -n 2 logs the next two connections set up:

```
# cmxtrc -n 2 -t ntp
clearing ntp trace buffer
comtr daemon writes into the following 2 files alternatively:
/comtr.bin_ntp.01 and /comtr.bin_ntp.02
```

cmxtrc -i -t ntp then outputs the following information:

```
# cmxtrc -i -t ntp
        Status of NTP TSP:     READY.
TSP ntp: Trace-Filter active
         Number of connection setups:    2
         Filter attribute: TRC_EV_CONN (connection setups without data
transfer)
```

Switch off trace (option -r)

This option resets the trace selection criteria for the specified transport systems. The trace entries are read from the trace buffer and stored in the current directory with the default name *comtr*, see page 291f, option *-f*.

10.12 Changing limits for the CMX automaton (cmxtune)

You can use *cmxtune* command to change the limits of the CMX automaton for the maximum number of transport endpoints (TEPs), transport service access points (TSAPs), transport connection endpoints (TCEPs), and attachments of processes. *cmxtune* overwrites the preset values in the *CMXlimits* file with the new limits. The new values do not become effective until the system is rebooted.

The command has the following syntax:

cmxtune ␣ [**att** ␣ num] [␣ **tep** ␣ num] [␣ **tsap** ␣ num] [␣ **tcep** ␣ num]

num
: Limit for *att*, *tep*, *tsap* or *tcep*.

  Value range: 1024 to 65535 (theoretical limit).

att
: Setting maximum number of attachments via ICMX or XTI.

tep
: Setting maximum number of transport endpoints (TEPs).

tsap
: Setting maximum number of transport service access points (TSAPs).

tcep
: Setting maximum number of transport connection endpoints (TCEPs).

Files

/opt/SMAW/SMAWcmx/lib/cmx/CMXinit
: Init script for CMX.

/opt/SMAW/SMAWcmx/lib/cmx/CMXlimits
: File with limits for the CMX automaton.

10.13 Traces for CMX drivers (comtr)

The *comtr* command provides a uniform trace interface for all CMX components. You may produce and process traces for the following components:

– for the CMX automaton,
– for the TPI adapter,
– for the Transport Service Providers NEA, RFC1006, TP0/2 and NTP,
– for the Forwarding Support Service,
– for the Communication Service CS-ROUTE,
– for the connectionless WAN access,
– for the Routing Scheduler,
– for the PPP interface,
– for the STREAMS-based and HSI-based parts of the WAN adapter,
– for the Transport Gateway.

The command offers continuous tracing even for high trace volume. The output format is uniform for all CMX components, the command syntax isidentical.

Traces may be read and evaluated from a global or component-specific list. A trace can be started or stopped at any time, and its content can be output at any time.

A global trace is always activated at system startup and starts writing into the error list. The command *comtr␣-m␣glob␣-t* processes this list.

Syntax

comtr can only be called by the system administrator or CMX administrator. The command syntax to be used depends on the function required:

– Query information
– Start/monitor trace
– End/evaluate trace
– Evaluate traces

For some functions, the specification of a CMX component is required. Where applicable, add the parameter *-m␣module-id*.

-m␣module-id

Module identification. The following values are possible for *module-id:*

glob Global error list

cxauto

CMX automaton

tpia TPI adapter

nea TSP NEA

rfc1006
TSP RFC1006

tp02 TSP TP0/2

ntp TSP NTP

fss forwarding support service

ads access data service

clw connectionless WAN access

ppx interface of point-to-point protocol

rs routing scheduler

cdsx FSS interface (CC requests)

cws STREAMS-based component of CC-WAN adapter

cwp PCI-based component of CC-WAN adapter

tgw transport gateway

Querying information

comtr ␣{ **-a** | **-h** }

-a

The following information is provided for all components:

- module name
- short identifier
- trace level set
- degree to which trace buffer is used or overwrite marker *w*, if overwritten.
- size of the files that are alternately written to by a daemon process and their full path name.

  Output ends with a maximum of two transport references (TREF) and a maximum of two process IDs, provided that at least one of these selective trace criteria has been set.

Example:

```
COMTR -> INFORMATION ABOUT TRACE STATUS
Module           Id     Level  % full  File sizes (1/2)- file
                                                        names
______           __     _____  ______  ________________________
Global           glob   unsp.     31
CMX-Automat      cxauto  1        00
CMX module       cdsx    1        26
CMX module       cws     1        24/w
CMX module       rs      1        07
FSS-Service      fss     1        01
TPI-Adapter      tpia    1        00
NEA TSP          nea     1        00
NTP TSP          ntp     1        00
RFC1006 TSP      rfc1006 1        00
TP02 TSP         tp02    1        01
CS-ROUTE module ads      1        00
CS-ROUTE module clw      1        03
CS-ROUTE module ppx      1        00
CC-WAN-Adapter cwp       1        00
```

-h

The help option may be called globally in CMX or for a specific component. In the global variant, the default values for the trace level and the size of the kernel-internal trace buffer set when a trace is switched on are output if no differing values are specified.

Example of global variant

```
COMTR -> INFORMATION ABOUT SELECTABLE TRACE LEVELS AND SIZE OF
TRACE BUFFER
Module           Id      default  buffer size
______           __      _______  ___________
CMX-Automat      cxauto    4       320 Kbytes
CMX module       cdsx      4        28 Kbytes
CMX module       cws       3        66 Kbytes
CMX module       rs        4        40 Kbytes
FSS-Service      fss       2        23 Kbytes
TPI-Adapter      tpia      3       144 Kbytes
NEA TSP          nea       3       304 Kbytes
NTP TSP          ntp       2       248 Kbytes
RFC1006 TSP      rfc1006   2       108 Kbytes
TP02 TSP         tp02      3       206 Kbytes
CS-ROUTE module  ads       2        72 Kbytes
CS-ROUTE module  clw       3       206 Kbytes
CS-ROUTE module  ppx       1       112 Kbytes
CC-WAN-Adapter   cwp       4        48 Kbytes
```

In the component-specific variant, the help option indicates the trace level that can be set for the component. Some components offer DEBUG levels but these require detailed knowledge of the internal processes in the component.

Example of component-specific variant

```
# comtr -m ntp -h

level    meaning
-----    -------
  1      error level (always switched on)
  2      Connection primitives without any user data
  3      Connection primitives and user data in connection
         primitives, if present
  4 *    Connection primitives and data primitives
  5      Connection-, data primitives and user data
         (up to 100 bytes)
  6      Connection, data primitives and user data. The length and
         offset of the user data may differ from the default
         values(offs 0;len 100)
  8      Connection primitives for passive connection setup

* Default, if trace switched on and no level with the -sl flag
specified
```

Starting and monitoring the trace mechanism

The options described are used to start traces and monitor the trace mechanism. The following commands are permitted:

comtr␣-m␣module-id **-c**

comtr␣-m␣module-id**␣-sl**[␣level] [␣**-v**] [␣**-K␣**wrap] [␣**-f␣**file] [␣**-u␣**size] [␣**-p␣**pid1[␣pid2]] [␣**-T␣**tref1[␣tref2]] [␣**-x␣**type]

comtr␣-u␣size

comtr␣-p␣pid1[␣pid2]

comtr␣-T␣tref1[␣tref2]

comtr␣-r

comtr␣-x␣type

-c

deletes the contenf of a trace component buffer.

-sl level [**-v**]

starts trace, sets the trace level for the specified component and allocates a trace buffer. You can specify a trace level with *level,* if you do not, the default level will be used (see *comtr -h*). Bei Angabe der Zusatzoption *-v* werden die anfallenden Trace-Daten direkt nach *stdout* geschrieben.

-K wrap

The decimal number *wrap* specifies the maximum size of the binary trace file in kilobytes. If *wrap* = 0, there is no restriction for the size of the binary file.

-K wrap not specified: *wrap=1024* is assumed.

-f file

Instead of the default names, writing is performed alternately to *file*.01 and *file*.02. If no file names are specified, the system will assign the names comtr.bin_*modid*.01 and comtr.bin_*modid*.02.

-u size

defines a modified size of a trace buffer for a component. *size* specifies the size of the trace buffer in Kbytes.

Maximum size: 8096 kbytes.

-p pid1 pid2

limits the number of trace entries to 2 process IDs.

-T tref1 tref2

specifies a transport reference that a trace is to be created for.

-r

Deletes set selective trace criteria, i. e. process IDs and transport references.

-x type

Terminates the filling of a module-specific trace buffer, as soon as a specific entry (“TYPE”) is written to the buffer.

End and evalutate trace

Use the following options to terminate a trace mechanism that has been started under control of a background process. The background process will be stopped.

comtr -m module-id **-t** [**-f** file]

comtr␣-m␣glob␣-t[␣**-f**␣file] [␣**-b**␣binary-file]

-t
stops the trace and sets the trace level back to error level. The trace evaluation is written in the file *comtr.module-id.ascii*.

-f␣file
writes the trace evaluation in the file.

-b␣binary-file
outputs the global trace also in a binary file (only when *glob* is specified).

Prepare traces

comtr␣-e␣-f␣*file*
prepares trace information for output in ASCII format contained in a binary file *file*. The results are output to *stdout*.

Example: d.

Examples

comtr -e -f comtr.bin_tpia.01 > tpia.rea
evaluates the trace file contained in *comtr.bin_tpia.01* and outputs the results to the file *tpia.read*.

comtr -m nea -sl5
starts the trace for the NEA-TSP with level 5. The trace information is written alternately to the files *comtr.bin_nea.01* and *comtr.bin_nea.02*.

comtr -p 1812 1814
restricts the filling of the trace buffer to entries for the process IDs 1812 and 1814.

10.14 Protocol traces with ethereal

The software protocol analyser *ethereal* is supplied with CMXV.6.0 as freeware. It logs the communication traffic on a LAN interface and can edit these log elements in symbolic form. Preparation includes the convergence protocol RFC 1006 so that communication traffic between CMX applications and a partner application via this TSP can be monitored and displayed. *ethereal* offers a number of filtering and display possibilities for communication traffic. These are described in detail in the protocol analyser documentation which you can access on the *ethereal* website at: *http://www.ethereal.com.*

10.15 Controlling and editing NEABX library trace (neal)

The trace mechanism of the NEABX library is activated and controlled via the environment variable NEATRACE. The trace entries of a process are collected in compressed, binary format in a dynamically created buffer and periodically saved in temporary files. These files are edited in a separate step using *neal*.

Controlling the trace mechanism - NEATRACE

Every *t_attach* CMX call issued by a process evaluates the environment variable NEATRACE and, when appropriate, activates the trace mechanism. NEATRACE must have been set before the application is started, i.e. prior to the first *t_attach* of the process to be monitored.

After activation of the trace mechanism, the temporary file *NEALpid* with process ID *pid* is opened if it is not already open. Access permissions 0600 are granted for the files. Storage is then dynamically reserved for buffering the trace entries.

Storage and trace files are reserved for the lifetime of the process.

The options specified in NEATRACE control the trace mechanism. Options *s* and *S* determine the extent of logging. Options *-p*, *-d* and *-r* control buffering, data length and cyclical overwriting of the files.

The variable NEATRACE is specified in the following syntax:

NEATRACE= ␣{ **-s** | **S** } [␣**-p**␣fac] [␣**-d**␣length␣**-r**␣wrap] [␣**-f**␣file]

export␣**NEATRACE**

-s and *-S* determine the type of tracing. Only one of the two values may be specified. To activate the trace mechanism one value must be specified.

-s

Normal logging: All calls and their arguments are logged. Options and user data are not logged.

-S

Detailed logging: All calls, their arguments, the contents of the options and the user data are logged.

-p␣fac

The decimal digit *fac* determines the buffering factor. The amount of buffering is *fac* * 1024.

If *fac=0* is specified, every trace entry is written immediately to the file (unbuffered).

Value range for *fac:* 0...8.

-p␣fac not specified: *fac=0* is assumed.

-d␣length

The decimal number *length* specifies in bytes the maximum TIDU length that can be logged when the control option *-S* is specified. The default value is 16.

length=0...16...65535.

If 0 is specified, no data is logged; otherwise, a number up to the maximum length is logged.

-r␣wrap

The decimal number *wrap* specifies that after *wrap* * 1024 a log is to be produced in the second temporary file *NEAMapid*.

The trace mechanism handles the second file *NEAMapid* in exactly the same way as *NEALapid*.

After every *wrap* * 1024 bytes, the trace mechanism switches between *NEALapid* and *NEAMapid*. The former contents of the file are then lost.

-r␣wrap not specified: *wrap=256* is assumed.

-f␣file

This option is used to specify a directory *file* where the trace files *NEA[LM]apid* are to be stored. In this regard, the argument can be both a relative and an absolute path name.

Editing the trace information - neal

neal reads the entries generated by the trace mechanism from the files you have specified for *file*, processes them in accordance with the specified options and outputs the result to *stdout*. The command has the following syntax:

neal [␣**-c**] [␣**-d**] [␣**-e**] [␣**-v**] [␣**-x**] [␣**-D**] [␣**-p**]␣file ...

The options selected specify which trace entries are to be edited. More than one of the values described in the following may be specified per *neal* call. Only options *-v* and *-x* are mutually exclusive. If no option is specified, *-cdex* is assumed.

-c

Editing is performed for the function calls:

– for attaching/detaching the TS application to/from NEABX
– for connection setup and disconnection

-d

Editing is performed for the function calls:

– for data exchange
– for flow control

-e

Editing is performed for the command calls for event handling.

-v

Detailed editing is performed for the command calls, their arguments, the options and user data. The extent of editing of the data depends on whether option *-s* or option *-S* was specified in NEATRACE.

-x

Limited editing is performed for the command calls and their arguments excluding options and user data.

-D

If the option *-D* was specified during trace control, the internal debugging information can be evaluated with *neal*.

-p

If option *-p* is set, for the *file* parameter you must specify the process ID (pid) of a TS application's process for which the trace entries are to be edited.

file ...

You must specify the name of one or more files that contain the trace information to be edited. If option *-p* was one of the options specified, for *file* enter only the process ID of the process for which trace entries are to be edited. *neal* then searches in directory */var/opt/SMAWcmx/tmp* for all trace files associated with this process.

Output formats

The format of the trace information edited using *neal* is described in section „Controlling and editing the CMX library trace (cmxl)“ on page 255.

Files

NEAL.apid, NEAMapid

Files with compressed trace entries in binary format.

Unless otherwise specified for NEATRACE, the files for the NEABX library trace *NEALapid* and *NEAMapid* are stored in the */var/opt/SMAWcmx/tmp* directory.

See also

cmxl.

10.16 Starting and stopping CMX and TSPs (StartStop)

StartStop is a set of commands that can be used to start, restart, control behavior at startup or stop the components of CMX (transport service provider NTP and RFC1006, FSS) and the installed TSPs (NEA, TP0/2).

Once the communication software has been installed, the (newly) installed components can be started manually using these commands.

The commands include all tasks to be performed in succession when starting or stopping the components. In addition, the scripts check the prerequisites, current status, and dependencies of the pending operations and thereby effectively ensure the consistency of the system. They log the process and result of the operations in log files for subsequent diagnosis.

The commands are output to standard output or standard error output.

These StartStop scripts are very helpful for handling CMX components and TSPs. System administrator authorization is required to execute these scripts. They can be called via the CMX menu or from the command level using the following commands:

Syntax

cmx [␣{ **autostart** | **start** | **restart** | **stop** | **boot** | **shutdown** | **autostop** | **diag** }]

cmxsnmp [␣{ **autostart** | **start** | **restart** | **stop** | **autostop** | **diag** }]

ntp [␣{ **autostart** | **start** | **restart** | **stop** | **boot** | **shutdown** | **autostop** | **diag** }]

nea [␣{ **autostart** | **start** | **restart** | **stop** | **boot** | **shutdown** | **autostop** | **diag** }]

tp02 [␣{ **autostart** | **start** | **restart** | **stop** | **boot** | **shutdown** | **autostop** | **diag** }]

rfc1006 [␣{ **autostart** | **start** | **restart** | **stop** | **boot** | **shutdown** | **autostop** | **diag** }]

csr [␣{ **autostart** | **start** | **stop** | **boot** | **shutdown** | **autostop** }]

autostart

inserts start routines in the system start files. These routines are then executed at every system startup and start the appropriate component.

start

starts the component following various checks. A message is output to standard output if this module is already started. The result of this operation corresponds to that of an *autostart* when the system is powered up.

Based on entries in the *crontab* file, *start* also initiates the regular execution of specified actions. These check whether the component is still functioning and restart it if it is not. These actions are executed until the component is explicitly terminated with *stop* or a restart fails three times in succession. In both cases, execution of the actions is halted and a corresponding message is written to the log file.

restart

Restarts the component. This operation has the same effect as executing *stop* and *start* in immediate succession.

stop

Stops the component. If a component is deactivated, its reserved system resources are freed. Regular actions called from the *crontab* file of the system are likewise terminated. This status remains until the next start or autostart.

boot

Restarts the newly installed or updated components after installation of the software.

The *cmx boot command* starts CMX and all product components that build on CMX. After installation/update of a product component building on CMX, e.g. TP0/2-TSP, the component is started by entering *tp02 boot* without impairing the other transport systems already in operation.

shutdown

Stops the specified component or, in the case of *cmx*, the entire CMX/CCP software and establishes the prerequisites for updating the installed software.

cmx shutdown is executed only if no applications are attached to CMX.

autostop

Removes the component-specific routines inserted by *autostart* in the system start files from these files. This means that the component is no longer loaded automatically at the system start.

diag

Outputs log files for the respective component.

Files

In the following file names, *$Name* must be replaced by one of the following component names: CMX, FSS, NEA, NTP, TP02, RFC1006, cmxsnmp.

/var/opt/SMAWcmx/adm/log/$Name.log

Log file of the respective component.

/var/opt/SMAWcmx/adm/log/CCP.log

Log file for restarting a CCP.

/etc/rc0.d/K[0-9] [0-9] $Name

Stop script for corresponding component.

/etc/rc2.d $Name

Start script for corresponding component.

10.17 Checking the TS directory (tnsxchk)

The command *tnsxchk* searches for errors in the specified TS directory. *tnsxchk* first checks the validity of the pointers and indices in the files of the TS directory DIR*num* (*num* = 1...9) being checked. If it finds no errors, *tnsxchk* checks the format of the entries in the file. *tnsxchk* states whether or not any of the files contains errors. On request, *tnsxchk* supplies detailed information on the errors it has found for diagnostic purposes. *tnsxchk* can detect that a TS directory does not have the structure expected by the TNSX daemon. The command has the following syntax:

tnsxchk [␣**-d**␣num] [␣**-v**] [␣**-f**]

-d␣num

num specifies the number of the TS directory DIR*num* to be checked.

If no value is specified, *num = 1* is set.

-v

tnsxchk checks whether TS directory DIR<num> has the version that the TNSX daemon can use.

If *-v* is not specified, *tnsxchk* checks all files in the TS directory.

-f

tnsxchk supplies detailed information on the errors found. This information is very complex and is only needed for diagnosis by the customer service department.

Output format

The output of *tnsxchk* is preceded by a header line containing the program name, the name of the TS directory checked, the date and the time the program started.

The contents of the VERSION file for the TS directory are then output. The values in the VERSION file depend on the operating system. Below is an example of output for the *tnsxchk-v* command:

```
VERSION:
========
VERSION=V6.0 BYTEORDER=MSB LONG=32BIT INTEGER=32BIT SHORT=16BIT
```

tnsxchk outputs the following two lines for every file in the TS directory checked by *tnsxchk*:

```
tnsxchk: checking file:
tnsxchk: result
```

The absolute pathname of the file just checked is specified for *file*. “OK” or “error” is output for *result*.

Files

DIR1 ... DIR9

TS directory, numbered in ascending order, maximum 9.

The TS directories *DIR1* to *DIR9* are located under the directory */opt/SMAW/SMAWcmx/lib/cmx*. They contain the following files (i = 1,...,5):

NAMEPi
NPVALUES
PROPERTIES
PRVALUE
ROOT
FREENPV
FREENPRV
VERSION

LOG

The *LOG* file for en bloc modification of TNS entries is stored in the */var/opt/SMAW/cmx/tmp* directory.

tnsxd.trc

The *tnsxd.trc* file for logging TNS accesses is located in the directory //*opt/SMAW/SMAWcmx/lib/cmx*. This directory also contains the tnsxd.pid file, in which the current process number of the active *tnsxd* process is stored.

See also

tnsxd

10.18 TS directory: create, update, output (tnsxcom)

tnsxcom processes entries in tnsxfrm format (i.e. the standard TNS entry format). The *mode* parameter can be used to select the TNS compiler's operating mode.

It is possible to specify the following operating modes:

UPDATE
: Update the TS directory in accordance with the entries in the *file* file (possibly more than one *file* file ...).

INTERAKTIV
: Update the TS directory according to specifications entered through stdin.

LOAD
: Load a previously empty TS directory with the entries contained in the *file* file (possibly more than one *file* file...).

DUMP
: (Sorted) output of the contents of a TS directory to the *file* file in *tnsxfrm* format.

CHECK
: Check the syntax of the *file* file.

CHECK_UPD
: Check the syntax of the *file* file and, if the syntax is correct, update the TS directory in accordance with the entries in *file*.

If no options are specified, *tnsxcom* compiles the entries from the file into the TS directory in UPDATE mode (a line at a time).

tnsxcom in UPDATE, INTERACTIVE, LOAD and CHECK_UPD modes (*mode = -u, -i, -l, -S*) may only be invoked by the system administrator (root authorization required).

As soon as *tnsxcom* has processed the *file* file, the TNS compiler outputs the following values to stderr.

– the number of errors and warnings that occurred
– the time needed to process the *file* file (real)
– the proportion of real required by the *tnsxcom* process (in %)
– the time required, divided into user mode (user) and system mode (sys)

If *tnsxcom* processes more than one file (*file ...*), at the end of processing *tnsxcom* outputs these values for the entire compilation run.

Syntax

tnsxcom [␣**-d**␣num] [␣mode] [␣**-p**␣prmt] [␣**-t**] [␣**-o**␣orig] [␣file ...]

-d␣num

Number of the TS directory to be processed; possible values for num: 1,2,...,9

If no value is specified, *num = 1* is set.

mode

Define operating mode of *tnsxcom*.

Possible values are *-D*, *-i*, *-l*, *-s*, *-S*, *-u*, they are mutually exclusive. If no value is specified, *-u* is set. The specifications for *mode* have the following meaning:

-D

DUMP mode

tnsxcom edits the contents of the TS directory in the form of entries in *tnsxfrm* format in the *file* file; this, in turn, can also be used as input. Output is sorted in ascending ASCII order by GLOBAL NAME according to name hierarchy and for each name part. In other words, each name part[1] is sorted first. If there are several TS applications in the TS directory that have the same name part [1], they are sorted according to name part[2], and so on.

-i

INTERACTIVE mode

tnsxcom reads entries from stdin in *tnsxfrm* format once it has indicated its readiness to receive input by issuing a prompt character sequence (see *-p prmt*). It then merges the entries into the TS directory by defining the entries that did not previously exist in the TS directory and updating those that did.

-l

LOAD mode

tnsxcom reads in the entries from the editable *file* file one at a time and loads the (previously empty) TS directory with the syntactically correct entries.

-s

CHECK mode

tnsxcom only uses the syntax check on the *file* file and logs any syntax errors. The TS directory is not changed.

-S

CHECK_UPD mode

As in option s, the syntax of the entire *file* file is checked in an initial run. If there are no errors in *file*, *tnsxcom* updates the TS directory in a second run.

-u

UPDATE mode

tnsxcom reads in the entries from the editable *file* file one at a time and merges the syntactically correct entries into the TS directory by creating the entries that did not exist previously and updating those that did (option *u* is the default value for *option*).

-p␣prmt

-p prmt causes the string *prmt* to be used as a prompt in interactive mode *-i*, *-p* is ignored in other modes.

-p prmt not specified: *prmt* = * is assumed.

-t

Switch on *tnsxcom* trace.

The trace is switched on and its output logged in the file *tnsxcom.trc* in the current directory. If *mode* = *-D* is specified, option *t* will be ignored.

-o␣orig

orig specifies the origin.

The origin is a sequence of name parts. The contents of the *name* field of a *tnsxfrm* entry has a . (period) and the value of *orig* added to it, provided the contents do not end in a . (period). The result must be a syntactically correct GLOBAL NAME with a maximum of five name parts.

file ...

Name of the file with entries in *tnsxfrm* format that are to be interpreted by *tnsxcom* for *mode* = *-l*, *-s*, *-S* or *-u*. It is possible to specify more than one file.

For *mode* = *-D*, specify the name of the file in which *tnsxcom* is to edit the contents of the TS directory.

Errors

tnsxcom outputs error messages relating to the syntax or semantics of the entries in *file* along with the name of the file and the line number to *stderr*.

Files

tnsxcom.trc
: Name of the trace file in the directory in which *tnsxcom* was called.

LOG
: The LOG file for en bloc modification of TNS entries is stored in the */var/opt/SMAW/cmx/tmp* directory.

tnsxd.trc
: The *tnsxd.trc* file for logging TNS accesses is located in the directory */opt/SMAW/SMAWcmx/lib/cmx*. This directory also contains the *tnsxd.pid* file, in which the current process number of the active *tnsxd* process is stored.

DIR1 ... DIR9
: TS directory, numbered in ascending order, maximum 9.

  The TS directories *DIR1* to *DIR9* are located under the directory */opt/SMAW/SMAWcmx/lib/cmx*.

See also

tnsxd, *tnsxprop*, as well as the description of the input format for TNSXCOM in section “Syntax of the TNS configuration file” on page 77.

Example

The call t*nsxcom -d 2 -l -o np4..np2.np1 input1 input2* transfers the entries from the files *input1* and *input2* to the empty TS directory 2; the contents of each name field that does not end in ‘.’ are to be expanded in the form "*contents.np4..np2.np1*" (with name part 3 blank).

10.19 Deleting TNS entries (tnsxdel)

You can use *tnsxdel* to delete individual entries from the TS directory. In addition to specific GLOBAL NAMES, you can also delete entire hierarchies of names by leaving blank the name parts that are different and only specifying the identical name parts when entering the command.

tnsxdel [␣**-d**␣num] [␣**-a**] [␣**-v**] [␣**-f**␣file] [␣name...]

The options of the command have the following meaning.

-d␣num
: The number of the desired TS directory is defined with *num*.
Value range: 1-9. Default value: 1

-a
: This option deletes all entries and the directory itself.

-v
: Sets the command mode to “verbose”, i.e. a line with the object name is written to *stdout* for each remote object. In combination with option *-a*, only one line with the name of the remote TS directory is created.

-f␣file
: Here you specify an input file *file*. This contains all GLOBAL NAMES or name parts that are to be deleted.

name
: Here you specify the GLOBAL NAME to be deleted.

Examples

A TS directory DIR5 contains the following entries (output is obtained with tnsxprop -d 5):

```
Gilbert.sales.dep1  \
     TA  LOOPSBKA A´Gilbert´
Meier.sales.dep1 \
     TA  LOOPSBKA A´Meier´
Ruth.sales.dep1 \
     TA  LOOPSBKA A´Ruth´
Schulz.purchase.dep2 \
     TA RFC1006 139.22.96.29 A´HUGO´
Huber.warehouse.dep2 \
     TA RFC1006 139.22.96.32 A´EGON´
Kruse.warehouse.dep2 \
     TA RFC1006 139.22.96.38 A´EMIL´
```

Your input file *input*, which contains the elements to be deleted, could contain the following entries.

Deletion of Gilbert (not in dep1):

```
Gilbert.sales.dep2
```

Deletion of Meier in the file:

```
Meier.sales.dep1
```

Deletion of entire hierarchy incl. Huber and Krause:

```
warehouse.dep2
```

The subsequent call of *tnsxcom -d 5 -v -f input one.more name* would create the following output:

```
entry Gilbert.sales.dep2 not existing
entry Meier.sales.dep1 removed
entry Kruse.warehouse.dep2 removed
entry Huber.warehouse.dep2 removed
entry one.more name not existing
```

A subsequent call of tnsxcom -d 5 produces:

```
Gilbert.sales.dep1  \
     TA  LOOPSBKA A´Gilbert´
Ruth.sales.dep1 \
     TA  LOOPSBKA A´Ruth´
Schulz.purchase.dep2 \
   TA  RFC1006 139.22.96.29 A´HUGO´
```

You can achieve considerably shorter runtimes when deleting TNS entries if you delete in reverse insertion order. In this case, you should initially insert the entries in sorted order when creating large TS directories using *tnsxcom*. Before deleting using *tnsxdel*, you can then sort the input file in reverse order using the *sort -r* command.

Example

The *tnsxdel.input* file contains the GLOBAL NAMES to be deleted. The entries are sorted in reverse order and then deleted from directory 1:

```
sort –r tnsxdel.input > tnsxdel.input.sort
tnsxdel –f tnsxdel.input.sort
```

Errors

Error messages are self-explanatory; error codes (decimal number) can be decoded with the *cmxdec* command. If specified object names are not contained in the TS directory, this is not interpreted as an error. The same applies if an non-existent TS directory is specified.

See also

tnsxcom, *tnsxprop*, *tnsxt*, *cmxdec*.

10.20 Displaying information on the TS directory (tnsxinfo)

tnsxinfo outputs the contents, the limits and the current contents of a TS directory to standard output *stdout* in readable form. *tnsxinfo* obtains its information through direct access to the respective TS directory, i.e. without making a request to the TSN daemon.

tnsxinfo furnishes statistical data on the TS directories, edits the GLOBAL NAMES of the TS application in the TS directory concerned into a tree structure, and outputs the properties assigned to the TS applications.

Syntax

tnsxinfo [␣**-d**␣num] [␣**-g**] [␣**-p**] [␣**-v**]

-d␣num
: Defines *num* as the number of the TS directory to be used.

-g
: *tnsxinfo* outputs the contents of the VERSION file in the TS directory, the utilization limits, and information about the current contents of the TS directory. All GLOBAL NAMES in the TS directory are output in a tree structure.

-p
: *tnsxinfo* outputs the contents of the VERSION file in the TS directory, the utilization limits and information about the current TS directory. Additionally, all the properties in the TS directory are output in edited form.

-v
: *tnsxinfo* only outputs the contents of the VERSION file of the TS directory.

If none of options *-g*, *-p* oder *-v* is specified, *tnsxinfo* outputs the contents of the VERSION file and tables with the TS directory limits and information about its current contents to *stdout*.

Output format

The tnsxinfo output always begins with two header lines and the contents of the VERSION file in the TS directory.

The header lines contain:

– name and version of the program
– current date and time of the program start
– name of the TS directory to which the output refers

Output of limits and current assignment

The limits and current assignment of the TS directory are output in the form of three tables. The information in the tables may be incomplete if the TNS daemon is running at the time the information is queried (e.g. if data is being modified by the TNS daemon at the time of the query or the TNS daemon is storing data in a cache for faster access). The first table contains details of the name parts of the GLOBAL NAMES. The table has the following structure:

```
LIMITS FOR NAME PARTS AND PROPERTIES:
NAme part  TS_COUNTRY TS_ADMD TS_PRMD  TS_OU  TS_PN
index             1       2       3      4      5
length            2      16      16     10     30
maximum         113     223    1093   4093  16381
limit           100     200    1000   4000  16000
filled            1       2       2      2      5
```

The first line in the table contains the symbolic designations of the name parts of the GLOBAL NAME. Below this is the index of each name part; index 1 belongs to name part1, etc. The entries in the next lines have the following meaning:

length
: Maximum length permitted for the name part values in bytes.

maximum
: Size of the hash tables in the individual files NAMEP*i* (*i* = 1,...,5).

limit
: Maximum number of name parts with index *i* that you can enter in the TS directory.

filled
: Number of name parts with index *i* that currently exist in the TS directory.

The second table contains the limits and current assignment of the properties, name part values and property values (corresponding to the maximum length and current length of files NPVALUES and PRVALUES).

```
                Properties   Name part values      Property values
Maximum              320000             560700             21406500
filled                   14                 54                  134
gaps (max.)             -/-            0 (100)              0 (50)
```

The entries have the following meaning:

maximum
: Maximum number of properties you can store in the TS directory, or maximum length of the file NPVALUES, which contains all the name part values of the GLOBAL NAMES, and the file PRVALUES, which contains all the property values. The lengths are given in bytes.

filles
: Number of properties stored in the TS directory or current length of the files NPVALUES and PRVALUES in bytes.

gaps (max.)
: Number of gaps in the files PROPERTIES (contains the names of the properties), NPVALUES and PRVALUES resulting from deletions. When the number of gaps exceeds the maximum value given in brackets, the TS directory is reorganized and the gaps are closed.

The third table contains the maximum permitted length of the various property values in bytes.

```
properties:
type           length(bytes) type             length(bytes)
TS_EMPTYPROP               0  TS_TRANS               200
TS_NEABX                   1  TS_TRSYS                 1
TS_LNAME                 200  TS_USER1PR             200
TS_USER2PR               200  TS_USER3PR             200
```

The property designations have the following meaning:

```
TS_EMPTYPROP   Blank entry
TS_LNAME       LOCAL NAME,
TS_USER2PR     USER2,
TS_TRANS       TRANSPORT ADDRESS,
TS_TRSYS       TRANSPORT SYSTEM,
TS_USER1PR     USER1,
TS_USER3PR     USER3.
```

The properties are described in section “Configuring with tnsxcom” on page 75.

Output of GLOBAL NAMES

The GLOBAL NAMES stored in the TS directory are output in the form of a naming tree (see section “GLOBAL NAME” on page 77 for an example of a naming tree). Only part of the information supplied by tnsxinfo in the example is printed here.

```
GLOBAL NAMES:
     ----i-[Npi] [x] NonLeaf, Name part i, Value Npi, Index x
     ====i=[Npi] [x] [p] Leaf, Name part i, Value Npi, Index x,
Propindex p

     ROOT
     |----1-[co] [53]
     |       .|----2-[admd1] [82]
     |.       |.      .|----3-[prmd1] [461]
     |      . |     .  |      . |----4-[ou1] [873]
     |        |      . |        |       .|====5=[pn4] [875] [10]
     .        .        .        .
       :
       :
```

Every single dashed line ---i- leads to a node in the naming tree. Every double dashed line ===i= leads to a leaf in the naming tree. Every leaf is assigned uniquely to one TS application. The path in the naming tree from the ROOT to the leaf is the GLOBAL NAME of the TS application. Properties are only assigned to leaves.

The values given have the following meaning:

i
: Index of designated name part.

[Npi]
: Value of designated name part.

[x]

Hash table index of designated name part in file NAMEPi. The value is between 1 and the maximum for the hash table given in the table.

[p]

Index of the first property of the TS application in the table in file PROPERTIES. This value is only output if you have also instructed *tnsxinfo* to output the properties.

Output of properties

The properties of all TS applications in the TS directory are edited as follows (only part of the information supplied by tnsxinfo is printed in the example):

```
PROPERTIES: 14 entries
  #  xfnl next [t property    lng   off] [np# ind]
  0: XF..    1 [I TS_...        1     0] [ 5  864]
          0  00                                   .
  1: X.N.    2 [I TS_TRANS     20     1] [ 5  864]
          0  02001400 01004000  0a005bc4 c9c1d3d6
         10  c7401208                            .
  2: X..L   -1 [I TS_TRSYS      1    15] [ 5  864]
          0  00                                  .
  3: XF..    4 [I TS_...        1    16] [ 5  868]
          0  00                                  .
  4: X.N.    5 [I TS_TRANS     37    17] [ 5  868]
          0  02002500 02001000  1b00300f 03490000 .
         10  00090001 08001410  1996fe80 08d3c1d5 .
         20  c1d5e640 40                         ..
  5: X.N.    6 [I TS_TRSYS      1    3c] [ 5  868]
          0  02                                   .
```

In the first line of the table, tnsxinfo states how many properties are in the TS directory. Several lines are output for each of these properties. The first line contains the values of the variables given in the table's header line, the following lines contain the value of the property in hexadecimal format.

The header line variables have the following meaning:

#

Index of the property in the PROPERTIES file in the TS directory.

xfnl

States whether this is the first or last property of the TS application in the list. Possible values for xfnl are:

X

Entry is assigned.

F

First entry for a property of the TS application.

L
: Last property entry for this TS application.

N
: Next entry for a property of the TS application.

The values F, L, N are mutually exclusive.

next
: Index # of the next property associated with the same TS application.

t
: Type of property. Currently only the value I for TS_ITEM is output.

property
: Designation of the property. The designations used for output of the permitted property value lengths are output (see above).

lng
: Length of the property value in bytes.

off
: Offset in the TS directory file PRVALUES which contains the property values. Offset gives the distance from the start of the file to the beginning of this property value in bytes.

np#
: Index name part to which the property is assigned (np# = 1,...,5).

ind
: Hash table index of this name part in file NAMEPnp#.

Errors

If the TS directory specified in *-d num* has an incorrect VERSION file, *tnsxinfo* outputs the contents of the VERSION file and the values expected in VERSION together with an error message.

tnsxinfo also outputs an error message if it cannot open the VERSION file of a TS directory and the ROOT file has the wrong length. Both indicate a TS directory that was generated with a previous version of CMX and was not converted as described in section “Migration” on page 98.

Files

DIR1 ... DIR9
: TS directory, numbered in ascending order.

10.21 Locking access to the TNS daemon (tnsxlock)

tnsxlock locks and unlocks accesses to the TNS daemon *tnsxd*. As long as the lock is set, *tnsxd* will reject all access attempts with the error message (type, class, value) = (TS_TEMPERR, TS_INTERR, TS_NORQ). This means that for the time being *tnsxd* is not processing any requests (see <*tnsx.h*>).

tnsxlock can only be called by the system administrator (root authorization required).

Syntax

tnsxlock ␣{ **on** | **off** }

on
Set lock

off
Release lock

When a lock is set, it remains effective until it is released or until tnsxd is started up again.

See also

tnsxd, *tnsxt*

10.22 Outputting properties of TS applications in a TS directory (tnsxprop)

tnsxprop outputs to *stdout* in a printable format the values of all the properties that are contained in a TS directory for the TS applications specified.

It is possible to use the *option* parameter to determine the format in which the properties are to be output.

The TS applications are determined by the parameter values specified for *name*. It is also possible for the parameter values for *name* to be passed to *tnsxprop* from the *file* file. If no value is specified for either *name* or *file*, *tnsxprop* will edit the properties of all the TS applications in the TS directory in the specified format.

Syntax

tnsxprop [␣**-d**␣num] [␣{ **-h** | **-S** }] [␣**-f**␣file] [␣name ...]

-d␣num

Number of the TS directory to be used.

Possible values of *num*: 1,2,...,9

If no value is specified, *num* = 1 is set.

-h | **-S**

This parameter can be used to determine the format in which the properties are to be output.

If no value is specified, *-S* is set.

The options *-h* and *-S* described below are mutually exclusive. They have the following meaning:

-h

The properties are edited in hexadecimal representation.

The property values are output in the form of a hexdigit string along with the corresponding bit representation, where the least-significant bit is in the right-most position.

-S The properties are edited in symbolic representation in the format *tnsxfrm*. See also section “Syntax of the TNS configuration file” on page 77.

-f␣file

For *file*, specify the name of a file containing the GLOBAL NAMES of the TS applications whose properties are queried. The GLOBAL NAMES must be specified as described under *name*.

name

For *name*, specify the GLOBAL NAME of the TS application in the TS directory as follows:

NP5.NP4.NP3.NP2.NP1. NPi stands for the name parts of the GLOBAL NAME. NP5, for instance, stands for name part[5], i.e. the name part at the lowest level in the hierarchy. NP1 is thus the name part[1], i.e. the name part that is highest in the hierarchy. The name parts are to be specified from left to right in ascending hierarchical order.

If one of the name parts of a GLOBAL NAME has not been assigned (e.g. NP4) but that name part is, nonetheless, followed by a name part higher up in the hierarchy (e.g. NP3), then the separator character (.) of the non-assigned name part must be included.

If there is a sequence of separator characters at the end of a particular value for *name*, they may be omitted.

If the name parts contain special characters whose special meaning could cause an ambiguity of syntax, then these special characters must be escaped with a \ (backslash). In cases of doubt, you should always escape all special characters. If the escape is superfluous, it is ignored by *tnsxprop* anyway.

If the value * (asterisk) is specified for a name part, *tnsxprop* supplies the properties of all the TS applications which can have any value for this particular name part and which match the value specified in *name* for their other name parts (filter mode TS_RESTRICTED).

Files

DIR1 ... DIR9

TS-Directory, aufsteigend nummeriert, maximal 9.

Example 1

The TS application in TS directory 1, which has only name part[5] with the value *Example_1*, has a number of properties assigned to it. These properties are to be output to *stdout* in hexadecimal representation. Please insert the following command:

tnsxprop -d 1 -h Example_1

Example 2

For the TS application *Example_1* in TS directory 1, the properties are to be output to *stdout* in symbolic representation (default value of *option*). Please insert the following command:

tnsxprop -d 1 Example_1

The calls *tnsxprop -S > DAT1* and *tnsxcom -D DAT1* are equivalent. Both write the contents of TS directory 1 to file *DAT1* in symbolic representation.

10.23 Starting and stopping TNS trace (tnsxt)

tnsxt can be called by any user. With *tnsxt*, the trace of accesses to the TNS can be stopped and restarted at any time. *tnsxt* collects the last accesses to the TNS in printable representation in a ring buffer for as long as the trace is switched on. The ring buffer is set up in the file */usr/lib/cmx/tnsxd.trc* and remains in existence for the entire life of the system. The trace can be started at any time but the start must take place before the accesses that are to be monitored. Once the trace has been stopped, it is possible to obtain an output of the ring buffer. If the trace is switched on again, the ring buffer continues to be written at the point it was broken off.

Syntax

tnsxt ␣{ **on** | **off** }

on
The trace is (re)started.

off
The trace is stopped.

File

/opt/SMAW/SMAWcmx/lib/cmx/tnsxd.trc
File with trace entries.

See also

tnsxd

11 SNMP subagent for CMX

This chapter contains information on the functionality and operation of the CMX agent.

The section “Functions of the CMX agent” on page 326 describes in particular the embedding of the CMX agent into the SNMP network management concept, as well as the groups and object classes of the CMX MIB.

The section “Running the CMX agent” on page 349 is aimed at the system administrator of the local UNIX system and outlines the necessary tasks involved in installing the CMX agent on the local UNIX system and the options for local administration.

11.1 Overview of the CMX agent

The EMANATE subagent of the CMX product is an SNMP agent for the communication products on UNIX systems. In this manual it is referred to as the CMX agent for short. The communication products referred to are CMX, CCP (Communication Control Program) and XTI (X/OPEN Transport Interface).

The CMX agent supports the management of CMX and transport systems implemented by means of the CCP products on the basis of SNMP. The CMX agent is installed on your local UNIX system. This management application enables the network administrator of the management station to request information directly from the CMX agent. Most of the CMX- and CCP-specific configuration and administration data managed locally by CMX can be requested from the CMX agent by the management station, and in some cases modified. Among the objects that can be managed are the communication controllers (CCs), the Transport Service Providers (TSPs), the central CMX automaton and the subnet profiles.

The CMX agent permits remote administration based on SNMP, the Simple Network Management Protocol. SNMP is a tool that allows the components of a network to be administered and monitored from a management station.

The central interface between the agent and management stations is the management information base (MIB). This describes the types of objects that can be managed and the operations that are permitted on them.

The CMX agent implements a CMX-specific management information base, referred to in the following as CMX MIB.

11.2 Functions of the CMX agent

The CMX agent is an SNMP agent for CMX as of version 5.0. It supports the administration of CMX and the transport systems implemented by the CCP products on the basis of SNMP.

This section describes:

– communication between the CMX agent and an SNMP management station

– the EMANATE-based architecture of the EMANATE Master Agent and UNIX SNMP Agent Adapter)

– the structure and operations of the management information base (MIB)

– the groups of the Internet MIB-II that are relevant to the CMX agent

– the object classes and groups of the CMX MIB

– the trap messages of the CMX MIB

11.2.1 The CMX agent and SNMP management stations

The failure of a network can be very costly and cause major problems. It is therefore important to monitor the network and its components, identify problems and introduce timely measures to deal with them.

A network contains one or more network management stations, from which the network is monitored and administered. SNMP (Simple Network Management Protocol - RFC1157) is a tool enabling the components of a computer network to be administered and monitored on the basis of TCP/IP. It is a protocol for transferring read and write requests for objects in a network.

SNMP is transferred - in its currently available implementations - via a TCP/IP-based network. In the network elements reached in this way, objects of all kinds can be managed, even objects that belong to other transport systems, such as those of CMX.

SNMP must be implemented in both the management station and the network components to be monitored. Examples of network components are UNIX end systems, bridges, routers and gateways. In the context we are concerned with, the network components are UNIX systems.

The CMX agent, which runs on a local UNIX system, communicates with a management station on the basis of SNMP. The management station requests CMX- and CCP-specific information from the CMX agent using the SNMP

operations *GET* and *GETNEXT* (see “GET request (and response)” on page 331 and “GETNEXT request (and response)” on page 331). It writes to the CMX agent using the SNMP operation *SET*. The CMX agent maps the requests of the management station to the CMX-specific system functions for local administration, e.g. *cmxinfo* (see section “Information on CMX configuration (cmxinfo)” on page 239) and *bstv*, and sends back the information requested by the management station. All the objects that can be managed and the operations permitted on them are defined in the CMX MIB (see section “The CMX MIB” on page 334).

The CMX agent does not become active until it receives a message from the management station, except in the case of the SNMP operation *TRAP*. The CMX agent can send trap messages to the management station unrequested when specific events (state transitions) are detected in the agent system (see section “Trap messages of the CMX MIB” on page 348).

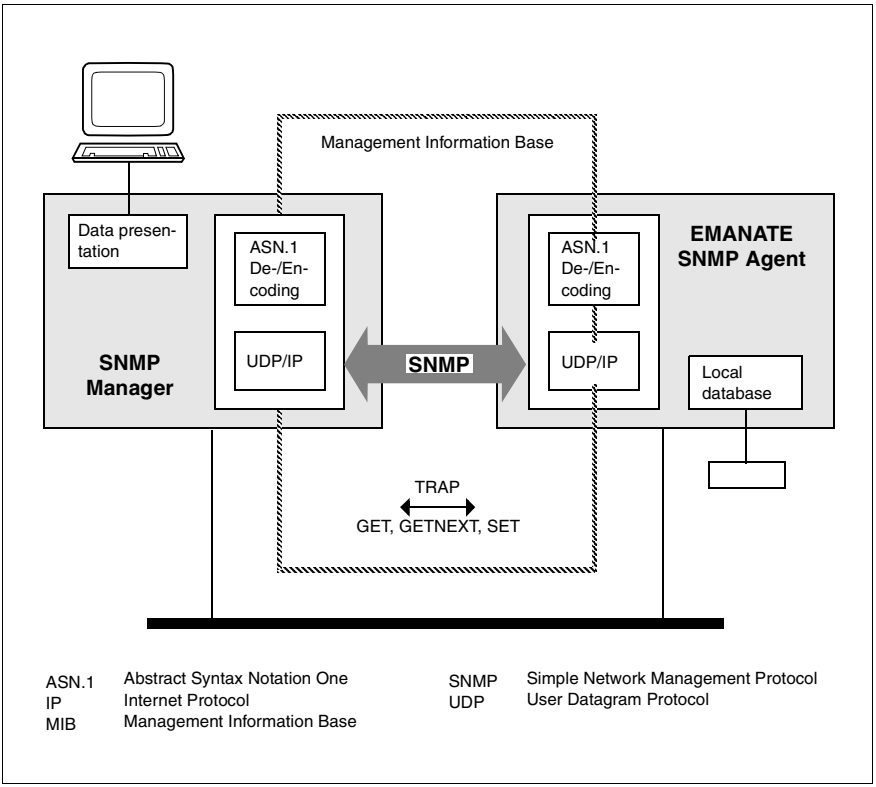


Figure 36: The SNMP network management strategy

11.2.2 EMANATE-based architecture of the agent

With the introduction of EMANATE, the UNIX SNMP agent (or core agent) is replaced by the UNIX SNMP Agent Adapter V3.0 and the EMANATE Master Agent. The previous TCP extension *SImpext* is replaced by the MIB-II subagent *SImib2*.

The CMX subagent is implemented as a separate daemon process. This can be started and stopped independently of the EMANATE Master Agent.

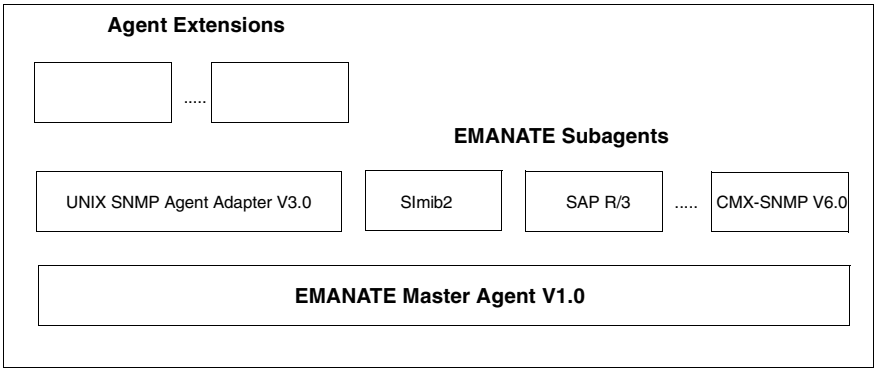


Figure 37: EMANATE-based architecture of the agent

11.2.3 The management information base (MIB)

In addition to the protocol used by the network management station and the SNMP agent to communicate with each other, the network component objects to which the management station has read or write access via the SNMP agent must also be specified.

These objects are defined in management information bases (MIBs). The standard MIB for the network management of TCP/IP-based networks is MIB-II (Management Information Base for Network Management of TCP/IP-based Internets - RFC1213). In conjunction with the TCP/IP subagent SMAWmibii, the “Solstice Enterprise Agent” contained in Solaris fully supports MIB-II.

The MIB describes the objects that can be managed and the operations permitted on them. The information on the agents that is to be managed by the management station is stored there. Each agent manages its static and dynamic information, such as measures and workloads, itself.

The MIB describes the set of all objects and operations available to a management station. It does not describe how the objects are implemented technically in the agent system or how the information is presented to the user.

In Request for Comment RFC1155 (Structure and Identification of Management Information for TCP/IP-based Internets), SNMP specifies how the definition of an SNMP MIB must be structured. RFC1157 (Simple Network Management Protocol) describes in ASN.1 notation the SNMP protocol for data transfer between a management station and the SNMP agent.

An *object identifier* is assigned to each object in the MIB, providing unique identification in a registration tree that is unique worldwide. The objects managed on the basis of SNMP appear as “leaves” in the registration tree. Intermediate nodes can be defined to structure the tree. These then have their own object identifiers. Subtrees at the lowest level of the hierarchy are sometimes referred to as *object classes*.

At the lowest level of the structure, the MIB describes all the objects managed in the agent system. OBJECT-TYPE macros are defined for this. These contain an object’s identifier, the syntax of the object value, the permitted operations and a description.

SNMP objects do not have individual attributes like those in network management based on OSI standards, for example; instead, they have a single attribute value. To comply with SNMP conventions, an object with several attributes must be converted to an SNMP object class, and the individual attributes must be converted to separate SNMP objects.

If an SNMP agent contains several objects of the same kind belonging to the same object class, they are presented in an SNMP table.

An object identifier in the registration tree that is unique worldwide indicates the object’s type, and its suffix identifies the object within an agent system. In the case of SNMP tables, this is the value of the table index; otherwise, it is zero.

system is understood by SNMP from the manager’s viewpoint as a unit identified by an IP address.

SNMP permits the following operations to be used on objects defined in the management information base:

GET request (and response)

This operation allows an object instance (or a list of object instances) to be requested from the agent system.

It is important that the SNMP manager, when making a GET request, specifies the object identifier of the instance (or list of instances), i.e.:

<object type>.0
: For simple instances (e.g. *cmxTsapMax.0*)

<object type>.<index value>
: For SNMP tables (e.g. *cmxCcType.3*)

GETNEXT request (and response)

This operation also allows an object instance (or a list of object instances) to be requested from the agent system. In this case, however, the SNMP agent determines the lexicographic successor of the object identifier specified in the GETNEXT request and makes it available in the GETNEXT response, together with the attribute value. The following examples from the CMX MIB clarify this:

Example 1

If at least one communication controller is installed, a GETNEXT request for *cmxCcType* receives the response *cmxCcType.1* and the value of the attribute (i.e. the controller type with an index of 1).

If more than one communication controller is installed, a GETNEXT request for *cmxCcType.1* receives the response *cmxCcType.2* and the value of the attribute (i.e. the controller type with an index of 2).

If no other communication controller is installed, a GETNEXT request for *cmxCcType.2* receives the response *cmxCcDescr.1* and the value of the attribute (i.e. the description of the controller with an index of 1). In the CMX MIB, the object types *cmxCcType* and *cmxCcDescr* are specified and registered one after the other.

This example shows how the SNMP manager can use GETNEXT to read out an entire SNMP table without previous knowledge of the lines in the table or their current index values.

Example 2

A GETNEXT request for *cmxTsapMax* receives the response *cmxTsapMax.0* and the value of the attribute.

A GETNEXT request for *cmxTsapMax.0* receives the response *cmxTcepMax.0* and the value of the attribute. These two attributes are specified and registered one after the other in the CMX MIB.

SET request (and response)

This operation allows the value of an object instance (or a list of object instances) to be changed in the agent system.

It is important that the SNMP manager, when making a SET request, specifies the object identifier of the instance (or a list of instances), i.e.:

<object type>.0
: For simple instances (e.g. *cmxNeateMaxConn*.O)

<object type>.<index value>
: For SNMP tables (e.g. *cmxCcAdminState*.3)

Trap messages

GET, GETNEXT and SET requests are made by the management station to the agent system, and the SNMP agent then makes an appropriate response.

Trap messages, on the other hand, are sent by an SNMP agent to the management station without being requested by the latter. These usually report serious errors in the system. In addition to the trap type, a trap message can contain additional information (attribute values for object types). The trap messages that can be sent by the CMX agent are described in section “Trap messages of the CMX MIB” on page 348.

11.2.4 The Internet MIB-II

– The MIB-II *System* group

– The MIB-II *Interface* group

The MIB-II System group

This group contains object types for identifying the whole system, which are administered by the central SNMP agent for UNIX systems; e.g. the *sysDescr* object type contains information on the Solaris system (e.g. Sun SNMP agent).

The MIB-II Interface Group

The term interface plays a central part in the SNMP data model. In particular, an interface describes a point of access to a subnet. All a system's subnet connections are presented in an SNMP table. The uniform representation of the subnet connections is independent of the LAN or WAN interface type, implementation type or product to which they belong in the UNIX system. The various SNMP subagents of a UNIX system enter here the subnet connections supported in their context. In particular, the WAN subnet connections managed by CMX and the CCP products also appear here.

The SNMP table *ifTable* contains object types for the type, subnet address and status of the subnet connection. The monitoring information offered depends on the type of the subnet connection. In this version, the CMX agent does not present any statistics for the WAN subnet connections. The counters in these cases always remain at zero.

The other MIB groups, such as *IP Group*, *UDP Group* and *TCP Group*, describe the protocol entities for Internet integration. These groups are not relevant for the CMX MIB.

11.2.5 The CMX MIB

he CMX MIB contains all the UNIX communications objects to which the management station has read access and, in some cases, write access via the CMX agent. When the CMX agent is installed, write access to the CMX MIB is disabled as standard. Write access must therefore be explicitly enabled (see section “Local administration” on page 350).

The set of objects that can be managed from the management station via a CMX agent includes all objects defined in the CMX MIB and a subset of the objects defined in RFC1213 MIB-II (the CMX-specific subnet connections in *ifTable* of MIB-II see page 333).

The objects of the CMX MIB are specified in Abstract Syntax Notation One (ASN.1). The CMX agent supports the following standards defined in the Requests for Comments (RFCs):

– RFC1155 SMI: Structure and Identification of Management Information for TCP/IP-based Internets.

– RFC1157 SNMP: Simple Network Management Protocol.

– RFC1212: Concise MIB Definitions

The ASN.1 module *cmx.asn1* describes the CMX MIB and is located in the */opt/SMAWsnmpm/asn1/snmpv1* directory of the local agent system. This specification is the formal description of the interface between a management station and a CMX agent.

A subtree with the root 1.3.6.1.4.1.231 (object identifier *sni(231)* in figure 38) is reserved for Fujitsu Siemens Computers GmbH in the registration tree that is unique worldwide.

The CMX MIB is implemented in this subtree as an intermediate node with the object identifier *sniCMX(2)*.

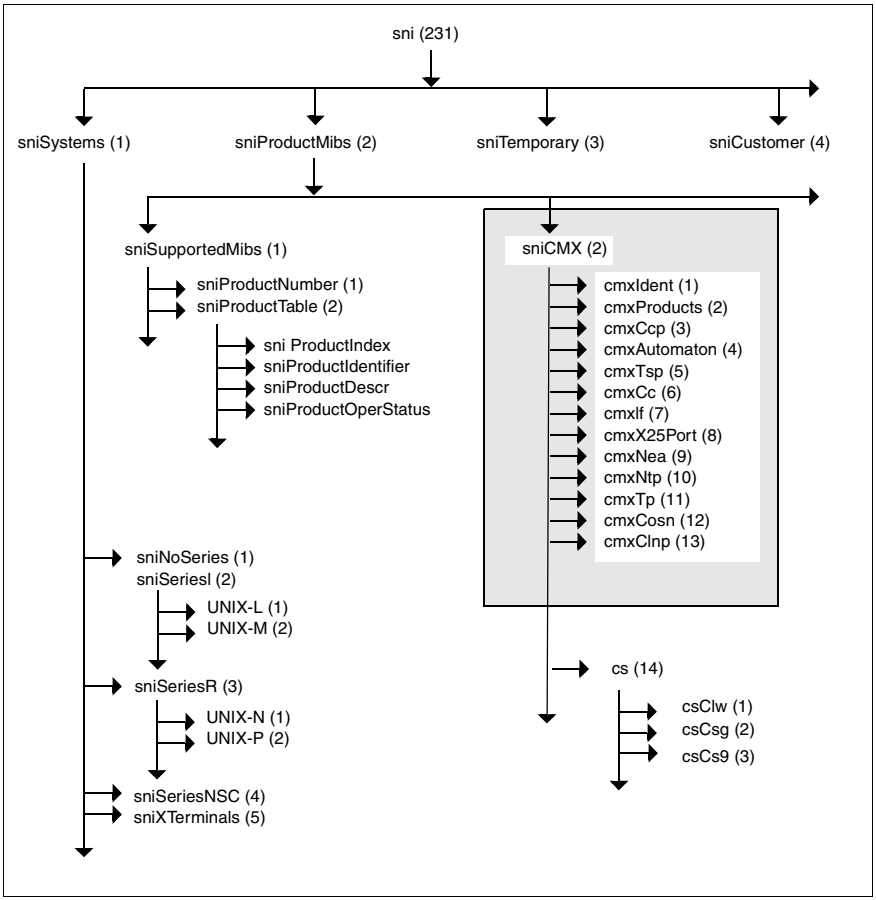


Figure 38: The CMX MIB in the SNI registration tree

The CMX MIB is organized into different CMX MIB groups, to which the various object classes are assigned. The groups and object classes of the CMX MIB model the logical structure of the communications which is shown in figure “OSI protocol stacks” on page 18.

A brief overview of the essentials of communication on UNIX systems is provided below before the individual MIB groups are described in more detail.

A central concept is the CCP profile. A CCP profile defines a protocol for each of the four lower layers of the OSI Reference Model. It thus defines specific characteristics of the network.

The protocol entities of a CCP profile are implemented differently depending on their type: partly in the UNIX kernel and/or as components of the loadware on the communication controllers. The attributes and operations that are available for network management are also dependent on this.

In order to obtain a uniform view for managing the various CCP profile types, the following object classes are defined:

– Transport Service Provider (TSP)

  This object class comprises all components (protocol entities) of CCP profiles that run in the UNIX kernel as manageable objects and control the subnet profiles on the communication controllers. The individual protocol entities in the TSPs are made available to the management station via their own MIB subgroups.

– Communication Controller (CC)

  This object class describes the communication hardware for connecting the UNIX system to a subnet in a uniform view for management. The communication controller is assigned as an attribute the subnet profile that runs.

– TSP Access Point

  At its access points to the communication components, the central CMX automaton provides a uniform view of configuration and monitoring attributes. The TSP Access Point object class was defined for the management of these attributes.

The diagram below illustrates how these three object classes interact. Refer also to the section “Architecture of CCP profiles” on page 23.

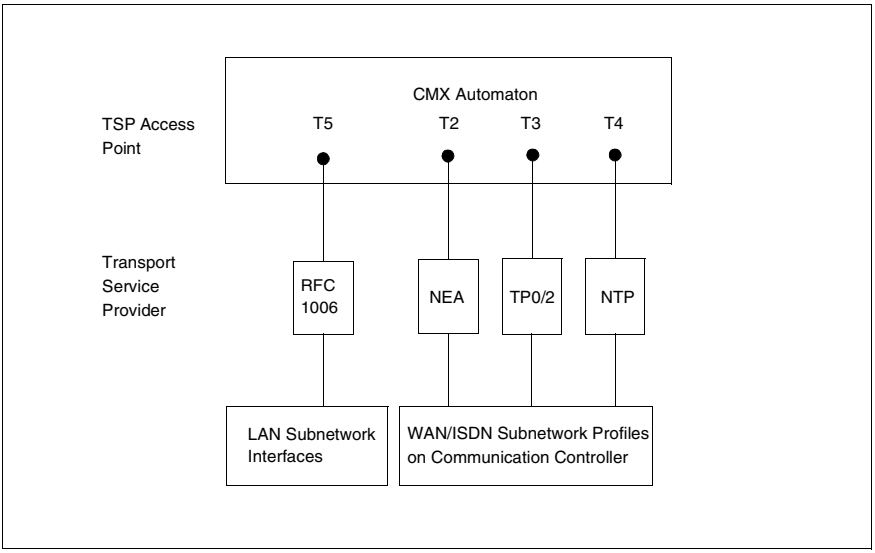


Figure 39: Logical structure of communication in UNIX systems

You will find more information on the architecture and the communication components in the manuals “CMX/CCP, ISDN Communication” [3] and “CMX/CCP, WAN Communication” [4].

The following figure illustrates the assignment of Transport Service Providers and subnet connections or subnet profiles to the groups of the CMX MIB.

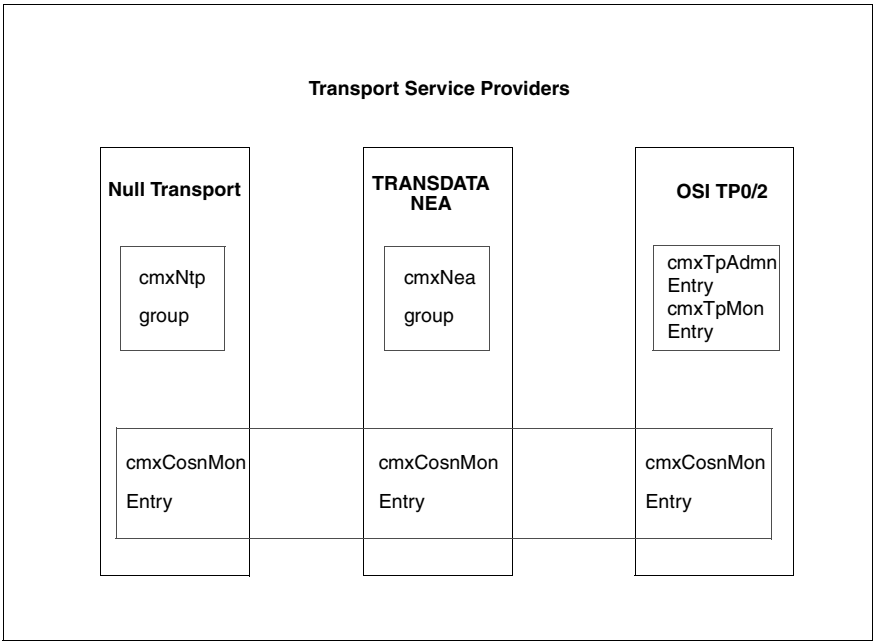


Figure 40: Structure of the Transport Service Providers from the viewpoint of SNMP

The transport entities in the OSI TP0/2 and OSI TP4/CLNP Transport Service Providers are presented in a uniform view in the *cmxTp* group of the CMX MIB. For each entity, an entry in *cmxTpAdmnTable* or *cmxTpMonTable* of the *cmxTp* group describes the configuration and monitoring attributes.

The connection-oriented subnet layer in the Transport Service Providers for WANs is described in the *cmxCosn* group of the CMX MIB. Each Transport Service Provider in the WAN has an entry in *cmxCosnMonTable* (monitoring).

The following sections outline the essential contents of each group in the CMX MIB in the order in which the groups occur in the CMX MIB (see figure 38). The focus is on describing the purpose and interrelationships of the various object classes, without going into details. You will find more information on the object classes in the inline descriptions of the ASN.1 module *cmx.asn1*.

11.2.5.1 The CMX MIB group cmxIdent

This MIB group contains three object types with general product and version information on CMX (*cmxProductDescr*) and the CMX agent (*cmxSnmpDescr*).

By means of *cmxMibVer*, the agent indicates the version of the MIB that is currently supported. CMX V5.1 enters the number 1 here. This version number changes when changes or additions are made subsequently to the MIB definitions. In this way, applications running on a management station can distinguish between agents with different CMX MIB version statuses.

11.2.5.2 The CMX MIB group cmxProducts

This MIB group provides an overview in two SNMP tables of all installed communication products and packages:

– *cmxProdTable* – table of products

– *cmxProdPkgTable* – table of packages with product assignment in each case

Both tables are necessary, because a product can consist of a number of packages and, on the other hand, a package of several products can also be used.

The information is provided in printable form. For indexing purposes, the product or package name is used as the display string. The *cmxProdPkgTable* is organized as a matrix with a two-level index.

11.2.5.3 The CMX MIB group cmxCcp

This MIB group provides information on the subnet profiles. *cmxCcpCfTable* lists all the existing configuration files (CFs) for the installed subnet profiles. You can see from the table which configuration files you can assign to a communication controller. This depends on the subnet profile. The files listed in *cmxCcpCfName* are permissible attribute values in an SNMP SET operation relating to *cmxCcCfAss* in *cmxCcTable* of the CMX MIB group *cmxCc*.

cmxCcpCfTable has two index attributes:

– *cmxCcpCfPIndex*

  This is the primary index attribute. It indicates the type of the subnet profile as an ASN.1 named number.

– *cmxCcpCfIndex*

  This is the secondary index attribute. It numbers the configuration files consecutively for each subnet profile.

  This is an exceptional case because it does not adhere to the SNMP convention that an index value should always indicate the same object instance during the runtime of the SNMP agent. The CMX agent lists the configuration files alphabetically for each subnet profile. The index values can change if files are added or deleted during the runtime of the agent.

11.2.5.4 The CMX MIB group cmxAutomaton

The MIB group *cmxAutomaton* contains the object types of the central CMX automaton. This MIB has four object classes:

– *cmxAutGlob* – Global CMX configuration and statistics

– *cmxTsapTable* – Transport Service Access Points

– *cmxTcepTable* – Transport Connection End Points

– *cmxTspAccCTable* – TSP Access Points

Global CMX configuration and statistics

The CMX automaton exists only once in the system. Its configuration and monitoring attributes are therefore registered in the MIB as a single subtree (*cmxAutGlob*).

Configuration attributes

The maximum number of Transport Service Access Points (*cmxTsapMax*) or Transport Connection End Points (*cmxTcepMax*) supported by CMX is specified here.

Simple monitoring attributes, counters

The counters run as of booting. Appropriate management applications can obtain the throughput values by polling the counters.

Only 32-bit counters can be defined in an SNMP MIB. The CMX MIB therefore presents the low/high values as separate object types that can be merged again, if necessary, by a management application.

Note the following feature of the counters for the sent and received bytes: in each case, the CMX automaton counts internally in two ‘unsigned longs’; the overflow from low to high value occurs at 10^9. This ensures unambiguous counts even when the system runs for an extended period at high throughput.

To support the simple presentation formats at a management station, the CMX agent also combines the low/high values in a readable exponential format and presents this as a separate object type. The management station decides which of these it wants to use.

Transport Service Access Points (TSAP)

The TSAPs active in the system are listed in the *cmxTsapTable*.

A TSAP can comprise several TSEL values. Each TSEL is in turn assigned to one or more Transport Service Providers, via which a connection can be established to this TSEL.

Each TSAP is identified in the CMX automaton by a number (ID) in the range from 0 to *cmxTsapMax* (see above). This ID is used as the primary table index *cmxTsapIndex*. It can be released and subsequently reassigned to a new TSAP that is to be opened; i.e. it does not provide unique identification when the system runs for an extended period. Therefore, each time the same ID is reassigned, an incarnation counter is incremented; this serves as the secondary table index (*cmxTsapInc*).

The third index *cmxTsapTselInd* sequentially numbers the TSEL values within a TSAP.

Transport Connection End Point (TCEP)

The TCEPs active in the system are listed in *cmxTcepTable*.

A TCEP is assigned to precisely one Transport Service Provider. The ID and incarnation counter of the TSAP are therefore used as the primary table index of the TCEP, whereby a unique relationship to the corresponding TSAP is guaranteed.

Each TCEP is identified in the CMX automaton by a number (ID) in the range from 0 to *cmxTcepMax* (see above). This ID is used as the third table index *cmxTcepIndex*. It can be released and subsequently reassigned to a new TCEP that is to be opened; i.e. it does not provide unique identification when the system runs for an extended period. Therefore, each time the same ID is reassigned, an incarnation counter is incremented; this serves as the fourth index (*cmxTcepInc*).

In addition to general information such as the state, time stamp of the connection setup, and Transport Service Provider used, address information and statistical values are provided for a TCEP.

Address information

In the case of a local application, the TSEL is displayed. In the case of a remote application, information on the address of the remote system is supplied in addition to the TSEL. If it is known, the network address (OSI, NEA or IP address) is displayed. If it is not known, the subnet address (MAC or DTE address, ISDN call number, etc.) by means of which the system is accessed is displayed. If the subnet address is not known either (e.g. in the case of a leased line), the local subnet connection used for communication is displayed.

Statistical values

As with the counters of the central CMX automaton, two object types are used in each case as low/high values with an overflow of 10^9. The explanation in section “The CMX MIB group cmxCcp” on page 339 applies here too.

TSP Access Points

The TSP Access Points known to the CMX automaton are listed in *cmxTspAccCTable*.

CMX defines a TSP Access Point for each Transport Service Provider (TSP). A TSP Access Point defines the access point to the Transport Service Provider. CMX presents the configuration and monitoring attributes in the same way for all TSPs.

The TSP Access Points of the TSPs implemented in the UNIX kernel are each assigned to a single TSP. These TSPs are listed in *cmxTspTable*.

The TSP Access Points of the TSPs implemented in the CC loadware are each assigned to a single CC. These TSPs are listed in *cmxTspTable*.

The attributes *cmxTspAccCTsp* and *cmxTspAccCCc* indicate assignment to a TSP or CC respectively and include a reference to *cmxTspIndex* or *cmxCcIndex* in *cmxTspTable* (MIB group *cmxTsp*) or *cmxCcTable* (MIB group *cmxCc*). The inapplicable attribute has the value zero in each case.

The object types for the configuration attributes, monitoring attributes, counters and the measurement operation request for throughput values are organized as in the *cmxAutGlob* subgroup of the CMX automaton (see section “The CMX MIB group cmxAutomaton” on page 340). Whereas there the count applies globally

for the CMX automaton, here the values apply to a single TSP Access Point in each case. The comments above on measurement operation requests in particular apply by analogy here too.

11.2.5.5 The CMX MIB group cmxTsp

Transport Service Providers play an important part in the CMX MIB concept. A TSP describes all the components (protocol entities) of CCP profiles required to control subnet profiles. All types of TSPs are described in a single object class, regardless of their implementation.

The SNMP table *cmxTspTable* provides an overview of the installed TSPs with an index of consecutive integers. Other object classes refer to a TSP by means of its index value.

cmxTspTable has only a few standard attributes, such as the type and state of the TSP. In addition, the START and STOP operations are anchored here. You can start or stop the TSP using a SET operation on the *cmxTspAdminState* object type.

Some of the protocol entities in the TSPs have very different configuration and monitoring attributes that cannot be included in the standard table *cmxTspTable*. The CMX MIB therefore provides separate MIB groups (e.g. *cmxNea* and *cmxNtp*) for managing these entity-specific object types (figure 40 provides an overview).

11.2.5.6 The CMX MIB group cmxCc

The communications controller (CC) object class describes the communication hardware for connecting the UNIX system to a subnet in a uniform view for management. The communication controller is assigned as an attribute the subnet profile that runs on it or a complete CCP profile. The CCs are listed in the SNMP table *cmxCcTable*.

Important attributes are cmxCcOperState, cmxCcAdminState, cmxCcCcpAss, and cmxCcCfAss.

– *cmxCcOperState.* This indicates the current state of an installed CC.

– *cmxCcAdminState.* You can use a SET operation on this object type to load, terminate or dump the CC. A GET operation always returns the value *none(0)*. A successful SNMP SET operation means that the request has been accepted in the agent system. After the request has been acknowl-

edged, it is executed in the background because it can take several seconds to load a CC, for example. Whether a loading operation has actually been successful must be established by subsequently polling *cmxCcOperState*.

– *cmxCcCcpAss*, *cmxCcCfAss*. Here, you use an SNMP operation to set which subnet profile and configuration file are to be used when loading the CC. *cmxCcpCfTable*, which is described in the MIB group *cmxCcp*, lists the subnet profiles for which configuration files are available on the agent system. Changes made to the assignments become effective the next time the CC is loaded.

  If only the *cmxCcCfAss* attribute is changed, the change applies to the subnet profile that has just been assigned.

  If only the *cmxCcCcpAss* attribute is changed, the configuration file last linked with the new subnet profile becomes effective as the assigned file.

– *cmxCcCcpLoad*, *cmxCcCfLoad*. When a CC is loaded, the subnet profile and configuration file used are displayed here. These values can deviate from the values in *cmxCcCcpAss* and *cmxCcCfAss* if other assignments were made during the runtime of a CC which do not become effective until the CC is next loaded.

11.2.5.7 The CMX MIB group cmxIf

This MIB group lists in *cmxIfTable* all configured subnet connections managed and served by the CMX. The LAN connections managed by the TCP/IP extension are not included in this table.

The subnet connection object class describes the access points to the subnets ISDN, DATEX-L, DATEX-P, etc. A subnet connection comprises all the attributes of a subnet point of access and defines a line to the subnet that is connected on the communication controller. *cmxIfTable* indicates the relevant CC by means of the *cmxIfCc* attribute.

You can use a SET operation on the *cmxIfAdminState* attribute to activate or deactivate the corresponding subnet connection.

The various subnet types (e.g. X.25) each have different types of configuration attributes that cannot be presented in *cmxIfTable*. They are presented in additional SNMP tables designed specifically for each connection type (e.g. *cmxX25PortTable* in the CMX MIB group *cmxX25Port*). If such an extension exists for an entry in *cmxIfTable*, the *cmxIfSpecific* and *cmxIfSpecificIndex* attributes refer to it.

11.2.5.8 The CMX-MIB group cmxX25Port

An extension of the *cmxIfTable* interface table for X.25 connections is defined in the CMX MIB (see the MIB group *cmxlf*). *cmxX25PortTable* contains additional information on subnet connections to an X.25 packet network. The following example illustrates how *cmxX25PortTable* and *cmxIfTable* are linked.

The SNMP table *cmxIfTable*

cmxIfIndex	**...**	**3**	**4**	**5**
. . .		Dx-P4	Dx-P	ISDN
cmxIfSpecific		cmxX25PortTable	cmxX25PortTable	0.0
cmxIfSpecificIndex		1	2	0

Table 28: cmxIfTable

The SNMP table *cmxX25PortTable*

cmxX25PortIndex	**1**	**2**	**3**	**4**	**5**
cmxX25IfIndex	3	4	0	0	0
.....					

Table 29: cmxX25PortTable

CMX supports the following three connection types to a packet-switched data network (PSDN).

- Direct access to the PSDN

  If the subnet connection of *cmxIfTable* describes a direct connection to a PSDN, an entry for this is created in *cmxX25PortTable* to display the X.25 configuration parameters of the packet level entity. The two assigned entries are linked to each other in both directions by the *cmxIfSpecificIndex* and *cmxX25IfIndex* attributes. In the example, these are entries 3 and 4 in *cmxIfTable* and entries 1 and 2 in *cmxX25PortTable*.

- Access to the PSDN via a dedicated ISDN B channel

  The B channel is displayed in *cmxIfTable* as an ISDN subnet connection to which a single X.25 parameter set is assigned in *cmxX25PortTable*. The links between the two tables are analogous to those described above for direct access to the PSDN.

- Switched access to the PSDN via the ISDN switched network

  The ISDN-S0 connection is displayed as a subnet connection in *cmxIfTable*. No X.25-specific characteristics can be assigned to this switched connection to the ISDN network.

  In two-stage switching to the packet-switched network, the X.25 parameters and the caller's DTE address to be used depend on the partner address. The X.25 parameter sets to be used for configuration are listed in *cmxX25PortTable*. However, these entries are not linked to *cmxIfTable*. In the example, they are entries 3, 4 and 5 in *cmxX25PortTable*.

11.2.5.9 The CMX MIB group cmxNea

This group contains all the configuration and monitoring attributes of the NEATE and NEAN protocol entities for the TRANSDATA NEA Transport Service Provider. Each of these protocol entities has its own subgroup in the MIB: *cmxNeate* and *cmxNean*.

When the TRANSDATA NEA TSP is started, all counters are reset. All configuration attributes refer to the currently running protocol entity. Changes affect the protocol entity immediately and are only possible when the TSP is active. Depending on the local configuration, the attributes set in the local agent system apply again when the Transport Service Provider is restarted or, at the latest, when the UNIX system is rebooted.

You will find the monitoring attributes of the connection-oriented subnet layer within the TRANSDATA NEA TSP in the MIB group *cmxCosn*.

11.2.5.10 The CMX MIB group cmxNtp

This group contains all the configuration and monitoring attributes of the NULLTP protocol entity for the Null Transport TSP.

When the Null Transport TSP is started, all counters are reset. All configuration attributes refer to the currently running protocol entity. Changes affect the protocol entity immediately and are only possible when the TSP is active.

Depending on the local configuration, the attributes set in the local agent system apply again when the Transport Service Provider is restarted or, at the latest, when the UNIX system is rebooted.

You will find the monitoring attributes of the connection-oriented subnet layer within the Null Transport TSP in the MIB group *cmxCosn*.

11.2.5.11 The CMX MIB group cmxTp

This group contains all the object classes defined for the management of the ISO protocol entities for the OSI TP0/2 Transport Service Provider. The SNMP tables each contain all the possible object types for the 0/2 transport protocol classes. Depending on the protocol class in each case, some object types are not relevant or have different ranges of values.

Two SNMP tables are defined. The index values number the transport entities consecutively; identical index values in both tables refer to the same protocol entity.

- *cmxTpAdmnTable* - SNMP table for configuring the ISO protocol entities

  Most of the object types defined here describe initial attribute values for new transport connections that are to be set up. Depending on the local configuration, the attributes set in the local agent system apply again when the Transport Service Provider is restarted or, at the latest, when the UNIX system is rebooted.

- *cmxTpMonTable* - SNMP table for monitoring the ISO protocol entities

  This contains all the attributes for monitoring the protocol entities.

  You will find the monitoring attributes of the connection-oriented subnet layer in the OSI TP0/2 TSP in the MIB group *cmxCosn*.

11.2.5.12 The CMX MIB group cmxCosn

This group contains all the object classes defined for the management of the connection-oriented subnet layer in the Transport Service Providers in the WAN.

Monitoring within the subnet layer applies in each case to a single TSP. This TSP is entered in the *cmxCosnMonTsp* index by means of a reference to the index value in *cmxTspTable*.

Depending on the type of the TSP, protocol data units (PDUs) and/or interface data units (IDUs) are included in the network layer. Inapplicable counters stay at zero. This makes it possible to have a standard SNMP table for all TSP types.

11.2.6 Trap messages of the CMX MIB

SNMP allows agents to send trap messages to management stations unrequested when they detect certain events.

Trap messages are either predefined in SNMP or are vendor-specific extensions for certain systems. RFC 1215 lays down how to define trap messages (RFC 1215: A Convention for Defining Traps for Use with the SNMP).

The trap messages linkDown and linkUp

The CMX agent supports the two trap messages *linkDown* and *linkUp* defined in RFC 1157 (Simple Network Management Protocol). They indicate respectively the failure and startup of a subnet connection. By default, the CMX agent can send these two trap messages. If it is not to send them, the system administrator in the local agent system must set the *IFPOLLTIME* parameter in the *AgentParams.rc* file to zero (see section “The AgentParams file” on page 351).

The trap messages cmxCcUp and cmxCcDown

The additional trap messages *cmxCcUp* and *cmxCcDown* are vendor-specific extensions for communications that inform the management station of the state transitions of the communication controllers. *cmxCcDown* indicates that a communication controller has failed, and *cmxCcUp* that one has been started up. If neither of these trap messages is to be sent, the system administrator in the local agent system must set the *CCPOLLTIME* parameter in the *AgentParams.rc* file to zero (see section “The AgentParams file” on page 351).

11.3 Running the CMX agent

The information provided in the chapter “Addressing concept” on page 37 is aimed exclusively at the system administrator of the local UNIX system, who is likewise referred to in this chapter as the “system administrator”.

When starting up the CMX agent in the local UNIX system you must perform the following tasks, which are described in more detail in the following sections:

1. install the CMX agent
2. administer the CMX agent locally, if necessary

11.3.1 Installing and starting the CMX agent

Installation

Before you can install the CMX agent, CMX and the EMANATE Master Agent must already be installed. You may have to install the EMANATE Master Agent SMAWsnmpm from a Fujitsu Siemens Computers add-on CD.

For more information, particularly on version dependencies, see the Release Notice.

To install the CMX agent, proceed as follows:

1. Ensure that the EMANATE Master Agent (SMAWsnmpm package) and the SMAWadapt (Siemens Native Agent Adapter) package are already installed.
2. Install the SMAWcxagt package.
3. Following successful installation, the EMANATE Master Agent and the CMX agent are started automatically.

   The CMX agent is now operational. The CMX agent is also activated automatically when you restart your UNIX system.

Once the CMX agent is installed, write access to the CMX MIB is disabled by default and must therefore be explicitly enabled (see section “Local administration” on page 350).

Starting and terminating, diagnosis

The CMX agent is started automatically when the operating system is started, and is stopped automatically when the system is stopped (init script */etc/init.d/cmxema*). Using the *cmxsnmp [start|stop]* command, the subagent can be started or stopped at any time independently of the Master Agent (see section “Starting and stopping CMX and TSPs (StartStop)” on page 302).

The automatic start of the CMX agent at system startup can be controlled with *cmxsnmp [autostart|autostop]*.

Using *cmxsnmp diag*, you can list trace entries of the CMX agent from the */var/opt/SMAWcmx/tmp/cmxsnmp.[12].trc* files.

11.3.2 Local administration

Two files containing default values that you can modify exist for the parameters required to set up and start the CMX agent:

– the *AgentParams.rc* file, in which you can set parameters for certain timers, for example

– the *AgentTraces.rc* file, in which you can define trace points

To make changes to the configuration, you can either edit these two files directly with an editor or use the *cmxsnmpadm* command (see section “Reconfiguration” on page 356). If you want to use an editor, please read the notes on this at the end of the section “Reconfiguration” on page 356.

The defaults in the two files are such that you can set up and start the CMX agent immediately without making any changes to parameters in the files.

Please note the following default settings in the *AgentParams.rc* file, which are of particular importance for your configuration:

– The management station is not permitted to write-access the objects of the CMX MIB (see the SETENABLE parameter).
– The CMX agent generates trap messages for state transitions of the *linkUp* and *linkDown* subnet connections (see the IFPOLLTIME parameter).
– The CMX agent generates trap messages for state transitions of the *cmxCcUp* and *cmxCcDown* communication controllers (see the CCPOLLTIME parameter).

To configure the EMANATE Master Agent, you have to specify which management stations should receive trap messages.

11.3.2.1 The AgentParams file

The *AgentParams.rc* file is shipped in the */opt/SMAW/SMAWcmx/lib/cmxsnmp* directory with the defaults listed below.

The AgentParams.rc file with defaults

```
# COPYRIGHT (C) Fujitsu Siemens Computers GmbH 2000
#               All Rights Reserved
#
#        SNMP EMANATE Subagent for CMX
#
#        Parameterfile
#
# Comments must be marked by # in the first line position.
#
############################################################
#
 SETENABLE 0     # SNMP SET Operations allowed (1) or not (0)
 MAXTRACE 100    # maximum length of trace files (in kilo bytes)
#
# Timer values in seconds.
#
 MAXHOLDTIME 10  # holding timer until update of internal tables
#                # (proposed value for MAXHOLDTIME: 10)
 CFPOLLTIME 3600 #if not 0: poll for updates in CCP config files
#                # (proposed value for CFPOLLTIME: minimum 900)
 CCPOLLTIME 180  # if not 0: poll for CC state an send CC-TRAP
#                # (proposed value for CCPOLLTIME: 60...300)
 IFPOLLTIME 1800 # if not 0: poll for IF state an send IF-TRAP
#                # (proposed value for IFPOLLTIME: minimum 900)
#
# The following values are used to determine the
# size of internal tables created during startup.
# Normally they need not be changed.
#
 MAXCC 70        # maximum number of elements in cmxCcTable
 MAXTSP 10       # maximum number of elements in cmxTspTable
 MAXTSPACC 40    # maximum number of elements in cmxTspAccCTable
 MAXIF 160       # maximum number of elements in cmxIfTable
 MAXX25 130      # maximum number of elements in cmxX25PortTable
 MAXTSPSET 100   # maximum number of elements in cmxTspSetTable
 MAXTSEL 0       # maximum number of elements in cmxTsapTable
#              # (0: CMXSNMP uses the CMX maxTSAP configuration)
 MAXTCEP 0       # maximum number of elements in cmxTcepTable
#              # (0: CMXSNMP uses the CMX maxTCEP configuration)
```

Parameter explanations

SETENABLE

You use the *SETENABLE* parameter to specify whether the management station can execute write operations on objects in the CMX MIB via the CMX agent. You can only authorize write access for the CMX agent globally, not for individual objects in the CMX MIB. The default is “0”; i.e.

that the management station does not have write access to objects in the CMX MIB, even when it is assigned write access in the EMANATE master agent configuration.

0:
no write access

1:
write access permitted

To configure the EMANATE Master Agent, you must specify the management stations from which write access is permitted.

MAXTRACE
You use this parameter to specify the maximum size of the two trace files *cmxsnmp.1.trc* and *cmxsnmp.2.trc* (see section “The AgentTraces file” on page 355).

Permitted range (in kilobytes): *1*.. 100 ...

MAXHOLDTIME
When an SNMP GET request is made, the CMX agent calls the required system function (e.g. *bstv* or *cmxinfo*) to obtain the information from the CMX MIB. It stores this data (e.g. *cmxCcTable*) internally in a buffer. The *MAXHOLDTIME* timer runs as of this point. The table is kept in the buffer until the timer runs out. If information is requested on the same CMX MIB table during this time, the CMX agent provides it from this buffer without calling a system function. If a GET request is made again after this time has elapsed, the appropriate system function is called again, the CMX MIB table set up in the internal buffer, and the timer restarted. This timer algorithm is implemented for some of the tables in the CMX MIB. *MAXHOLDTIME* determines how long these timers run for.

This mechanism improves the performance of the CMX agent. A sequence of GETNEXT requests to read out a CMX MIB table generally results in the relevant system function being called only once.

Permitted range (in seconds): *1* .. 10 ...

By default, the timer runs for 10 seconds. If you set a lower value, this may adversely affect the performance of the CMX agent. If you set a higher value, you must remember that any changes made to the table while the timer is running (e.g. status changes or new object instances) will not yet be visible in the CMX MIB when a GET request is made.

CFPOLLTIME

At startup, the CMX agent reads out the specific configuration files of the subnet profiles on the communication controllers in order to set up the CMX MIB tables *cmxIfTable* and *cmxX25PortTable*. So as not to impair performance, these configuration files are not updated immediately in the internal tables (the tables in the internal buffer) unless the changed configuration file also changes its name. If the name remains the same, changes made in a configuration file during the runtime of the CMX agent are not taken into account until the time specified for *CFPOLLTIME* has elapsed. The state of a subnet connection is of course kept constantly up to date in *cmxIfTable*.

If you expect critical changes in these configuration files of the subnet profiles that have to be communicated to the CMX agent immediately, you can use the *CFPOLLTIME* timer to specify more frequent cyclic updating of the internal tables. A long runtime (e.g. 3600 seconds) is recommended for the timer for performance reasons.

0:
no cyclic updating of the internal tables

60 ... 3600 ...:
cyclic updating of the internal tables (value in seconds)

CCPOLLTIME

You can use the *CCPOLLTIME* parameter to specify whether or not the CC-specific trap messages *cmxCcUp* and *cmxCcDown* are to be sent. If you want the management station to be informed of the state transitions of a communication controller, you must use the *CCPOLLTIME* timer to specify that the CC status be polled. When the timer runs out, the *cmxinfo* system function is called. For performance reasons, a value in the range from 60 to 300 seconds is therefore recommended.

0:
CC trap messages are not set.

If you disable the sending of CC trap messages by specifying a value of zero, you should set a lower timer value for *IFPOLLTIME* for the monitoring of the subnet connections (see below).

60 ...180 ...:
CC trap messages are sent (value in seconds).

To configure the EMANATE Master Agent, you must specify which management stations are to receive trap messages.

IFPOLLTIME

You can use the *IFPOLLTIME* parameter to specify whether the SNMP trap messages *linkUp* and *linkDown* should be sent when the state of a subnet connection changes.

If you have disabled the sending of CC trap messages (by setting the *CCPOLLTIME* parameter to zero; see above), you should set a low value (e.g. 180 seconds) for the *IFPOLLTIME* parameter so that if a communication controller fails, the failure is not detected too late.

0:
The *linkUp* and *linkDown* trap messages are not sent.

1... 1800 ...:
The *linkUp* and *linkDown* trap messages are sent (value in seconds).

Normally, the CMX agent is informed directly of the state changes of subnet connections by news messages of the communication controllers. It can then send an appropriate SNMP trap message (*linkUp* or *linkDown*) straight away. To prevent news messages being lost in special error situations, the CMX agent also polls the news files (*/var/opt/SMAWcmx/tmp/cc_NEWSFILE_0/1*). You set the time interval by means of the *IFPOLLTIME* parameter. For performance reasons - and because, as described above, the news is normally detected immediately - a long runtime (e.g. 1800 seconds) is recommended.

To configure the EMANATE Master Agent, you must specify which management stations are to receive trap messages.

The remaining eight *MAX** parameters relate to the size of the internal tables of the CMX agent and are described by the respective comment in the file. When the product is shipped, these parameters are defined with a sufficient value. The parameter values should not be modified.

11.3.2.2 The AgentTraces file

The *AgentTraces.rc* file is supplied in the */opt/SMAW/SMAWcmx/lib/cmxsnmp* directory with the default values listed below.

```
# COPYRIGHT (C) Fujitsu Siemens Computers GmbH 2000
#               All Rights Reserved
#
#        SNMP EMANATE Subagent for CMX
#
#        Tracepointfile
#
# Comments must be marked by # in the first line position.
#
# Delete the # to activate the appropriate trace.
#
#############################################################
#
# TRAPS  # trace trap generation
# POLLS  # trace all cyclic polling algorithms
# CFUPD  # trace evaluation of CFs and update of interface table
# HOLDT  # trace start and expiration of holding timers
# GETARG    # trace all SNMP GET request
# SYSCALL # trace all calls of system functions and scripts
# ADDR    # trace address evaluation in TSAPs and TCEPs
# NEWS    # trace news reception from bstvd
```

The meaning of each trace point is explained in the comment lines of the file.

Trace points that do not have a hash character (#) at the beginning of the line are activated. Trace points that do are inactive. By default, all trace points are inactive.

The trace information is written first to the */var/opt/SMAWcmx/tmp/cmxsnmp.1.trc* file and then on overflow to the */var/opt/SMAWcmx/tmp/cmxsnmp.2.trc* file, and vice versa. The size of the trace files is defined by the *MAXTRACE* parameter in the *AgentParams.rc* file.

When the CMX agent is started or restarted, some information, such as the current parameter values and the activated trace points, is written to the first trace file.

Trace outputs are written for all activated trace points. When system errors occur, there is always trace output regardless of whether or not trace points are activated.

The file layout and number of trace entries may change in subsequent versions. There is no guarantee that the layout of these files will stay the same.

11.3.2.3 Reconfiguration

If you want to run the CMX agent with the above standard configuration, you can skip the rest of this section.

If you want to run the CMX agent with another configuration, you have to customize the *AgentParams.rc* and *AgentTraces.rc* files to suit your requirements.

To change parameter values in the *AgentParams.rc* and *AgentTraces.rc* files, you can use the *cmxsnmpadm* command or any editor.

Using the cmxsnmpadm command to make changes

The *cmxsnmpadm* command lets you make the required changes interactively in one or both files.

After calling *cmxsnmpadm*, you are asked whether you want to change each parameter value and, if you reply that you do, requested to enter the new value in the next line; e.g.:

```
MAXTRACE 100 # maximum length of trace files (in kilo bytes)
change parametervalue? y | n | q (default: n) y
new value for MAXTRACE : 60
```

This is done for the parameters in the *AgentParams.rc* file one after the other, and then for the parameters in the *AgentTraces.rc* file. The order in which the parameters are dealt with corresponds to the order in which they occur in the files.

You can terminate this procedure by entering *q* (for quit). The changes you have already entered take effect.

At the end of the command you are asked whether the changes should be activated immediately or after the system has been restarted:

```
activate updates by restarting CMX Subagent? y | n (default: y)
```

There are three ways to call the command:

Format 1**: cmxsnmpadm**

Format 2: **cmxsnmpadm␣-p**[␣parameter]

Format 3: **cmxsnmpadm␣-t**[␣tracepoint]

Format 1: Making changes in both files

If you call the *cmxsnmpadm* command without parameters, the parameters of the *AgentParams.rc* file are offered to you for changing one after the other, followed by those of the *AgentTraces.rc* file.

Format 2: Making changes in the AgentParams.rc file

-p

If you do not explicitly specify one of the parameters in the *Agent-Params.rc* file after the *-p* option, you are prompted to make a change for every parameter in the file.

parameter

Enter the name of the parameter you want to change in the *Agent-Params.rc* file. You are then prompted to make a change for this parameter only.

parameter = {SETENABLE | MAXTRACE | MAXHOLDTIME | CFPOLLTIME| CCPOLLTIME | IFPOLLTIME | TIMEOUT | TIMEOUTLOG | MAXCC | MAXTSP | MAXTSPACC | MAXIF | MAXX25 | MAXTSPSET | MAXTSEL | MAXTCEP}

Changing the *MAX** parameter during operation invokes a restart of the CMX agent.

Example

You want to change the values for the *MAXTRACE* and *CCPOLLTIME* parameters and activate the changes immediately:

cmxsnmpadm -p

```
##### process parameters in /opt/SMAW/SMAWcmx/lib/cmxsnmp/AgentParams.rc ####

SETENABLE 0      # SNMP SET Operations allowed (1) or not (0)
change parametervalue? y | n | q (default: n) n

MAXTRACE 100     # maximum length of trace files (in kilo bytes)
change parametervalue? y | n | q (default: n) y
new value for MAXTRACE : 60

MAXHOLDTIME 10   # holding timer until update of internal tables
change parametervalue? y | n | q (default: n) n

CFPOLLTIME 3600  #if not 0: poll for updates in CCP config files
change parametervalue? y | n | q (default: n) n

CCPOLLTIME 180   # if not 0: poll for CC state an send CC-TRAP
change parametervalue? y | n | q (default: n) y
new value for CCPOLLTIME : 100

IFPOLLTIME 1800  # if not 0: poll for IF state an send IF-TRAP
change parametervalue? y | n | q (default: n) q

activate updates by restarting CMX Subagent? y| n (default: y) n
```

Format 3: Making changes in the AgentTraces.rc file

-t

If you do not explicitly specify one of the parameters in the *AgentTraces.rc* file after the *-t* option, you are prompted to make a change for each parameter in the file.

tracepoint

Enter the name of the trace point you want to activate or deactivate in the *AgentTraces.rc* file. You are then prompted to make a change for this trace point only.

traces = { TRAPS | POLLS | CFUPD | HOLDT | GETARG | SYSCALL | ADDR | NEWS }

Example

You want to activate the *TRAPS* trace point. You do not want this change to take effect until the next time the CMX agent is started:

cmxsnmpadm -t TRAPS

```
####
##### process parameters in
/opt/SMAW/SMAWcmx/lib/cmxsnmp/AgentTraces.rc ####
####

# TRAPS     # trace trap generation
trace is OFF, do you want to switch ON? y| n | q (default: n)  y

activate updates by restarting CMX Subagent? y| n (default: y) n
```

End status

0

cmxsnmpadm has been successfully executed.

≠0

An error has occurred during execution of *cmxsnmpadm*.

Files

/opt/SMAW/SMAWcmx/bin/cmxsnmpadm
The *cmxsnmpadm* command.

/opt/SMAW/SMAWcmx/lib/cmxsnmp/AgentParams.rc
Contains the local administration parameters.

/opt/SMAW/SMAWcmx/lib/cmxsnmp/AgentTraces.rc
Contains the trace points.

Making changes using an editor

1. Use any editor to make the required changes in the relevant file (*AgentParams.rc* or *AgentTraces.rc*).
2. If you want to activate the changes immediately, call *cmxsnmp restart*. The changes then apply to the current CMX subagent. Otherwise, the changes take effect automatically the next time the CMX agent is started.

Changing the *MAX** parameter during operation invokes a restart of the CMX subagent.

12 Using TLI applications

CMX supports the use of TLI applications, the transport systems provided by the product groups CCP-WAN and CCP-ISDN.

TLI is a program interface in the Solaris operating system, which is related to XTI (X/Open Transport Interface).

TLI applications can run on the CCP transport systems provided the following conditions are fulfilled:

1. The TLI application must be designed to run on any ISO transport system. For example, before connection setup, it must ensure that all sent data has been received by the partner, since ISO transport systems do not recognize the concept of orderly release.
2. The TLI application must use the network selection mechanism provided in System V Release 4 (via */etc/netconfig*), as well as the NETDIR functions for mapping names and addresses.
   It must not make any assumptions on the name of the transport service or on the address format of the transport system.

The configuration file /etc/netconfig

The Solaris base system comes with a network configuration file which helps TLI applications to select the right transport system. A description of this file and of the selection mechanism can be found in the manual “XTI, X/Open Transport Interface” [2]. The system administrator must extend this file to include the following entries:

For ISO transport services:

Network ID: cx-osicots

Semantics: tpi_cots

Flag: –

Protocol family: osi

Protocol name: –

Network device file: /dev/osicots3

Reference libraries: /usr/lib/tnsxaddr.so

For NEA transport services (without NEABX):

Network ID: cx-nea

Semantics: tpi_cots

Flag: –

Protocol family: nea

Protocol name: –

Network device file: /dev/neat3

Reference libraries: /usr/lib/tnsxaddr.so

For ISO and NEA transport services (without NEABX) together:

Network ID: cx-msg

Semantics: tpi_cots

Flag: –

Protocol family: msg

Protocol name: –

Network device file: /dev/msg3

Reference libraries: /usr/lib/tnsxaddr.so

The table below shows the CCP profiles associated with the transport services:

Transport service	**CCP profiles**
ISO	CCP-ISDN-CONS CCP-WAN-CONS
NEA	CCP-ISDN-NEA CCP-ISDN-NX25 CCP-WAN-NEA CCP-WAN-NX25

Table 30: CCP profiles associated with transport services

Mapping names and addresses

Like ICMX applications, TLI applications require access to a name service which maps symbolic names to addresses and vice versa. TLI applications that use the NETDIR program interface work with two-part symbolic names, which identify the host and the service on the host. Once the */etc/netconfig* file has been extended as described above, TLI applications will be able to access the TNSX via the NETDIR program interface. TNSX entries are made in exactly the same way as for ICMX applications. However, the system administrator must observe the following rules when assigning GLOBAL NAMES:

1. GLOBAL NAMES for LOCAL NAMES (local address)

   The name parts NP1, NP2, and NP3 of the GLOBAL NAME must be left blank. NP4 must be set to the name of the local host, as supplied by the *uname -n* command. NP5 can be set as desired (but must not be left blank) and identifies the TLI application within the local end system.

2. GLOBAL NAMES for TRANSPORT ADDRESSES (remote address)

   The name parts NP1, NP2, NP3, and NP4 of the GLOBAL NAME can be set as desired (but at least one name part must be set). They identify the remote end system from the point of view of the TLI application. NP5 can also be set as desired (but must not be left blank) and identifies the TS application within the remote system.

Glossary

application

An application is a system of programs which implements a particular range of services of a DP system in order to provide a higher-quality service to the human or electronic user. Communication applications are applications that use the communication functions of a DP system in order to provide global services when a network is in operation.

Most applications are qualified by a prefix which identifies the underlying service range (*CMX application*, UTM application, DCAM application, Motif application, Windows application, etc.). Examples of communication applications are file transfer, terminal emulation, electronic mail, world wide web browser and server, transaction systems such as *open*UTM, and in general all applications based on the client-server principle.

API (application program interface)

APIs are program interfaces that provide the functions of a program system. As the programmer, you use the APIs when programming applications. APIs offer functions for connection management, data exchange, and mapping names to addresses. APIs in the CMX environment are ICMX, XTI,TLI and NLI.

CC (communications controller)

A CC is a component for connecting a Solaris system to a network. You need a CC to physically attach your system to a subnetwork, unless the interface is integrated on a different module, e.g. the motherboard (onboard interface).

In order to establish a logical connection to the network, the CCs are loaded together with the corresponding subnetwork profile. The subnetwork profile is a component of the *CCP*. PWXV, PWS0 and PWS2 are examples of loadable CCs for connecting to X.25, telephone networks and ISDN.

CCP (communication control program)

A CCP is a program system which, together with one or more *CCs*, provides the logical access of a Solaris system to a *network*. A CCP implements the four lower layers (transport system) of the OSI reference model for data communication.

A CCP consists of *subnetwork profiles* and *Transport Service Providers*.

CLI (command line interface)

The CLI is the sum of the *OA&M* commands of CMX and the *CCPs*. As the administrator, you can implement the initialization, monitoring, control, and maintenance functions of CMX, the *CCPs* and the *communication services* via the command line of the UNIX system (the commands cmxinfo, cmxm(onitor), tnsxcom, bstv, ccpgen, etc.).

CLIs offer a wide range of options, some with complex syntax. The user interface *CMXCUI* enables the desired routine actions to be performed simply and interactively.

CMX (Communications Manager UNIX)

CMX provides communication services for using **CMX applications** and **communication services** in the network, and enables the programming of CMX applications. CMX standardizes the services of different networks and thereby permits utilization of the same CMX application regardless of the underlying network. As the runtime system, CMX switches between the current network environment and CMX applications, and offers the network administrator uniform functions for *OA&M* (Operation, Administration, Maintenance) of *CCPs* and *CCs*. As a development system, CMX provides interfaces (APIs) and procedures for programming network-independent CMX applications.

CMX applications

CMX applications are applications that use the services of CMX. CMX applications have a network address known as the **TRANSPORT ADDRESS**. They are identified uniquely by means of a symbolic name, the **GLOBAL NAME** of an application.

CMXCUI (character user interface)

The CMXCUI is a character-oriented user interface to the *OA&M* functions of CMX and the *CCPs*. As the administrator, you can perform *OA&M* functions using menus and forms. CMXCUI uses FMLI and is based on the *CLI*.

communication services

Communication services are used for linking heterogeneous networks of various architectures or different technologies. You can implement the most diverse LAN-WAN connections using communication services, whereby the respective communication service is a software component on a server, for example.

FSS (forwarding support service)

The FSS is a component of CMX which supports the correct addressing of applications in the network and the selection of a route through the **network** and its subnetworks. As the administrator, you can configure the FSS with network-specific information which you have defined for your network or have agreed with the network operator.

Important information in the FSS includes the map of a network address, e.g. the NEA address “47/11”, to a subnetwork address of the remote system, e.g. the X.25 address “8963647658”. The definition of a route with its local starting point and the various stations through the subnetworks to the remote system is also key information. The local starting point of a route is called the **subnetwork ID**, which identifies one of a number of subnetwork interfaces.

GLOBAL NAME of an application

Each *CMX application* identifies itself and its communication partners in the network by symbolic, hierarchical GLOBAL NAMES. A GLOBAL NAME consists of up to five name parts (NP[1- 5]), which you can use to define the application (NP5), the processor (NP4), and (up to three) administrative domains (NP[3-1]).

Example: The GLOBAL NAME “YourApplication.D018S065.mch-p.sni.de” means:
“YourApplication” resides on the host “D018S065” in the domain “mch-p.sni.de”.

When you, as administrator, are choosing a GLOBAL NAME, you must adhere to the regulations and recommendations of the specific application.

As the administrator you assign a *TRANSPORT ADDRESS* or a *LOCAL NAME* of an application to the GLOBAL NAME of the application on a 1:1 basis. As the programmer, you can obtain the TRANSPORT ADDRESS or LOCAL NAME expected by CMX from the GLOBAL NAME using the function calls of the *transport name service* (TNS).

KOGS (configuration-oriented generator language)

KOGS is the configuration-oriented generator language with which the physical and logical properties of the subnetwork interfaces of a processor are described in a text file. Language elements of KOGS are macros, operands, and operand values. Normally, the system administrator or network administrator defines the specific properties of a subnetwork interface using the *CMXCUI*. KOGS is only used in exceptional cases.

LOCAL NAME of an application

A CMX application uses the LOCAL NAME to attach to CMX in its local system for communication. The LOCAL NAME comprises one or more *T-selectors*, which identify the transport system via which the CMX application is to communicate. As the administrator, you can enable or disable the communication of a CMX application via particular transport systems and fulfill any requirements of the CMX application for specific T-selector values, e.g. in file transfer.

Example: An application is to use the T-selector “cmxappl” (in lowercase letters!) for communication via the TCP/IP- RFC1006 transport system, and the T-selector “$CMXAPPL” (in uppercase letters!) for communication via the NEA transport system.

As the administrator in CMX, you can use the user interface *CMXCUI* to assign the LOCAL NAME of an application to the *GLOBAL NAME* of the application. As the programmer, you can obtain the LOCAL NAME expected by CMX from the GLOBAL NAME using the function calls of the *transport name service* (TNS).

network

A network is a set of interoperating communication components (lines, switching nodes, procedures) with uniformly defined services, protocols, and access equipment for DP systems. A network connects processors for the purpose of using global applications. The network of a network operator can be used immediately for applications or for defining

overlying, private network structures. The following networks are relevant in the UNIX environment: the Internet, SNA, TRANSDATA, and OSI networks.

A network can comprise one or more **subnetworks**, which are linked using the homogeneous end-to-end protocol of the network. The networks listed above as examples can be overlappings of public or private subnetworks such as the X.25 network, the telephone or data network, the ISDN or ATM network, and various private local networks based on Ethernet, Token Ring, and FDDI.

network address

Each processor in a *network* is uniquely identified by its network address. A processor can be integrated in different networks and has a specific network address for each of these networks.

In Internet, a network address is called an IP address. IP addresses are explicitely assigned to a IP interface. A single computer can have a number of IP interfaces. A single IP interface can only support IP version 4, IP version 6 or both IP versions 4 and 6 simultaneously. One IP address is assigned to each supported IP version on the interface (example of as IPv4 address: 129.144.89.171, example of an IPv6 address: fe80::280:17ff:fe28:7b08).

In the NEA network, a processor has an NEA network address consisting of the processor/region number (e.g. 124/213).

The OSI network address (NSAP address) is made up of the Initial Domain Part (IDP) and the Domain Specific Part (DSP), and has the format: IDP+DSP (e.g. 470058+0144458100007391100308001411961301).

OA&M (operation, administration, and maintenance)

OA&M is the sum of the functions for commissioning, operation monitoring and control, configuration, and maintenance of CMX and CCP components. Essential OA&M activities in the CMX environment include loading and monitoring a *CC*, configuring the runtime parameters of the *CCP*, and generating traces.

Routine OA&M actions can be performed simply and interactively using the *CMXCUI*. For special, extraordinary administration tasks, you can also use the *CLI*.

route

A route defines the path from the local system to a remote system within a **subnetwork**. If the remote system is located in a different subnetwork, the route defines the path from the local system to the network link (“next hop”), and routing continues from there to the remote system. A route is defined by its endpoints: the **subnetwork ID** in the local system and the **subnetwork address** of the remote system if the remote system is located in the same subnetwork, or the subnetwork address of the “next hop” if the remote system is located in a different subnetwork. If a system has several subnetwork addresses, it can be reached via several routes.

subnetwork

A subnetwork is a part of a **network** which is technically or administratively homogeneous. Subnetworks include the X.25 network, the telephone or data network, the ISDN or ATM network, and various private local networks based on Ethernet, Token Ring, and FDDI. A subnetwork can be accessed via one or more subnetwork interfaces. A subnetwork interface is identified by its **subnetwork address**.

subnetwork address

The subnetwork address uniquely describes a subnetwork interface which enables access to the **subnetwork**. The subnetwork address can be an ISDN dial number, a DTE address, or an Ethernet address, for example.

subnetwork ID

The subnetwork ID, also called the SNID, identifies a group of subnetwork interfaces of the same type which provide access to the same **subnetwork**. The subnetwork ID specifies the type of subnetwork and identifies the group of interfaces to this subnetwork. A subnetwork can stand for two ISDN interfaces or a number of X.25 interfaces in a subnetwork, for example.

subnetwork profile

The subnetwork profile identifies the components of a *CCP* which control a *Communications Controller*.

SWC (software configuration)

An SWC is a defined combination of versions of software products which together cover a limited and verified performance range.

An SWC of CMX and *CCP* product versions guarantees that they will interoperate in a defined way. If you mix CMX and CCP product versions that are not defined as an SWC or are not expressly identified as compatible, unexpected malfunctions and failure situations may occur with undefined consequences.

TNS (transport name service)

The TNS is a component of CMX which supports the correct mapping of the *GLOBAL NAMES* of *CMX applications* in the network to *TRANSPORT ADDRESSES* and *LOCAL NAMES*. As the administrator, you configure your chosen assignment of GLOBAL NAME to TRANSPORT ADDRESS for remote applications, as well as the assignment of GLOBAL NAME to LOCAL NAME for local applications. As the applications programmer, you can use these maps via an *API* and thereby work solely with the GLOBAL NAMES of applications without assessing the maps.

The TNS provides network-wide identification of applications by means of logical GLOBAL NAMES and their mapping to corresponding *network addresses*. This means that you can identify applications without having to know their network addresses. Together with the *FSS*, the TNS provides a complete mapping of the logical name to a concrete *subnetwork address* and a *route* through the various subnetworks of the network.

TRANSPORT ADDRESS of an application

A calling *CMX application* transfers the TRANSPORT ADDRESS of a called communication partner to CMX when communication is being established. CMX uses the TRANSPORT ADDRESS to locate the communication partner in the network and determine a **route** through the network. The TRANSPORT ADDRESS generally depends on the logical and physical structure of the network (and its subnetworks). The TRANSPORT ADDRESS contains the specifications of your network operator(s) which are specific to your network. As the administrator, you can influence the TRANSPORT ADDRESS and hence the communication paths independently of the application.

The components of a TRANSPORT ADDRESS are: a network address for uniquely identifying the remote system on which the application resides, the type of *transport system* via which the remote application can be reached, and the *T-selector* that identifies the remote application in the remote system.

As an administrator, you can assign a TRANSPORT ADDRESS of an application to the *GLOBAL NAME* of an application one-to-one.

As a programmer, you can obtain the TRANSPORT ADDRESS expected by CMX from the GLOBAL NAME using the function calls of the *transport name service* (TNS).

transport system

The transport system is represented by the four lower layers of the *OSI Reference Model*. A *CCP* implements the four layers of the transport system. The transport system guarantees the secure exchange of data between systems whose *applications* communicate with each other, regardless of the underlying network structures. The transport system uses protocols for this purpose.

T-selector

The T-selector identifies a communication application within the system on which the application is running. Together with the *network address* of the system, the T-selector forms the *TRANSPORT ADDRESS* of an application which uniquely identifies this application within the network. The format and value range of the T-selector depend on the type of **network**. In the NEA network, the T-selector corresponds to the station name (e.g. T'DSS01').

TSP (transport service provider)

A TSP is a component of a *CCP* or of CMX which, with the exception of the NTP (null transport), provides the OSI transport service in the network using a transport protocol. As the administrator, you can determine the usage of a particular TSP for the communication of *applications*. RFC1006 is the TSP in CMX which, together with TCP/IP, provides the OSI transport service in the Internet. NTP (null transport) offers *CMX applications* direct access to the network services of the X.25 subnetwork. TP0/2, and NEA are the TSPs for an OSI environment and the TRANSDATA network.

Together with a *subnetwork profile*, a TSP forms a *transport system*. It offers a set of configurable runtime and tuning parameters, assesses the *TRANSPORT ADDRESS*, and finds a suitable route through the network. To do this, the TSP uses your specifications in the *FSS*, if necessary.

Abbreviations

ASCII
American Standard Code of Information Interchange

CC
Communications Controller

CCITT
Comité Consultatif International Télégraphique et Téléphonique

CCP
Communication Control Program

CMX
Communications Manager for UNIX Systems

CMXCUI
CMX Character User Interface

DCAM
Data Communication Access Method

EBCDIC
Extended Binary Coded Decimals Interchange Code

ETHN
ETHERNET

ETSDU
Expedited Transport Service Data Unit

FSB
Forwarding Support Base

FSS
Forwarding Support Servoce

FT
File Transfer

ICMX
Programming Interface CMX

ISDN
Integrated Services Digital Network

ISO
International Organization for Standardization

KD
Configuration file

KOGS
Configuration-oriented generator language

LAN
Local Area Network

MIB
Management Information Base

MT
Multi-Threading, multi-threaded

NEA
Network architecture for TRANSDATA systems

NLI
Network Layer Interface

NSAP
Network Service Access Point

OSI
Open Systems Interconnection

PDN
Public Data Network

PID
Process Identifier

PSDN
Packet Switched Data Network

PSTN
Packet Switched Telephone Network

PVC
Permanent Virtual Circuit

SNA
Systems Network Architecture

SNID
Subnet identification

SNPA
Subnet Point of Access

SVC
Switched Virtual Circuit

TCEP
Transport Connection Endpoint

TCP/IP
Transmission Control Protocol/Internet Protocol

TEP
Transport Endpoint

TIDU
Transport Interface Data Unit

TLI
Transport Layer Interface

TNS
Transport Name Service

TPDU
Transport Protocol Data Unit

TPI
Transport Provider Interface

TREF
Transport Reference

TS
Transport Service

TSAP
Transport Service Access Point

TSDU
Transport Service Data Unit

TSP
Transport Service Provider

TSTAT
TEP Status

WAN
Wide Area Network

XTI
X/OPEN Transport Interface

Related publications

The manuals are available as online manuals, see *http://manuals.fujitsu-siemens.com*, or in printed form which must be payed and ordered separately at *http://FSC-manualshop.com*.

[1] **CMX V6.0**
Programming Applications

Target group
Programmers

Contents
The manual describes the program interface of CMX, i.e. all tools that you can use for developing TS applications.

[2] **XTI V6.0**
X/Open Transport Interface
User Guide

Target group
Programmers of TS applications.

Contents
The manual contains implementation-specific supplements to the function calls of XTI.

[3] **CMX/CCP V6.0** (Solaris)
ISDN Communication
User Guide

Target group
Network administrators

Contents
The manual describes the computer-to-computer connection via ISDN (Integrated Services Digital Network).

[4] **CMX/CCP V6.0** (Solaris)
WAN Communication
User Guide

Target group
Network administrators and system administrators

Contents
The manual describes the computer-to-computer connection via WAN (Wide Area Network) allowing communication in the remote area (Wide Area Network, WAN).

[5] **CMX V6.0** (Solaris)
TCP/IP via WAN/ISDN
User Guide

Target group
Network and system administrators.

Contents
The manual describes how CMX makes possible the connectionless IP traffic via the connection-oriented WAN.

[6] **Interfacing to SNA Networks**
TRANSIT-BAS
Core Manual

Target group
Solaris users in SNA networks

Contents
Basic description of the TRANSIT products

[7] **System Administration Guide, Volume 2** (Solaris 8/9)
System Administrator Manual

Target group
Solaris system administrators

Contents
Introduction to system administration of Solaris

[8] **openNet Server V3.0** (BS2000/OSD)
IPSec V1.0
User Guide

Target group
The manual is targeted at network administrators, developers of network applications in a BS2000/OSD environment and all those interested in questions of Internet security, particularly in a BS2000/OSD environment.

Contents
After the general overview of threats to Internet security, the manual describes the concept of the IPSec protocol in detail. The manual also describes the implementation of IPSec in BS2000/OS, supplying all the necessary information about installing, configuring and operating the IPSec subsystem in BS2000/OSD.

Other publications

[9] **WebSysAdmin/DomainAdmin V2.1**
System Administration within a Domain

Target group
Solaris system administrators

Contents
System Administration within a Domain.

[10] **[Sol_LU]**
Solaris Live Upgrade 2.0 Guide
SUN Microsystems October 2001

Target group
Solaris system administrators

Contents
Description of Solaris Installation per Live Upgrade.

Index

T

Fujitsu Siemens Computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

email: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on CMX V6.0B (Solaris)
Operation and Administration

Fujitsu Siemens Computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

email: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on CMX V6.0B (Solaris)
Operation and Administration

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009