FSM CALA User's Guide

FileNet System Monitor 4.0.0

FileNet Corporation

Table of Contents

List of Tables

List of Figures

List of Examples

List of Screenshots

Chapter 1. Copyright Notice

FileNet System Monitor

(June, 2007)

Trademarks

The following product names are trademarks of Tivoli Systems or IBM Corporation: AIX, IBM, OS/2, RS/6000, Tivoli Management Environment, TME 10, Tivoli, Tivoli Enterprise Console (T/EC).

Microsoft, Windows, Windows NT, Windows 95 and the Windows logo are trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Hewlett Packard, HP, and HP-UX are trademarks or registered trademarks of Hewlett Packard Corporation.

Other company, product, and service names mentioned in this document may be trademarks or servicemarks of others.

Notice

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to Tivoli Systems or IBM's valid intellectual property or other legally protectable right, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user.

CENIT AG Systemhaus may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the

CENIT AG Systemhaus, Product Marketing Tivoli Plus Modules, Industriestr. 52-54, 70565 Stuttgart, Germany

Chapter 2. Notices

FileNet Corporation 3565 Harbor Boulevard Costa Mesa, California 92626 USA
800.FILENET (345.3638) Outside the U.S., call: 1.714.327.3400
www.filenet.com (http://www.filenet.com)

Chapter 3. About this document

Who Should Read This Guide

The target audience for this guide are system administrators who use the FSM CALA. Users of the guide should have some knowledge about Unix and/or Windows operating system and the FSM CALA.

List of documents

FileNet System Monitor CALA Guide

Datatypes that can be processed by the FSM CALA

FileNet System Monitor Monitoring Guide

Description of all monitors contained in FileNet System Monitor

FileNet System Monitor Task Guide

Description of all tasks contained in FileNet System Monitor

FileNet System Monitor Users Guide

Installation guide

FileNet System Monitor Release Notes

Description of changes and bugfixes

General information

Where you find this guide

You can find this documentation on the FSM installation CDROM in the following folder:

UNIX: `<Mount point>/INSTALL/docs`

Windows: `<Drive letter>:\INSTALL\docs`

Typeface Conventions

The guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

code Keywords and code examples occur like this

`varname` Variable names occur like `this`

`filename` File names occur like `this`

`constant` Constants and names of tasks, monitors etc. appear like `this`

command Command names appear like **this**

`parameter` Parameters and options for commands apperar like *`this`*

`userinput` Values that th user must provide appear like **`this`**

`Computer output` Output from programs appears like `this`

guilabel Names of windows, dialogs, and other controls appear like this

Programlistings appear like this:

```
# a program listing
echo "This is an example program listing (shell script) with nothing bu ↙
t an extremly long echo command"
exit 0
```

> **Note:** The character ↙ at the end of a line in a `computer output` or program listing shows, that the line has been wrapped and is continued in the next line.

Contacting FileNet Support

We are very interested in hearing from you about your experience with the product. We welcome your suggestions for improvements.

If you encounter difficulties with the FSM please contact the FileNet support (http://www.filenet.com).

Chapter 4. FSM CALA Overview

FSM CALA (CALA)

CALA consists of the following components

- CALA binaries The programs which implement the different CALA functions
- CALA GUI The graphical User Interface to design, test and create CALA environments

CALA Binaries

- Receive (read) events from different event sources (active and passive event management, Logfile, Event log, Syslog, SNMP, active Monitoring)
- Process events (filtering, manipulation, correlation, formatting)
- Sends events to different destinations (T/EC, SNMP manager, SMPT email, report files, execution of commands)

CALA GUI

- Used to design a layout of CALA components (CALA architecture)
- Design testing

Chapter 5. Installation

General Installation Information

This chapter describes the CALA installation process.

Non-Tivoli CALA Installer

This version of CALA supports the graphical installation of CALA on all supported platforms.

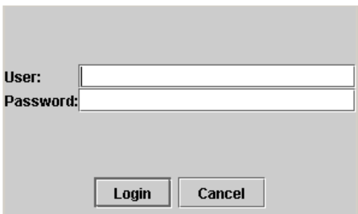
The CALA Installer can be invoked by executing the script `setup.sh` on the FSM CDROM. On Windows platforms you can use the Batch program `setup.bat` to start CALA Installer.

- For Windows JRE 1.4 is installed on the CD
- Non-Windows users need JRE or JDK 1.4 to start the CALA installation GUI.
- You need not set the environment variable `JDK` if the java binary can be found in your `PATH`.

General description

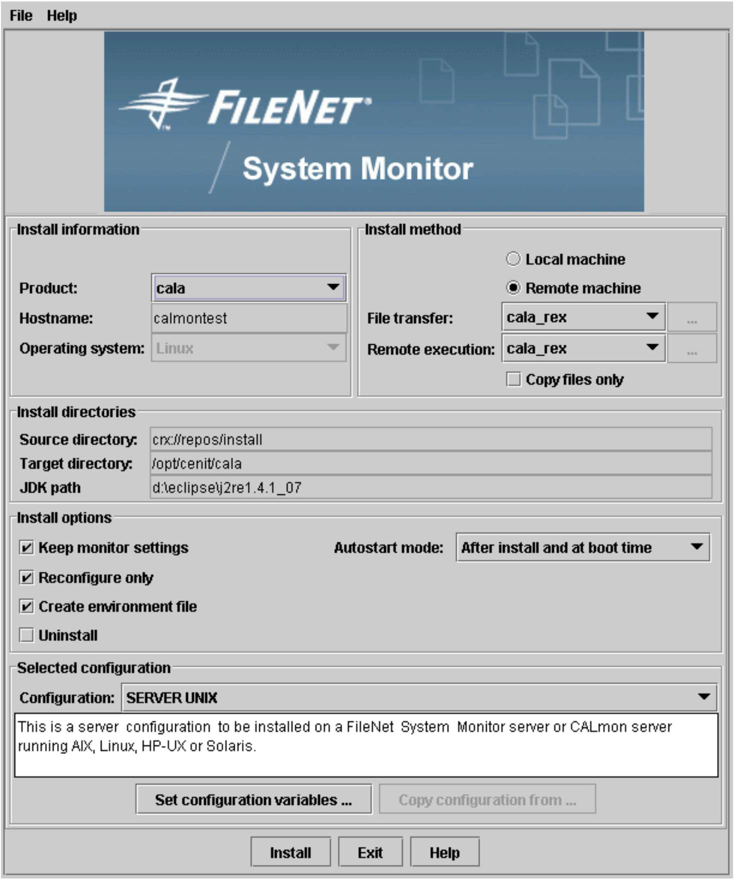
`setup.sh` or `setup.bat` starts the CALA installation GUI. CALA Installer is a Java GUI interface for the installation script `install_cala.sh` (see Annex for further information about `install_cala.sh`).

If the CALA installer is started from the WebConsole in a full FSM environment, you must login to the cala_rex server before the installer window is shown.



Installer login screen

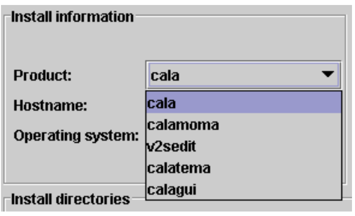
This is the installer main window:



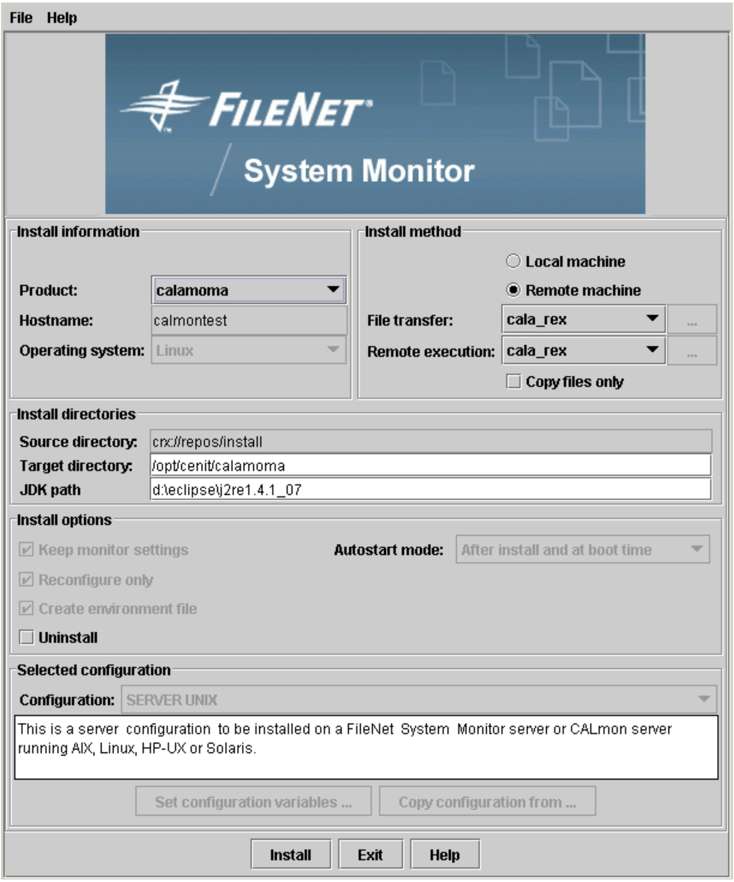
The installers' main window

The CALA installation GUI is used to install CALA itself and its associated tools CALAGUI, V2SEdit and CalaMoMa.

The product is selected from the `product` listbox. Depending on product selection, some of the GUI components are disabled.



Selecting the product in the installer main window



The installers main window for installation of calamoma

The configuration selection area and the CALA cache dir entry field are only enabled if cala is selected, they are not needed for the installation of any CALA tool.

The JDK path is only editabled for the java tools calamoma, v2sedit and calagui.

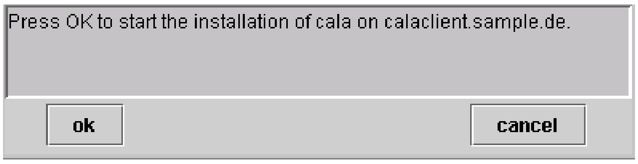
The source directory is shown for informational purpose only and is not editable in any case.

Installing a product on the local machine

To install a product on the local machine, the following steps have to be done:

- select a product
- customize target directory and possibly JDK path
- CALA only: you may choose a default configuration from the configurations listbox and set its parameters in the settings dialog which appears after pressing the Set configuration variables button. You can copy and adjusrt an existing configuration by pressing the Copy configuration from ... button.
- press the Install button

This will show the installation confirmation dialog window.

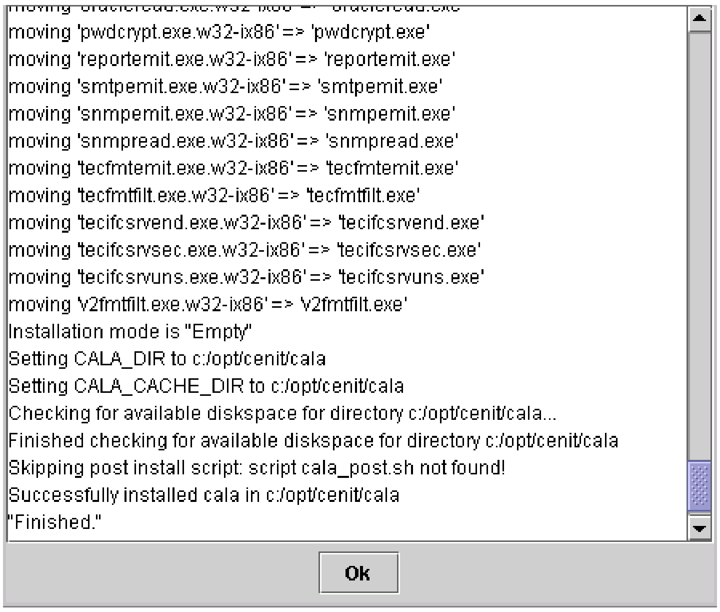


The installation confirmation dialog

If the installation has been confirmed, the installation files are transferred from the source (cd or ftp server) to the target directory,

The file transfer progress window

and the installation process is started. The installation progress is displayed in the results window. The button of this window is enabled only after the installation process has terminated (with or without success).



The installation result window

Remote installation

The CALA installer is enabled to install products on a remote machine if the following preconditions are fulfilled:

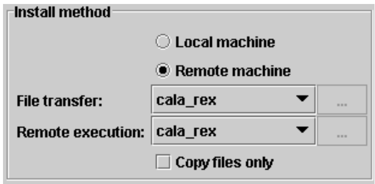
- the target directory on the remote machine can be accessed via a local mounted drive or directory, ftp or cala_rex .
- the target machine allows rlogin, telnet or cala_rex to start the installation process. If the target directory is accessible, but no remote login is allowed (neither rlogin nor telnet nor cala_rex), at least the installation files can be copied to it (see description of the Copy files only checkbox below).

If the CALA installer is started directly from CD, the File transfer method box contains the option Connected drive/dir only. The Remote execution method box contains the option Rlogin only.

If the CALA installer is started from the WebConsole in a full FSM environment, the File transfer method box contains the options cala_rex and Connected drive/dir. The Remote execution method box contains the option cala_rex only.

The configuration can be changed to allow more protocols if required.

To install on a remote machine select the remote machine radio button. This enables the comboboxes for Filetransfer and Remote execution as shown in the screenshot below



The installer remote settings panel

The Install method area contains two lines: the first line configures the file transfer to the target host and the second configures the remote execution.

If the Copy files only checkbox is selected, the installation files are only copied to the target directory without starting the installation process. A setup script or batchfile is created which can be run manually to complete the installation. A dialog box shows how the installation can be completed.

The file transfer can be done either by ftp, via a locally mounted drive (Windows system) or directory (nfs on unix) or via cala_rex. Select the transfer type from the listbox.

If ftp is selected, a user and password for accessing the remote host must be given. When choosing Connected drive/dir , the mount point (or drive) on the installing host and the drives or directories original name on the target host are needed. These additional parameters can be entered after pressing the ... button to the right of the Filetransfer combobox.

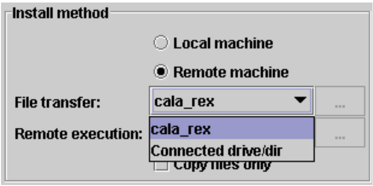
When choosing cala_rex , no further parameters are needed.

CALA should be installed on drive `C:` on a remote host. This drive is connected to `Z:` on the installing machine, so `Z:` is the mount point and `C:` is the original drive.

Example 5-1. Example for using Connected drive/dir on MS Windows

CALA should be installed in directory `/opt/FileNet/SysMon/cala` on a remote host. The directory `/opt/FileNet/SysMon` is connected to `/mnt` on the installing machine, so `/mnt` is the mount point and `/opt/FileNet/SysMon` is the original directory.

Example 5-2. Example for using Connected drive/dir on Unix

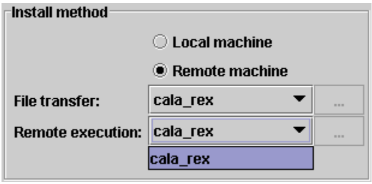


Choosing the file transfer method

The second line sets the remote execution protocol telnet, rlogin ot cala_rex.

If Rlogin is selected, a password may not be needed the appropiate field can therefor be left empty. If telnet is selected, a user and password for accessing the remote host must be given. These additional parameters can be entered after pressing the ... button to the right of the Remote execution combobox.,

When choosing cala_rex , no further parameters are needed.



Choosing the remote execution methd

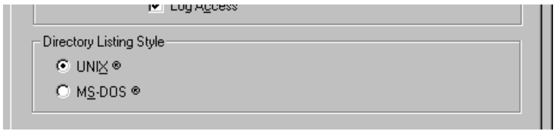
To start the installation, perform the following actions:

- select a product
- select the remote machine button

- enter the ftp user and password or the mount point and original directory if required
- enter the telnet or rlogin user and password if required
- customize target directory and possibly JDK path
- CALA only: you may choose a default configuration from the configurations listbox and set its parameters in the settings dialog which appears after pressing the Set configuration variables button.
- press the Install button

The following installation process is the same as for local installation, see above for details.

Information for users of the Microsoft Windows ftp server and telnet servers: When the Microsoft Windows ftp server (part of the internet information server IIS) is used, it must be ensured, that the directory listing style is set to Unix for all directories accessed by the CALA tools.

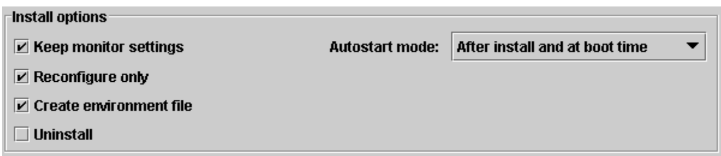


Setting MS Windows ftp server to UNIX mode

The Windows telnet server must be configured not to use NTLM authentification. The NTLM authentification parameter must be set to 0. (Use the tlntadmn tool to configure the telnet server.)

Further installation options

The options located in the Install options panel control system-specific settings such as autostart.



CALA installer further options

Keep monitor settings

If this checkbox is selected, the current monitoring configuration on the client will not be replaced by original monitor settings contained in the selected configuration archive.

Reconfigure only

If this checkbox is selected, the binaries will not be transferred during the installation process. The configuration file will be recreated according to the settings in the Set configuration variables dialog.

Create environment file

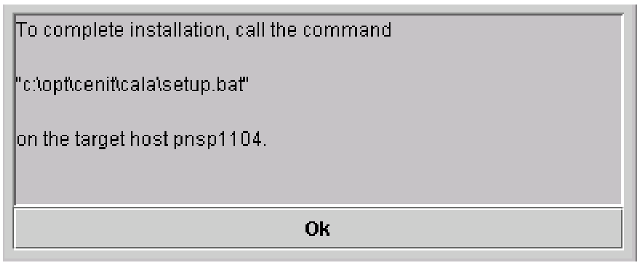
Tells the installation script to create the environment file `set_cenit_env.sh` in the directory `/etc/cenit` (on UNIX) or `/etc/cenit` (on Windows) or in a subdirectory of this directory.

Uninstall

Check this option to remove the product from the selected client.

Autostart

- Select After installation and at boot time to create the links that are required to start CALA at boot time (UNIX) or to register the CALA service for automatic startup (Windows). CALA will be started after successful installation as well.
- Select After installation to start CALA only after installation.
- Select None if CALA should not be started automatically at all.



CALA installer informational messagebox for completing installation

The Unpack binaries checkbox is available for CALA only and is selected by default. If it is unchecked, only the CALA configuration, but not the binaries are copied or updated at the target host. This feature is useful when using the CALA Configurator and should only be used by experts.

Select the Verbose mode box to get further information from the installation process. This may be helpful if the installation fails.

Relationship between installer GUI install_cala.sh

The following table shows the relation between the parameters and the command line options of install_cala.sh.

parameter	corresponds to	Default value (if any)
Product	-product	
source directory	-sourcedir	subdirectory `Images` on the same level as current directory; if `../Images` does not exist, current directory

parameter	corresponds to	Default value (if any)
Target directory	-targetdir	current directory
CALA cache dir:	-cachedir	current directory
JDK path	-jdkdir	Directory where the java binary used to start the CALA installation GUI is located
Unpack binaries	-untar	
Uninstall product	-remove	(not specified)
Keep monitor settings	-keepmonitors	(not specified)

Table 5-1. Relationship between GUI parameters and command line options of install_cala.sh

For details see description of install_cala.sh in *Annex 9: Additional tools*.

Chapter 6. Component Architecture

CALA is realized as a multi component Client/Server architecture, which enables customers to realize any kind of centralized and distributed Logfile and monitoring architecture. Almost all components are available on a comprehensive list of platforms (see restrictions on below site).

CALA System platforms

For detailed information about supported server and client platforms check the latest release notes.

Supported JAVA JRE or JDK versions (CALAGUI and CALA V2S Editor prerequisite)

For detailed information about required Java JRE or JDK versions for JAVA tools check latest release notes.

Implementation on Microsoft Windows based systems

Implementation of CALA on the Windows system platform has been implemented as an Windows Service.

CALA installation as Windows Service

Installation of the Windows Service is performed using the program **cala_srv.exe**.

Installation with start mode manual : **cala_srv.exe** *`-install`*

Installation with start mode automatic : **cala_srv.exe** *`-auto`*

CALA de-installation on Windows systems

To remove the CALA Windows service start **cala_srv.exe** *`-remove`* from the command line.

Configuration file logctlsrv.conf for a Windows service installation

If CALA is installed as an Windows service, configuration file `logctlsrv.conf` must either be placed in directory/folder `%SystemRoot%\system32\config` or in a directory/folder of your choice, which is mapped to the environment variable CALA_DIR of the Windows system environment.

The CALA Windows service reads environment variables CALA_DIR and CALA_CACHE_DIR out of the registry (registry key

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cala_srv) if they are not mapped in the environment.

If all registry keys are set properly (see chapter Tivoli integration, Post distribution: CALA installation for details) there is no need to reboot the Windows system.

> **Note:** The CALA processes can also be started as a normal program instead of an Windows service. In this case the CLI-program **logctlcmd** (refer to description) should be used should be used for starting and stopping of the CALA components.

> **Note:** On Windows, CALA needs the file `PSAPI.DLL` to be available on the system. The file is automatically installed with CALAin the CALA installation directory and must not be removed.

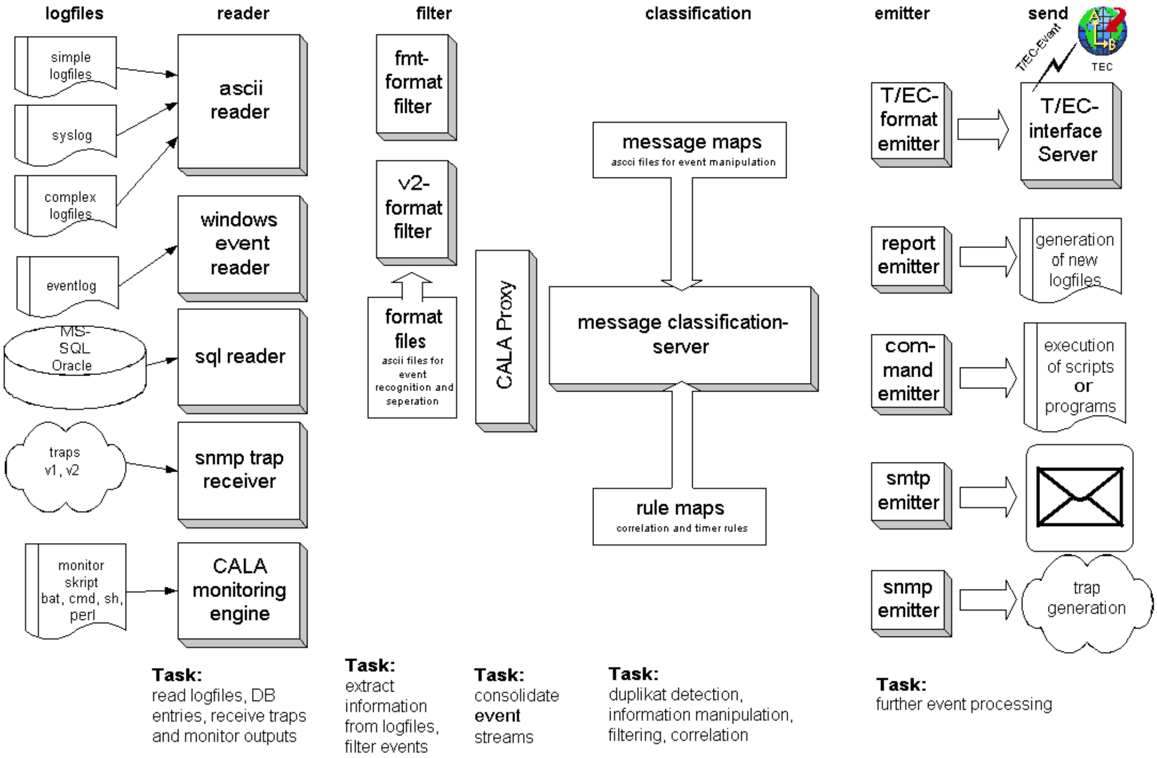
Client / Server Architecture

To satisfy various requirements (source-independent duplicate recognition, performance, configuration during operation), a specific client/server architecture was developed for the CALA system which distinguishes between the following functions:

- Reading of event sources
- Event filtering
- Classification and duplicate recognition
- Transmitting data (e.g. sending events to the T/EC)

All component of this architecture can be implemented on various systems, or just on a single system (computer).

This diagram illustrates a possible architecture of the CALA system:



This open architecture enables CALA to be implemented at almost any desired level of complexity or heterogeneity.

In addition, the CALA firewall component calaproxy can be interposed between any FIR-based component.

Implemented components

Read-only component (Reader)

Readers can be used to read event sources. Event sources can be regular files (Logfiles) or pipes, but also Windows Event Logs. The following readers are components of the CALA module:

Component name	Component type	Description
ascfileread	ASCII File reader	This component reads files and pipes available in ASCII format.
ntevtlogread	Windows Event Log reader	This component reads out the Windows Event Log.
mssqlread	MS SQL database reader	This component reads logfiles written into a ms sql database.
oracleread	Oracle database reader	This component reads logfiles written into a oracle database.
jdbcread	JDBC database reader	This component reads logfiles written into a database accessible via JDBC.

For all readers there is a special CLI parameter (`-E`), which should be used whenever the intention is solely to process events from the sources being read and which have been written to (generated) since the adapter was started.

Note: As an option, other readers can be implemented on a customer-specific basis.

Filter component (Filter)

Filters are used to disassemble data streams, which were read out of event sources by readers. In all cases, readers are only able to transmit to filters because only they have the ability to disassemble unstructured data streams into events (FIRs) based on format definitions (format files).

In technical terms, filters are arranged between the reader and the processing process or the emitter process.

Component name	Component type	Description
tecfmtfilt	T/EC Format filter	Interprets and classifies input from components ascfileread and ntevtlogread based on Tivoli .fmt files. This protects existing customer investment in format files.
v2fmtfilt	Complex filter	This component filters and interprets input from reader ascfileread based on the CALA-format description.

Event generating components

Component name	Component type	Description
snmpread	SNMP trap receiver	Receives SNMP traps and forwards them as CALA events (FIRs) to the specified targets.
calamon	Monitoring engine	Executes monitoring scripts/programs and generates events (FIRs) depending on return codes or output.

Processing server msgclsfsrv (Message Classification Server)

Correlation system msgclsfsrv is the brain behind CALA and is used for duplicate detection, event handling and computing (basic forms of computing).

Event handling includes the functions of event suppression as well as escalation (change in severity).

Component name	Component type	Description

Component name	Component type	Description
msgclsfsrv	Classification server	This component is the brain behind CALA. It contains the functions of classification, duplicate detection, event suppression and escalation.

Sub components Rules Engine and Message Mapping (msgclsfsrv sub components)

The rules engine and the massage mapping component are subcomponents of the message classification server and are used to process correlating events and timer on events as well as any manipulation on events.

For detailed information about the Rules Engine and the Message Mapping Component refer to related sub chapters within this chapter.

Sub component Completer (msgclsfsrv sub component)

The sub component Completer in process msgclsfsrv is used for downstream (i.e. after processing by the central processing server) bulk setting or deleting of slots (Tivoli T/EC slots) as a function of other existing or unmapped slots, or to fade out slots.

Language constructs such as *if!*, *unless!* or *for!* are implemented for processing purposes based on the existence (if) or absence (unless) of slots.

Application area:

Mapping of unmapped default slots, e.g. severity.

Fading out of mapped slots, e.g. those required for internal processing.

Sub component Remapper (msgclsfsrv sub component)

Sub component Remapper in process msgclsfsrv is capable of re-mapping slot contents, or of defining new slots.

This enables the system to rename event class names in a customer-specific manner.

Example:

Changing LogfileBase to <customername>_LogfileBase.

Comment: the event class <customername>_LogfileBase must be defined in a BAROC file.

Emitter components

Emitters (Senders) are used for sending or subsequently processing events, e.g. after a report has been raised. The following emitters/senders are available:

Component name	Component type	Description

Component name	Component type	Description
tecfmtemit	T/EC-Event-Preparation	This component prepares the logs to be processed and sends them to the T/EC transmit component (see below).
cmdemit	Task Engine	This component is capable of executing any commands. Parameters are read out of Message Map files.
reportemit	Reporting Emitter	This component is used for all the events. Events may be reported in T/EC Event dump format or in a own format specified in a template file.
snmpemit	SNMP Trap Emitter	This components throws SNMP events from received CALA FIRs.
smtpemit	SMTP Emitter	This component sends emails to report the received CALA FIRs.
jdbcemit	Database Emitter	This component writes events to a database accessible via JDBC.

T/EC transmit component

Component name	Component type	Description
tecifcsrv	T/EC Interface Server	This component sends prepared events to the T/EC. There are 3 variants of this component: • Secure Version: oserv communication (ManagedNode) • Unsecure Version: TCP/IP communication (EIF) • Endpoint Version: Tivoli TMA communication

Application proxy for DMZ

Component name	Component type	Description
calaproxy	Application proxy	This component is used as an application proxy in Demilitarized Zones (DMZ). This sends received FIRs to a downstream component on a computer on the far side of a firewall.

Control component logctlsrv

Component name	Component type	Description

Component name	Component type	Description
logctlsrv	Control server	This component is used to control and configure all other CALA components.

Logctlsrv is controlled using the CLI (command line interface) **logctlcmd**. **Logctlcmd** reads configuration information from the file `logctlsrv.conf` and starts the process **logctlsrv**, which then takes control of the configuration management of all other CALA component.

CLI logctlcmd

The Command Line Interface **logctlcmd** is used on all platforms supported by external control of the CALA component.

> **Note:** Starting and stopping the CALA component on Windows systems can be implemented by the Windows Service Manager (refer to CALA installation as an Windows service) as well.

Supported logctlcmd commands

startup

Starting the CALA component. A CALA Windows installation is started by the command **net start cala_srv** if CALA is installed as a service.

shutdown

Stopping the CALA component. A CALA Windows installation is stopped by the command **net stop cala_srv** if CALA is installed as a service.

restart

Restarts the CALA component

status

Status query of the CALA component. In addition to the output of all status information for the installed CALA component, output includes information about the Tivoli environment employed as well as important CALA environment variables.

reconfigure

Reconfiguration of the CALA component during runtime. Changes to the configuration (configuration file `logctlsrv.conf`) are taken into account at this time.

maintenance_on

Activation of the Maintenance Level (processing is delayed)

maintenance_off

Deactivation of the Maintenance Level (processing is restarted) test <logical_name>

This command is employed in order to generate a CALA test event from a component. This test event can be used to test communication between the component (local on a computer or on a remote computer).

Note: The CALA programs need a shared library which has to be available to them. See the following table for the name of the library and the environment variable to be set to it s path.

operating system	filename of shared library	environment variable
MS Windows	libcala.dll	`PATH`
AIX	libcala.so	`LIBPATH`
Solaris	libcala.so	`LD_LIBRARY_PATH`
Linux	libcala.so	`LD_LIBRARY_PATH`
HP-UX	libcala.sl	`SHLIB_PATH`

Generating test events

Apart from the emitters (senders), all components are able to generate test events. This means that communication can be tested between the individual components. Test events can be generated using the CLI call:

logctlcmd *`test <logical componentname>`*

Possible component architecture (predecessors / successors)

The following table illustrates the possible component architecture with predecessors (previous stage) and successor (subsequent stage).

Componentname	predecessor	successor
ascfileread	-	tecfmtfilt, v2fmtfilt
ntevtlogread	-	tecfmtfilt v2fmtfilt
tecfmtfilt	ascfileread, ntevtlogread	any FIR processing component[a]
v2fmtfilt	ascfileread, ntevtlogread	any FIR processing component[a]
snmpread	-	any FIR processing component[a]
calamon	-	any FIR processing component[a]
mssqlread	-	any FIR processing component[a]
oracleread	-	any FIR processing component[a]
jdbcread	-	any FIR processing component[a]

Componentname	predecessor	successor
javasrv / pchread	-	any FIR processing component[a]
msgclsfsrv	any FIR generating component[b]	any FIR processing component[a]
cmdemit	any FIR generating component[b]	-
reportemit	any FIR generating component[a]	-
snmpemit	any FIR generating component[a]	-
smtpemit	any FIR generating component[a]	-
jdbcemit	any FIR generating component[a]	-
tecfmtemit	any FIR generating component[a]	tecifcsrv (end, sec, uns)
calaproxy	any FIR generating component[a]	any FIR processing component[a]
remote component	any FIR generating component[a]	any FIR processing component[a], tecifcsrv (end, sec, uns)
tecifcsrv (end, sec, uns)	tecfmtfilt	-

Notes:

a. FIR processing components are:

- msgclsfsrv
- cmdemit
- reportemit
- tecfmtmit
- calaproxy
- snmpemit
- smtpemit
- jdbcemit
- remote component

b. FIR generating components are:

- tecfmtfilt
- v2fmtfilt
- calamon
- snmpread
- msslread, oracleread, jdbcread
- javasrv / pchread
- msgclsfsrv
- calaproxy
- remote component

Communication between CALA components

Communication between individual CALA components is based on TCP/IP communication with variable package size.

The data records read in (Logs, Windows Event Log, Syslog, etc.) are transferred by the filter processes to Filter Input Records (FIR) which form the basis for communication between all other CALA components.

When this standardized data object (FIR) is implemented for CALA component communication, CALA components are able to link up in almost any conceivable order.

The data (FIRs) can be transmitted through ports configured in any desired manner, and every component can also receive or transmit data via any desired number of ports.

Default tcp ports used by CALA components

The following table shows the default tcp ports used by CALA components. Chapter 10 "Configuration file logctlsrv.conf" describes how the port settings can ne changed.

component name	default port
logctlsrv	23861
logctlcmd	23860
ascfileread	23831
ntevtlogread	23832
calamon	23833
snmpread	23834
oracleread	23835
mssqlread	23836
jdbcread	23837
tecfmtfilt	23838
V2fmtfilt	23839
msgclsfsrv	23840
calaproxy	23841
tecfmtemit	23842
cmdemit	23843
reportemit	23844
snmpemit	23845
smtpemit	23846
tecifcsrv	23847
jdbcemit	23848

Event caching

If a component looses contact with a downstream component during the transmission of events, these events are then stored in a cache file. In this case, the client process tries to reconnect to the server every 5 seconds.

As soon as a new connection can be established, the cached event can be transmitted. Once the transmission confirmation has been received, the cache entries are deleted.

Cache files are stored in the directory/folder defined by the environment variable CALA_CACHE_DIR. If CALA_CACHE_DIR has not been set, environment variables `TEMP` and `TMP` (with Windows also the `SystemRoot`) are evaluated. If none of these variables has been set, the cache file is stored in the current directory/folder.

> **Note:** The cache files are named `.<client>.<server>.cache`, so they may not be displayed by a normal **ls** call.

Chapter 7. Configuration file `logctlsrv.conf`

The entire configuration of CALA is performed by the file called `logctlsrv.conf` which must be present on every system equipped with any CALA component.

`logctlsrv.conf` contains all communication and start information required for the CALA configuration component implemented on the computer to operate locally, as well as other component-dependent options.

> **Important:** This chapter is only intended to explain the format. Configuration file `logctlsrv.conf` is generated using the graphic configuration program CALAGUI which is part of the CALA Plus Module. CALAGUI only supports configuration files for later changes (re-opening a configuration) which were generated from itself. It may not be possible for the CALAGUI to further process manually changed `logctlsrv.conf` files (i.e. to read them back in).

Format of configuration instructions

All logctlsrv.conf file entries follow the same format

<instruction>=<parameter>{,<parameter>}

Comment: these instructions are constructed hierarchically and reflect the tree structure.

Global configuration instructions applicable to all components

The following chapter contains the global (once per configuration) and supra-component parameters/instructions. These instructions are now described in abridged form, each one being explained with an example.

Configuration instruction serverlist

The instruction is the only instruction which is a mandatory requirement for the configuration file. serverlist describes the list of components which have to be started on the corresponding system by the control command **logctlcmd**.

serverlist=<list of processes being started >

Figure 7-1. Format of the serverlist instruction

If a component has to send data (FIRs) for a downstream component to another computer, the name of the component on the downstream computer has to be part of the list.

```
serverlist=tecifcsrv,tecfmtemit,msgclsfsrv,tecfmtfilt,v2fmtfilt,ascfileread
```

Example 7-1. Example using serverlist

Note: Comment: The sequence reflects the starting sequence for components.

Every component featured in the serverlist instructions list requires a separate configuration entry which controls the parameters and communication routes governing this component and downstream components.

The general structure is as follows:

<componentname>=<sub instruction>!<parameter>{,<sub instruction>!<parameter>}

run instruction

The run instruction describes the commandline call of the relevant component with parameters

run!<program and parameters>

Figure 7-2. Format of run instruction

The parameter -P <port number> is a mandatory requirement for each component, it defines the communication port for the component.

All CALA components also support parameter -d<Debug filename> which, in the event of error diagnosis being required, describes the diagnosis file for all outputs. Parameter -d must directly be followed by the name of the logfile (no blanks are allowed) with or without path indication.

> **Caution**
>
> Diagnosis files from various components can reach a size of several hundred Megabytes within just a few minutes.

```
run!tecfmtemit P 51967 dtecfmtemit.diag
```

Example 7-2. Example using the run statement

The above example defines port `51967` as the communication port for the component `tecfmtemit`. Diagnosis information is directed to filename `tecfmtemit.diag`. In case of several ports being used, these port numbers must be indicated in the form of a list, separated by a semicolon.

For complete description of all parameters refer to the appendix.

target Instruction

The target instruction describes the list of components which follow this component and therefore receive data (FIRs) from the current component.

targets!<target component>{,<target component>}

Figure 7-3. Format of targets instruction

```
targets!tecifcsrv
```

Example 7-3. Example targets usage

This example defines the CALA component `tecifcsrv` as a target component for output data (FIRs)

By default the outgoing port (the local port which is used to establish the connection with a server) is choosen by the system and can therefore by any available port. For security purpose when communication via a firewall, it may be useful to define the outgoing port, which can be done by adding :<portno> to the target component.

```
targets!tecfmtemit:5656,calaproxy:5655
```

Example 7-4. Example targets usage with outgoing port

port instruction

The port instruction is needed for inter-process communication. This port must match the port number of the *run* instruction. In the case of several ports being used, these port numbers must be indicated in the form of a list, separated by a semicolon.

port!<port number>{;<port number>}

Figure 7-4. Format of port instruction

port number: tcp communication port, on which the process listens for incoming data.

```
port!51967
```

Example 7-5. Example port usage

Note: The server components can be configured for a range of ports.

Port list functionality

On both client and server side, a list of ports (separated by semicolons) can be indicated. Components can use these for communication purposes (on the output end) and can be addressed via them in return (input end).

conf instruction

The conf instruction is used for configuration management (**logctlcmd** and **logctlsrv**).

All sub instructions are defined by using the conf instruction are assigned to the relevant process, then made available to the remaining CALA components (configuration management).

conf!<config info>{,<config info>}

Figure 7-5. Format of conf instruction

```
conf!run;port;targets
```

Example 7-6. Example conf usage

The aforementioned conf instruction defines sub instructions run, port and targets for these components. The configuration management function supplies/updates all CALA components with this information.

Note: run and port do not have to be specified explicitly.

ip instruction

The ip instruction is needed whenever the component being described is installed on a different computer (multi-stage CALA concept).

ip!<IP address or hostname>

Figure 7-6. Format of ip instruction

Note: If the ip instruction is set, no run and no conf instruction can exist for these components.

```
msgclsfsrv=ip!foo.bar.com
```

Example 7-7. Example ip usage

In this case, the server `foo.bar.com` is used as a target system. Data (FIRs) for `msgclsfsrv` is sent to this host for further processing.

Serverlist functionality

Instead of one IP address or hostname, a list of targets hosts can be specified. The single hostnames/IP addresses are separated with semicolons (`;`). When connecting to the server, a client will try the hosts in the given order and connect to the first host accepting its request. In other words, if the first server fails to respond, the next server on the list is used.

```
msgclsfsrv=ip!foo.bar.com;thud.grunt.com;194.39.165.97
```

Example 7-8. Example ip instruction using the serverlist functionality

Broadcast functionality

Client components can use the BROADCAST wildcard instead of an IP address/hostname to make use of the implemented broadcast functionality. After a start or after a breakdown in communication with a server, the component searches for a new server via the defined port (port list) in the local network.

Entry BROADCAST must be used within the server list of the remote component, "enable broadcasting" must be ticked at the receiving component.

For broadcasting on a specific subnet, BROADCAST:<subnet-mask> can be used.

Chapter 8. Configuration GUI

There is a graphical user interface for configuration which creates the CALA-configuration file as well as a Tivoli ACP Profile for distribution purposes.

Using the CALA Configuration GUI

The configuration GUI is used for creating CALA configurations. Every component-specific parameter can be set using the graphical system.

The menu item Open (Configuration⟶Open) is used to read an existing configurations, the menu item Create (Configuration⟶Create) enables the creation of a new configuration. The configuration system is able to create pure Tivoli ACP Profiles as well as only the CALA configuration file.

Starting the GUI

CALAGUI can be started with the batch file **calagui.bat** on Windows systems or with the shell script **calagui.sh** on UNIX. Both scripts are located in the `calagui` directory.

CALAGUI supports the following options:

-c <dir>

specifies the directory that contains the configuration files for CALAGUI.

Default: `conf` (This value should not be changed.)

-o <dir>

specifies the standard output directory for configurations. It is recommended that you create a subdirectory for each configuration and its related files.

Default: `data`

-s <dir>

specifies the path to additional scripts that are executed when a configuration is saved.

Default: `scripts`

-x<dir>

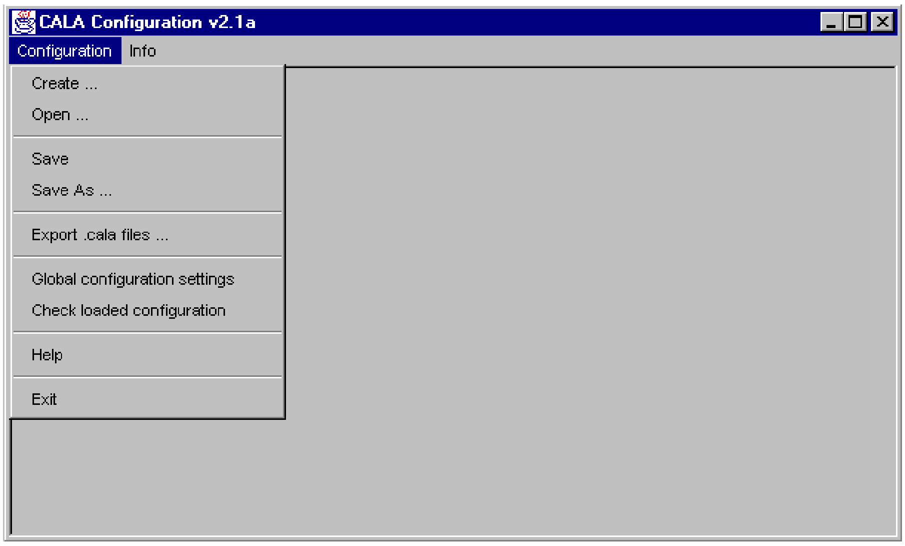
specifies the output path for exporting `.cala` files.

Default: `export`

Note: The `export` directory is used as repository for files related to the CALA Configurator. At the moment, you must synchronize your CALAGUI installation(s) and the TMR Server manually. For details about the subdirectories created in the export directory and the synchronization process between CALAGUI, TMR Server and ManagedNodes see the annex, section *Directory structure in the export directory of CALAGUI*.

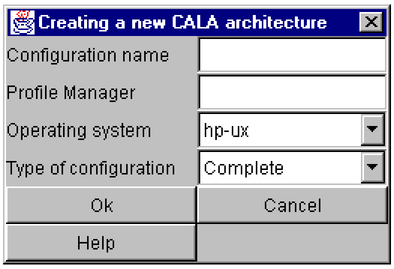
If you want to change the default values, you must edit the corresponding start script. It is currently not possible to specify the options on the command line.

The architecture window opens whenever the configuration tool starts. All available functions (Create, Open, Save, etc.) are arranged under menu item Configuration



The CALAGUI Configuration menu

Setting up a new configuration (Create ...)



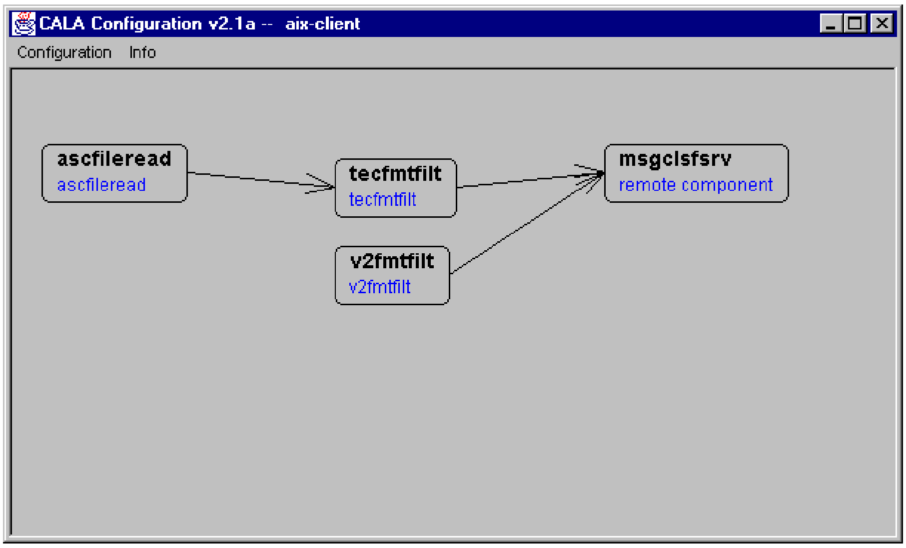
The CALAGUI new configuration dialog

To create a configuration, please select menu item Create ... in the Configuration menu.

Select a configuration name of your choice, a Tivoli ProfileManager name for generating an ACP Profile, the target platform and the type of configuration.

> **Note:** The type of configuration is only used to select the default settings to be used for the architecture. These settings can then be modified later.

Pressing the OK button loads the selected architecture into the architecture window, based on standard definitions.

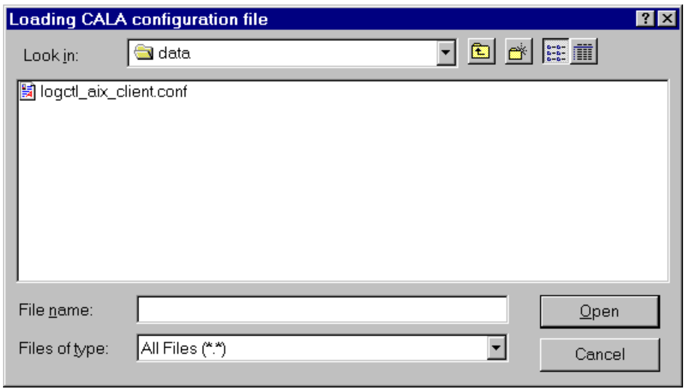


The default HP-UX client configuration.

The illustrated example contains the default client HP-UX architecture.

Opening an existing configuration (Open ...)

To open an existing configuration, please select menu item Open ... in the Configuration menu. Select the desired configuration by selecting the appropriate configuration file.

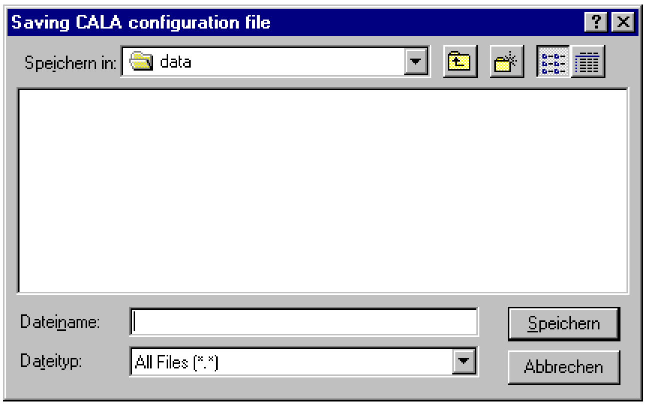


The open configuration dialog

The existing configuration is then loaded into the architecture window.

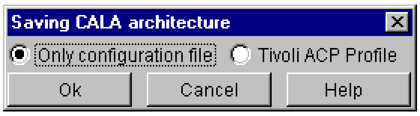
Saving a created or changed configuration (Save, Save as)

Saving with the menu option Save is used to save a new or existing configuration, Save as ... is used to save an existing configuration under a different name. If the configuration being saved has not already been assigned a configuration file (using *Create ...* or Save as ...), the following file browser window is opened.



The save configuration file dialog

Enter the desired name of the configuration file (any name can be selected) and confirm by pressing the Save button. Another menu box then appears on screen.

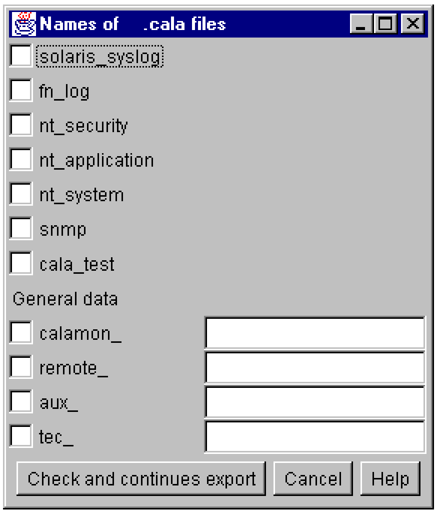


The options dialog when saving a configuration

Select the type of Save process (Only configuration file or ACP Profile) and confirm by pressing the OK button. The configuration created is then saved under the specified name.It is recommended that you create a subdirectory for each configuration and its related files.

Exporting parts of the configuration for use with the CALA configurator

Input files for the CALA configurator can be created by selecting the menu entry Export .cala files. A new dialog window opens, where the user can select the data types to be exported.



The `.cala` files export dialog

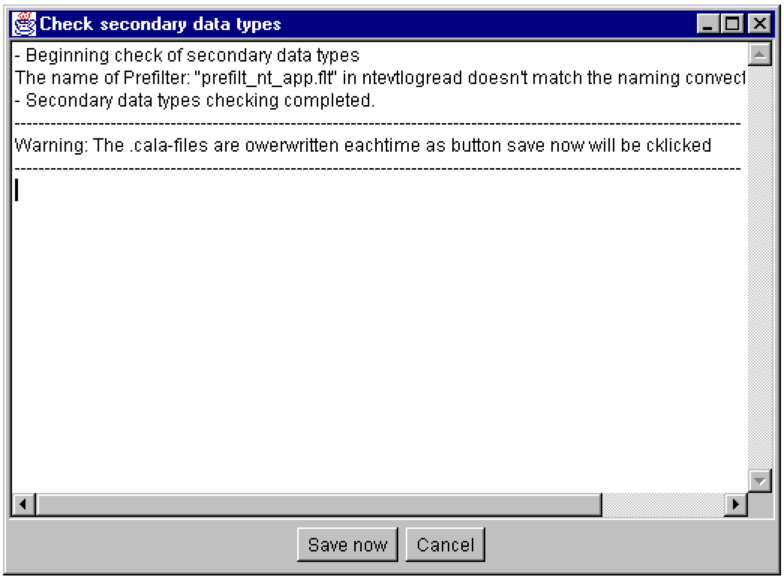
Secondary data type export

Each of the secondary data types of the loaded configuration can be selected for export.

General data export

The general data types need a unique postfix to be given. This enables for example several remote components to be configured.

After selecting the data types to be exported and pressing Check and continues export , the configuration of the selected data types is checked e.g. if the name of the prefilter and message map files are conform to the naming conventions. Warnings from this check shouldn t be ignored non-conform files will be ignored by the CALA configurator.



The Check secondary data types dialog

Selecting Save now from the check window creates the export files. All files are created in subdirectories of the export directory specified with the *x* switch in the start script of CALAGUI. A subdirectory is created for each secondary data type. All referenced files (format files, map files) that can be found in the directory where the configuration is saved and that are conform to the naming convention are copied to the corresponding export directory.

Note: If any referenced file do not exist within the directory where the configuration file is located, not all required files will be present after export. All missing files need to be copied manually afterwards.

For more information about the CALA configurator and the exported file format refer to chapter *CALA Configurator* in the appendix.

Differences between CALAGUI configurations and CALA Configurator

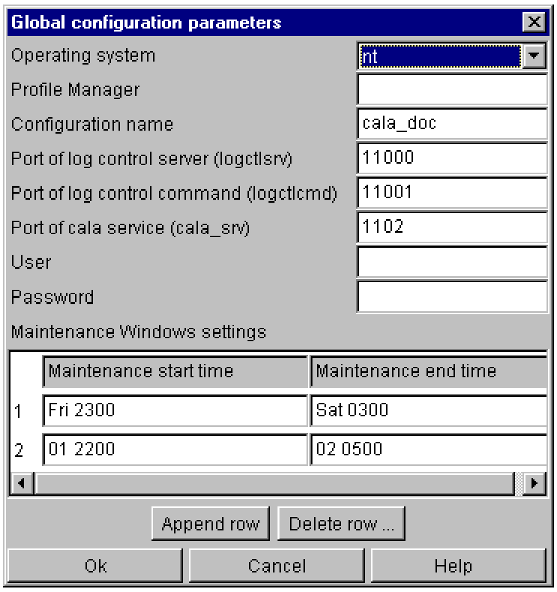
There are some differences between configuration files that are saved from CALAGUI and configurations that are generated by CALA Configurator even if you compare an "original" configuration and a configuration that was generated from the .cala files exported from the "original":

- The port numbers for the components are taken from the template file. The port for the remote component is the only one that is taken from the original configuration.
- The logical names of the components as well as the binary names in the run-statements are taken from the template file.
- If you want to start CALA under a specific user you must specify this user and the corresponding password as parameter for the task `Generate profile for CALACFG`. The user specified in the global configuration settings in CALAGUI will not be passed to the ACP record generated for CALA Configurator.
- If you want to export settings for a client configuration as well as for a server configuration that contain the same secondary data types, you must include both configurations in one configuration file. This is required because the `.cala` files are rewritten each time you export a configuration file that contains definitions for a secondary data type that was exported before.

Altering the global configuration settings

The menu option Global configuration settings can be used for subsequent changes to global settings in a configuration.

Note: Maintenance windows for the CALA can be also specified in this window.



The Global configuration parameters dialog

Any desired number of maintenance windows can be configured. Simply ensure that settings do not overlap. Global settings are set by pressing the OK button.

Global settings in the configuration file

The configuration shown above will result in the following configuration lines:

```
logctlsrv_port=11000
logctlcmd_port=11001
cala_srv_port=10999
maintenance=Fri 2300;Sat 0300;01 2200;02 0500
```

Example 8-1. Global settings in the `logctlsrv.conf` file

Configuration instructions logctlsrv_port and logctlcmd_port

To change the TCP port used by the `logctlsrv` and `logctlcmd` programs, use the instructions logctlsrv_port and logctlcmd_port in the configuration file. If no port is set for one of these programs, the default port (51956 for `logctlsrv` and 51952 for `logctlcmd`) is used.

The ports for log control server (`logctlsrv`) and log control command (`logctlcmd`) are taken from the GUIs entry fields Port of log control server and Port of log control command

> **Note:** If any of these ports are provided as command line argument to **logctlcmd**, the given port(s) is/are used instead of the port(s) from the configuration file.

Note: CALA should can only be configured to use port numbers in the range from 1025 to 65535. Do not change the ports manually to any number outside this range!

```
logctlsrv_port=<port no.>
logctlcmd_port=<port no.>
```

Figure 8-1. Format of logctlsrv_port and logctlcmd_port instruction

```
logctlsrv_port=11000
logctlcmd_port=11001
```

Example 8-2. Example for logctlsrv_port and logctlcmd_port usage

Configuration instruction cala_srv_port (Windows systems only)

If the Windows Service is used, the port for this service can also be set in the configuration file. If not specified `cala_srv` uses the default port 51951.

```
cala_srv_port=<port no.>
logctlcmd_port=<port no.>
```

Figure 8-2. Format of cala_srv_port instruction

```
cala_srv_port=10999
```

Example 8-3. Example for cala_srv_port usage

The configuration instructions logctlsrv_adapters and logctlcmd_adapters

These instruction are used to specify the network adapters used by **logctlcmd** and **logctlsrv**. By default these programs listen on the loopback device only, which means, that only local processes can connect to them.

This behavior prevents the **logctlsrv** from being attacked by remote invaders, but is also denies requests from remote **logctlcmd**s.To open the log control server for communication with remote processes set logctlsrv_adapters to the network adapters from which connections are allowed.

To enable a log control command to communicate to remote processes, the affected network devices have to be given in the logctlcmd_adapters instruction.This instructions can be overwritten by using the **logctlcmd** command line parameter

```
logctlsrv_adapters=<ip-address>{:<ip-address>}
logctlcmd_adapters=<ip-address>{:<ip-address>}
```

Figure 8-3. Format of logctlsrv_adapters and logctlcmd_adapters instructions

```
logctlsrv_adapters=10.0.114.201
logctlcmd_adapters=10.0.114.201:192.168.1.1
```

Example 8-4. Example for logctlsrv_adapters and logctlcmd_adapters usage

Maintenance instruction

The maintenance instruction defines fix maintenance windows which occur periodically. Those maintenance windows are set in the GUIs Maintenance Window settings table.

Within a maintenance window, CALA does not read any events from the event sources. Reading from sources is resumed when the maintenance window is over. All events created within a maintenance window are discarded by the CALA components, even if they are read outside a maintenance window.

```
maintenance=[<dayofweek> | <dayofmonth>] <2digit hours><2digit minutes>{
;[<dayofweek> | <dayofmonth>] <2digit hours><2digit minutes>}
```

Figure 8-4. Format of maintenance instruction

```
maintenance=Fri 2300;Sat 0300;01 2200;02 0500
```

Example 8-5. Example for logctlsrv_adapters and logctlcmd_adapters usage

Fixed maintenance windows are configured in the configuration file by using the maintenance instruction. Each maintenance window consist of a pair of dates, given in the following format:

[<dayofweek> **|** <dayofmonth>] <2digit hours><2digit minutes>

A daily maintenance window only contains the part for hours and minutes (no dayofweek or dayofmonth is given). Hours are given in the 24 hours format.

A weekly maintenance window also contains a three letter abbreviation of the weekday followed by a blank and the hour- and minute-string. A monthly maintenance window is configured giving the day of the month, followed by a blank and the hour- and minute-string.

The example given above defines one weekly maintenance window from 23:00 on Friday to 03:00 on Saturday and one monthly maintenance windows from 22:00 on each 1st of the month to 5:00 on each 2nd .

More global settings

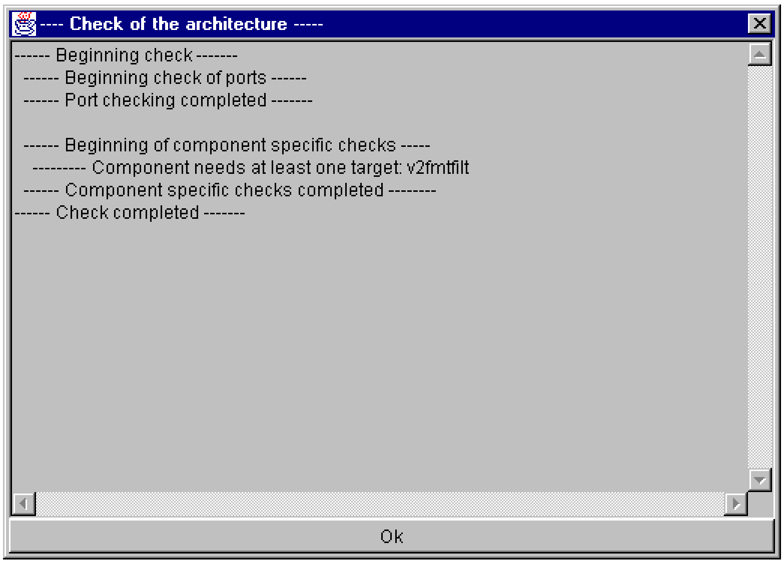
There are some more global settings like operating system, profile manager, configuration name, user and password, which are needed for configuration purposes. For details refer to chapter *Analysis of configuration file*.

```
#operating-system: nt
#profile-manager: ACP for CALA
#name of configuration: NT_Client
#user: cala
#password: 1b14460e00
```

Example 8-6. Example for additional settings in `logctlsrv.conf`

Configuration check

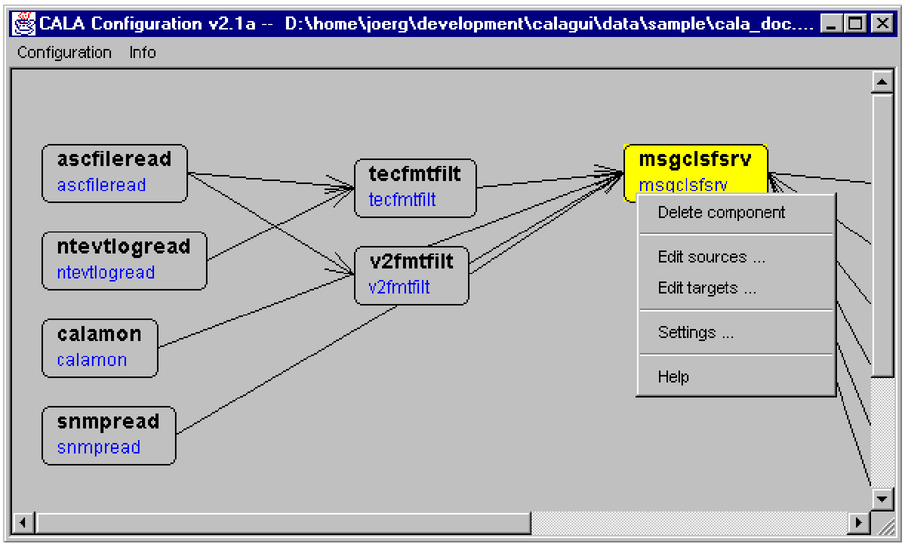
You can use the menu option Check loaded configuration to check your CALA configuration.



Theresult window of Check loaded configuration

component configuration

Each component has a context menu, which opens when pressing the right mouse button while the cursor is placed over the component.

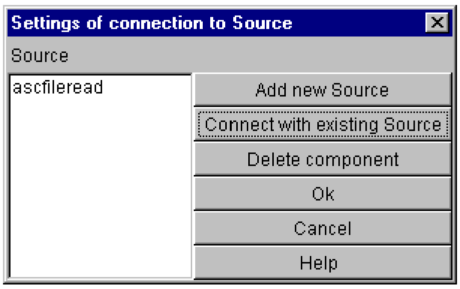


The components context menu

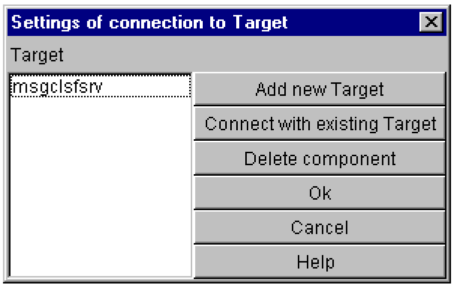
If the menu item Delete component is chosen, the selected component is removed from the configuration.

The menu entry Settings opens the component specific configuration window. Due to individual parameters every component has its own configuration window, see chapter *Component-specific configuration* for details. The components configuration window can also be opened by double clicking the component.

Selecting one of the menu options Edit sources and Edit targets will open another dialog.



The Settings of connection to Source dialog



The Settings of connection to Target dialog

Both dialogs are very similar:

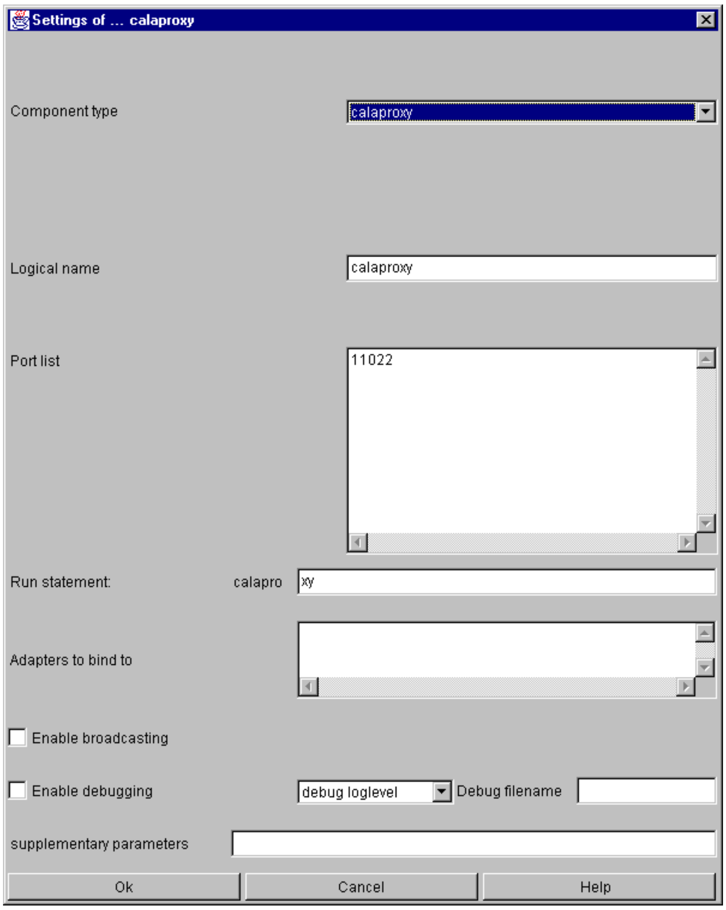
- Add new Source or Add new Target will create a new component (a dialog for selecting a logical name and component type will appear) and connect it to the currently selected component.
- Connect with existing Source Connect or Connect with existing Target shows a list box where the user can choose the component to connect to.
- To remove a component from the source/target list, select the source/target in the left list box and choose Delete component . This will remove the selected source/target component from the list.
- The changes can be applied by pressing Ok or discarded by selecting Cancel

Chapter 9. Component-specific configuration

By double clicking on a component symbol, the components settings dialog window appears. This settings dialog differs between the components, but some fields are common.

Common settings

This is a sample settings window, the window of the component `calaproxy`, which is explained later.



A sample settings window for a CALA component

The first element of each settings dialog is the choice `Component type` where the type of the component is specified (`calaproxy` in this example). The dialogs face depends on the components type and may change if the type is changed.

Logical name

This entry field contains the logical name of the component, which must be unique within the configuration.

The logical name is used to address the components configuration. The logical name will appear in the serverlist and as the identifier for the components configuration line.

```
serverlist= calaproxy
calaproxy=run!calaproxy -P 11022,port!11022,targets!remote_emit, conf!port;run;targets
```

Example 9-1. Example configuration of logical server name

Portlist (port instruction)

The port list gives the ports, the process listens on, at least one port has to be defined for each process. The same port number must not be used for two components on the same machine.

```
port!<port no.>{;<port no.>}
```

Figure 9-1. The port list configuration entry has the following format:

```
port!11022
```

Example 9-2. Example portlist

For local processes, there is a need to given the port number on the argument line.

```
<run statement> -P <port no.>{:<port no> | -<port no>:<port no.>}[-<port no.>]
```

Figure 9-2. Format of argument line containing port assignment

```
run!calaproxy -P 11022
```

Example 9-3. Example run statement containing port assignment

Run statement (run instruction)

The entry field Run statement sets the binary of this component. The first 7 characters are fix, because they specify the components type which is `calapro` (which stand for `calaproxy`) in this case. When using the same component twice on one machine, different binary files must be used.

```
run!<run statement>
```

Figure 9-3. Format of run statement

```
run!calaproxy -P 11022
```

Example 9-4. Example run statement

Debugging

If the Checkbox Enable debug mode is set, the process creates a log file for debugging. If no filename is given in Debug filename , the log is written to `diag_log.txt` There is also a list box for setting the debug level. The debug level can be set to any number between 0 (report everything) and 9 (report only fatal failures). If no debug level is chosen, all messages are written into the debug file.

```
<run statement> -d[[:<loglevel>:]<logfile name>]
```

Figure 9-4. Format of run statement containing debug arguments

```
run!calaproxy -P 11022 d:5:calaproxy.log
```

Example 9-5. Example of run statement containing debug arguments (not from the window above):

Supplementary parameters

Any other program argument can be set here. There are some common arguments, which are described here. Some components have special arguments, please refer to the components description.

parameter	example	description	default value
-M `<no.>`	-M100	max. number of client connections to accept	100
-SF	-SF	stop sever if bind to socket fails	disabled
-NSR	-NSR	don’t allow socket rebind	allowed
-*CT*`<secs>`	-CT30	set connection timeout before caching (a cache file is created if no connection to a server could be created for <secs> seconds)	30
-CM `<size>`	-CM5000000	sets the maximum size of the cache file (in bytes)	5000000

parameter	example	description	default value
-CD <min>	-CD1440	sets the max. age for cached events before they are discarded, 0 disabled discarding	0
-AT <secs>	-AT60	timeout for receiving acknowledges from server (seconds)	120
-AS <secs>	-AS2	server acknowledge sending period (seconds)	2
-CLE	-CLE	create connection lost events	disabled
-CAE	-CAE	create connection accepted events for all accepted connection from remote clients	disabled
-CAT <secs.>	-CAT60	Sets the connection accept timeout this is the time (in seconds) in which the client has to send a first data package after connecting. If <secs.> is a positive value, a accept timeout event is created if any client connected but didn t send any data , if <secs.> is negative, no event is created. *-CAT0* disables this feature	-30
-ZHEARTBEAT_PERIOD= <secs>	-ZHEARTBEAT_PERIOD=120	Tells the component to create heartbeat events periodically. (Does not work for ascfileread, snmpemit and smtpemit.)	disabled
-ZCREATE_STATUS_EVENTS=1	-ZCREATE_STATUS_EVENTS=1	Creates CALA startup and shutdown events. These events are only send to remote servers or to T/EC. (Does not work with ascfileread.) There is a special implementation of startup/shutdown events for snmpemit and smtpemit refer to chapter *Configuring status events of snmpemit and smtpemit* in the appendix for details.	disabled

There are some more parameters for encryption, please refer to chapter *Security* for further information.

The following parameters are set by the CALAGUI and must not appear in the supplementary parameters text field:

parameter	example	description	default value
-P <port no.>	-P 16001	port(s) to open for event reception several ports can be given separated by colons, a port range can be given like this: <1st port>-<last port>.	none
-d[:<loglevel>:]filename	-d:3:msgclsfsrv.log	creates a log file loglevel can be a number between 0 (log everything) and 9 (log only fatal errors), be aware, that log files may grow very fast if loglevel is set low	don’t create a logfile
-SB	-SB	enable broadcast server	disabled
-AB <ip>{:<ip>}	-AB 10.0.114.201	sets on which network device (give by ip address) the component should listen for events, several devices may be separated by colons the loopback device is added automatically	listen on all devices

Display version information

All component binaries can also be called from the command line using the *-v* parameter, which prints a version information. The SNMP components `snmpread` and `snmpemit` also display the version of the used snmp library.

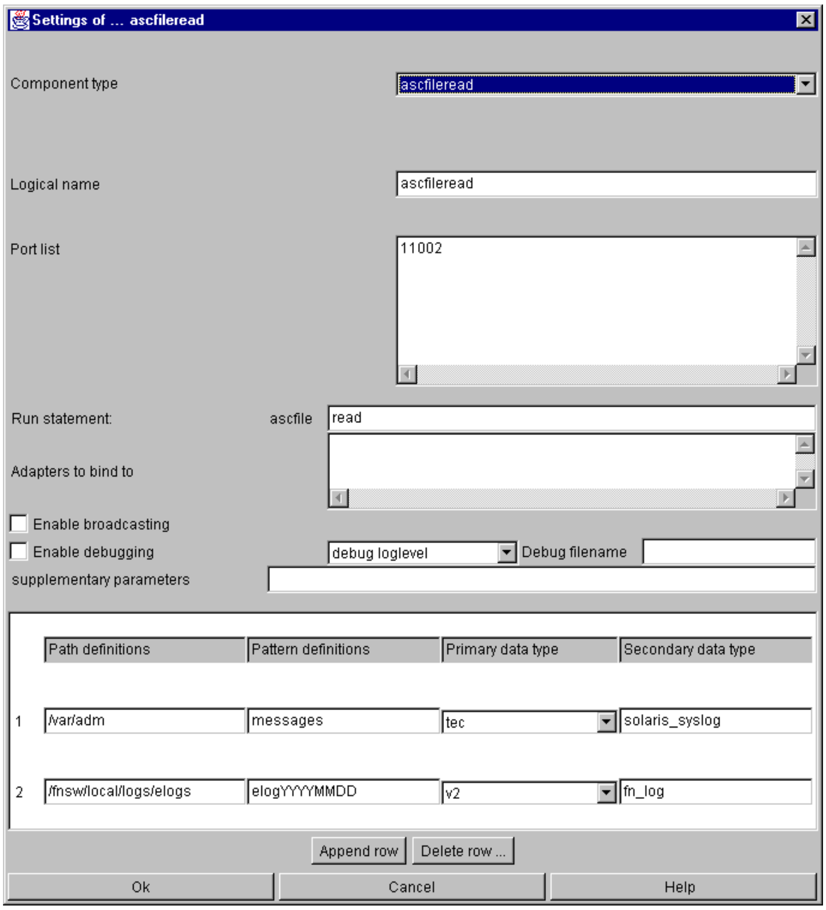
```
[c:\opt\cenit\cala].\snmpread -v
*******************************************************************
**                                                               **
**      snmpread is part of the CENIT Advanced Logfile Adapter   **
**                                                               **
**   version: 2.01-064 - generation date: Jun 26 2003 10:57:18   **
**                                                               **
**              (c)1999-2002 CENIT AG Systemhaus                 **
**                                                               **
*******************************************************************
**                    NET-SNMP version 4.2.3                     **
*******************************************************************
[c:\opt\cenit\cala]
```

Figure 9-5. Displaying version information of `snmpread`

ascfileread

The ASCII file reader component reads the data from one or more configured files and sends it to a filter for further processing. The ascfileread window is used for defining all files, directories, folders and the related data formats for reading out files and pipes.

Wildcards (*, ?) can be used for directories/folders as well as for filenames. There are also additional wildcards to define logfiles and to specify hours, days, months and years within file names. E.g. if a logfile is to be monitored with a 2-digit month and a 4-digit year number at the end of its name (`logfile_06-2000`), the wildcard used for defining its name would look like this: logfile_MM-YYYY



The Settings of ascfileread dialog

ascfileread specific parameters and their setting in the configuration file

The following configuration line was created from the window above:

```
001  ascfileread=run!ascfileread -P 11002,port!11002,targets!tecfmtfilt;v2fmt ⁄
...  filt,pathlist!1;/var/adm;2;/fnsw/local/logs/elogs,ptrnlist!1;messages;2 ⁄
...  ;elogYYYYMMDD,assoc!1;1;tec;solaris_syslog;2;2;v2;fn_log,conf!port;run; ⁄
```

```
...  targets;pathlist;ptrnlist;assoc
```

Figure 9-6. An example configuration line for ascfileread

pathlist instruction

pathlist defines the list of paths (directories/folders) in which logfiles are searched. Its parameters are taken from the GUI configuration from the column Path definitions.

```
pathlist!<Number>;<Path>{;<Number>;<Path>}
```

Figure 9-7. Format of pathlist instruction

```
pathlist!1;/var/adm;2;/fnsw/local/logs/elogs
```

Example 9-6. Example pathlist instruction

The above example defines two directories/folders in which the logfiles being processed may exist. A unique number must be defined for each path. This number is referenced using the assoc instruction described below.

The separate configuration of paths and filename simplifies configuration if the same pattern has to be used for several paths. (E.g. if you are looking for the pattern `*.log` in three different paths, the pattern has to be configured only once.)

Wildcards and variables requiring interpretation that can be used in pathnames.

Supported wildcards:

`*`

designates any sequence of characters

?

designates any character

Variables requiring interpretation:

`HH`

two-digit hour display

`MM`

number of month, two-digit

`DD`

day, two-digit

`YY`

number of year, two-digit

`YYYY`

number of year, four-digit

ptrnlist instruction (pattern list)

The ptrnlist instruction defines the list of file patterns used for processing purposes. The ptrnlist parameters are taken from the GUI configuration from the column Pattern definitions.

```
001  ptrnlist!<Number>;<Pattern match>[:<encoding>]{;<Number>;<Pattern match>
...  [:<encoding>]}
```

Figure 9-8. Format of ptrnlist instruction

```
001  ptrnlist!1;messages:UTF-8;2;elogYYYYMMDD
```

Example 9-7. Example ptrnlist instruction

Wildcards and variables requiring interpretation are used to describe filenames.

Supported wildcards:

*

designates any sequence of characters

?

designates any character

Variables requiring interpretation:

`HH`

two-digit hour display

`MM`

number of month, two-digit

`DD`

day, two-digit

`YY`

number of year, two-digit

`YYYY`

number of year, four-digit

For a list of supported encoding refer to *Supported character sets*.

For every pattern entry a number is assigned which reflects the assoc instruction described below for reflecting the path/filename combination to be processed.

- The above example defines two file patterns which are interpreted at run time.
- The first pattern addresses a file named `messages` which is expected to be UTF-8 encoded.
- Sample 2 is used to define precise daily logfiles, starting with `elog`. On 20.12.2000 this configuration would, for example, process filename `elog20001220`. The file is exptected to use the default system encoding.
- `sna*.err` would address all filenames beginning with prefix `sna` and extension `.err`.
- `messages.?` would identify all messages files having a one character extension.

By default the `ascfileread` checks every 5 minutes for new matching paths and files.

assoc instruction

The assoc instruction associates paths from the pathlist with file patterns from the ptrnlist instruction. Each row from the Settings of ascfileread window's table generates one assoc entry in the configuration file.

```
assoc!<path1istX>;<ptrn1istX>;<primary type>;<secondary type>{;<path1istX>;<ptrn1istX>;<primarytype>;<secondary type>}
```

Figure 9-9. Format of assoc instruction

```
assoc!1;1;tec;solaris_syslog;2;2;v2;fn_log
```

Example 9-8. Example assoc instruction

PathlistX represents a previously defined path number, ptrnlistX represents a previously defined filename number (pattern).

The parameter `<primary type>` can be selected from values and represents the relation to logfiles which can be described as `.fmt`, represents the logical link to complex data formats.

The parameter `<secondary type>` can be selected from one of the format names defined under formatlist. (see configuration of filter components).

This prompts a search for the following directory/folder combination:

- `/var/adm/messages` (type tec/solaris_syslog)
- `/fnsw/local/logs/elogs/elogYYYYMMD` (type v2/fn_log)

Example 9-9. Files that would match with above pathlist, pattrnlist and assoc configuration

Note: To read pipes on Microsoft Windows `\\pipe\` must be given as path, the pipes name has to be given as filename.

ascfileread command line parameters

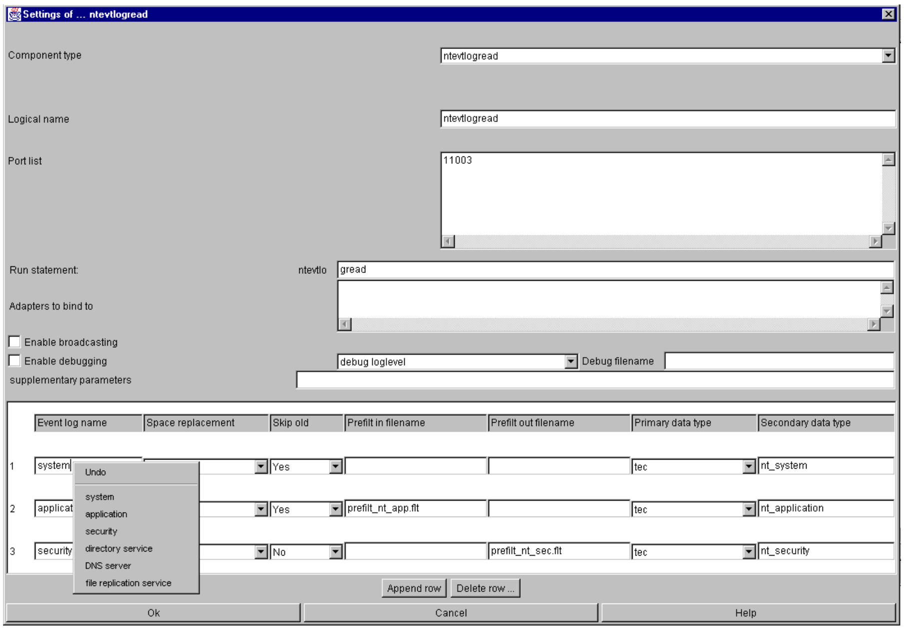
These parameters can be set in the field supplementary parameters

parameter	example	description	default value
-E	-E	When opening a logfile the first time: skip all old events, send only new events	disabled
-e	-e	Skip all old events each time a logfile is opened, send only new events	disabled
-U <seconds>	*-U 60*	Sets the period when `ascfileread` looks for new files in seconds.	300
-H <hostname>	-H foo.bar.com	Sets the name of the host, ascfileread runs on. The hostname is requested by gethostname() function if this parameter is not given.	use gethost-name()
-B<count>	-B5	Specifies the max. number of 16K blocks to be send to the filters each second. This parameter should be used with care, because it may result in some unknown error events.	no limit
-O	-OUTF-8	Specifies the default character set to be used for reading files if no encoding is specified.	the system default

ntevtlogread

The `ntevtlogread` (NT Event log Reader) is used for reading the Microsoft Windows Event log.

The Settings of ntevtlogread window is used to define all read functions and parameters. For every event log (system, security, application or any user defined eventlog), a dedicated secondary data type can be assigned. This makes it possible to use a separate format file for every type of eventlog.



The Settings of ntevtlogread dialog

ntevtlogread specific parameters and their setting in the configuration file

The following configuration line was created from the window above

```
ntevtlogread=run!ntevtlogread -P 11003,port!11003,targets!tecfmtfilt,evtlog!1;system;2;application;3;security,spacereplacement!1;1;2;1;3;1,assoc!1;tec;nt_system;2;tec;nt_application;3;tec;nt_security,skip_old!1;1;2;1;3;0,prefilt_in!2;prefilt_nt_app.flt,prefilt_out!3;prefilt_nt_sec.flt,conf!port;run;targets;evtlog;spacereplacement;assoc;skip_old;prefilt_in;prefilt_out
```

Example 9-10. An example configuration line for `ntevtlogread`

evtlog instruction

> evtlog defines which eventlogs the reader should read. The eventlogs are given as a pair <numeric id>;<logfile_id>.

The popup menu of the text field Eventlog name shows a selection of standard eventlog ids for Windows NT and Windows 2000 systems. Selecting a id from the popup menu pastes this id into the text field.

```
evtlog!<numeric id>;<logfile id>{;<numeric id>;<logfile id>}
```

Figure 9-10. Format of evtlog instruction

```
evtlog!1;system;2;application;3;security
```

Example 9-11. Example evtlog instruction

This defines, that the Microsoft Windows `system`, `application` and `security` eventlogs must be read and each of them is given a numeric id (system=1, application=2, security=3).

spacereplacement instruction

Defines if blanks should be replaced by underscores for fields source and sid. The instruction consist of a pair <numeric id>;<flag> for each logfile. If <flag> is set to 1 this means "spacereplacement on", 0 means "spacereplacement off".

```
spacereplacement!<numeric id>;0|1{;<numeric id>;0|1}
```

Figure 9-11. Format of spacereplacement instruction

```
spacereplacement!1;1;2;1;3;1
```

Example 9-12. Example spacereplacement instruction

This example switches space replacement on for the three defined event logs (`system`, `application` and `security`).

skip_old instruction

If this parameter is set for a logfile, all entries which have a timestamp before 0:00 clock of the current day, are discarded. The instruction consist of a pair <numeric id>;<flag> for each logfile. If <flag> is set to 1 this means "skip old entries", 0 means "process old entries".

```
skip_old!<numeric id>;[0|1]{;<numeric id>;[0|1]}
```

Figure 9-12. Format of skip_old instruction

```
skip_old!1;1;2;1;3;0
```

Example 9-13. Example skip_old instruction

The example switches skip_old on for the system log (1) and the application log (2). skip_old is switched of for the security log (3).

prefilt_in and prefilt_out instructions

These instructions set pre-filters for each logfile. The association consist of a pair <numeric id>;<prefilt_file>.

Pre-filters are used to discard events before sending them to any other process. The in-filter specifies events that should not be discarded, the out-filter specifies events that should be discarded. Pre-filters are optionally. If no filter is set, all events are send to the target processes.

```
prefilt_in!<numeric id>;<filter_file>{;<numeric id>;<filter_file>}
prefilt_out!<numeric id>;<filter_file>{;<numeric id>;<filter_file>}
```

Figure 9-13. Format of prefilt_in and prefile_out instructions

```
prefilt_in!2;prefilt_nt_app.flt
prefilt_out!3;prefilt_nt_sec.flt
```

Example 9-14. Example prefilt_in and prefilt_out instructions

This defines an in-filter for the application log (2) and an out-filter for the security log (3).

The pre-filter files are text files, structured like this:

- each line contains a list of assignments <key>=<value>
- assignments are separated by semicolons: <key1>=<value1>;<key2>=<value2>
- several possible values for one key can be separated by a comma (<key>=<value1>,<value2>)
- a filter matches if any line matches
- only events that match any prefilt_in and do not match any prefilt_out are sent to the filter process
- if no pre-filter is set, all events are sent to the filter process

Possible pre-filter keys are: eventid , eventtype and source

```
source=SNMP,Print;
```

Example 9-15. Example for a pre-filter file to match all events from the SNMP and Print source:

assoc instruction

The assoc instruction associates an event log with a type and logical name.

```
assoc!<numeric id>;<primary type>;<secondary type>{;<numeric id>;<primary type>;<secondary type>}
```

Figure 9-14. Format of assoc instruction

```
assoc!1;tec;nt_system;2;tec;nt_application;3;tec;nt_security
```

Example 9-16. Example assoc instruction

The `<primary type>` parameter can be selected from values `tec` and `v2s`. `tec` represents the relation to logfiles which can be described as `.fmt`, `v2s` represents the logical link to complex data formats.

The parameter `<secondary type>` can be selected from one of the format names defined under formatlist (see parameters of `v2fmtfilt` and `tecfmtfilt`).

```
assoc!1;tec;nt_system;2;v2;nt_application;3;tec;nt_security
```

Example 9-17. Another example for assoc instruction using different primary types

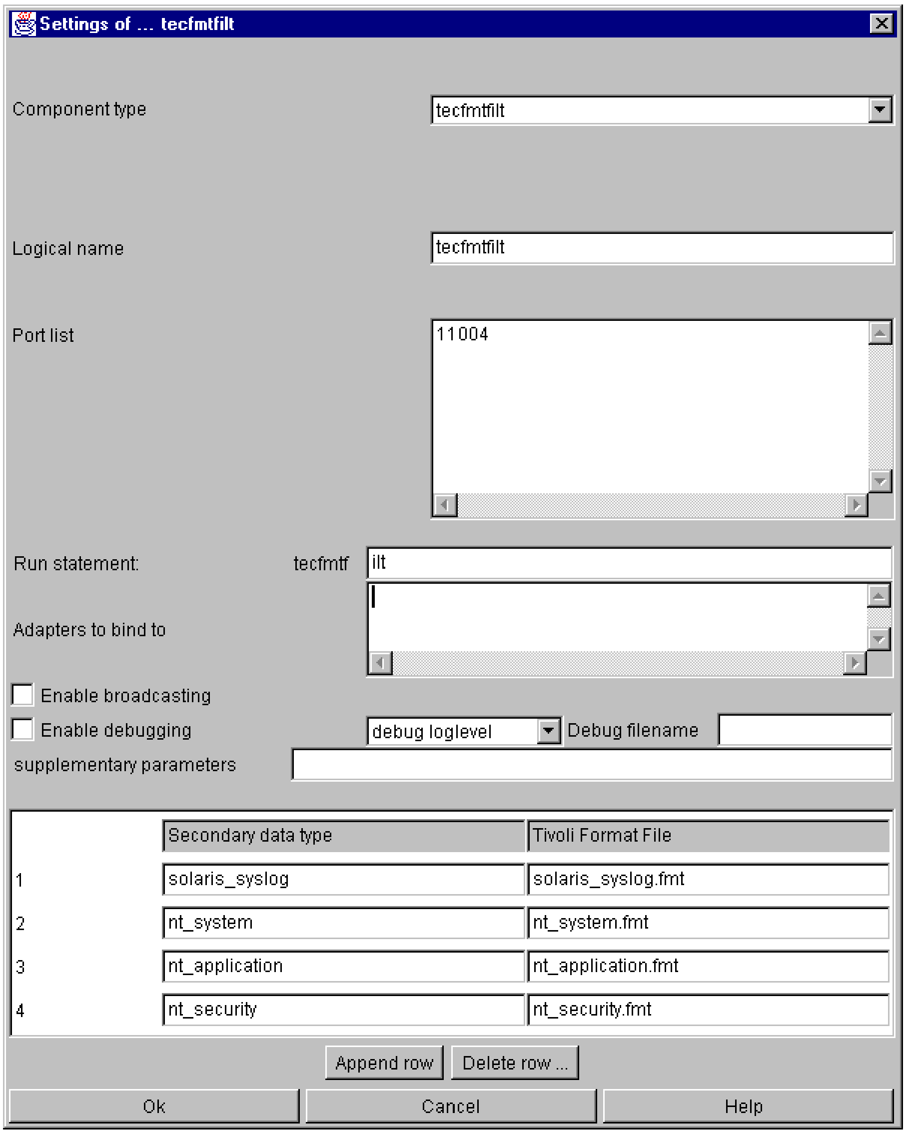
ntevtlogread command line parameters

These parameters can be set in the field supplementary parameters

parameter	**example**	**description**	**default value**
-E	-E	When opening an eventlogthe first time: skip all old events, send only new events	disabled
-e	-e	Skip all old events each time a eventlog is opened, send only new events	disabled
-H `<hostname>`	-H foo.bar.com	Sets the name of the host, `ntevtlogread` runs on. The hostname is requested by gethostname() function if this parameter is not given.	use gethost-name()
-O	-OUTF-16	Specifies the default character set to be used for reading files if no encoding is specified.	UCS2-LE

tecfmtfilt

The T/EC format filter window (`tecfmtfilt`) is used for defining all secondary data formats, which are not multi-line (these can be described using a standard Tivoli `.fmt` file).



The Settings of tecfmtfilt dialog

Filter processing of data by component `tecfmtfilt` is based on Tivoli `.fmt` files. These are read in directly from format definitions (without prior compilation).

For every format file, a logical secondary data type must be defined with the formatlist instruction. The secondary data type identifier should have any coherence with the format filename.

The primary data type of the `tecfmtfilt` filter is always `tec`.

tecfmtfilt specific parameters and their setting in the configuration file

This is the configuration line created from the settings windows above:

```
tecfmtfilt=run!tecfmtfilt -P 11004,port!11004,targets!msgclsfsrv,formatlist!solaris_syslog;solaris_syslog.fmt;nt_system;nt_system.fmt;nt_application;nt_application.fmt;nt_security;nt_security.fmt,conf!port;run;targets;formatlist
```

Example 9-18. An example configuration line for `tecfmtfilt`

formatlist instruction

The formatlist instruction defines an association between secondary data types and Tivoli `.fmt` files which describe how to create events from the data stream. The association is taken from the GUI's table.

```
formatlist!<secondary type>;<name of fmt file>{;<secondary type>;<name of fmt file>}
```

Figure 9-15. Format of formatlist instruction

```
formatlist!solaris_syslog;solaris_syslog.fmt;nt_system;nt_system.fmt;nt_application;nt_application.fmt;nt_security;nt_security.fmt
```

Example 9-19. Example formatlist instruction

Any desired name can be used as a logical name (= secondary data type), e.g. aix4r1 for the format file `tecad_logfile_aix4-r1.fmt`

The example defines the `tecfmtfilt` to process the four data types `solaris_syslog`, `nt_system`, `nt_application` and `nt_security` which are defined in the following format files:

secondary data type	**format file name**
`solaris_syslog`	`solaris_syslog.fmt`
`nt_system`	`nt_system.fmt`
`nt_application`	`nt_application.fmt`
`nt_security`	`nt_security.fmt`

Note: Events which are assigned a classname starting with *DISCARD are discarded by this component. (This is an enhancement of the Tivoli adapter, which discards only events from the one class *DISCARD*).

Note: Since CALA 2.03 tec format files need to be saved in UTF-8 encoding if they contain Non-ASCII-127 characters.

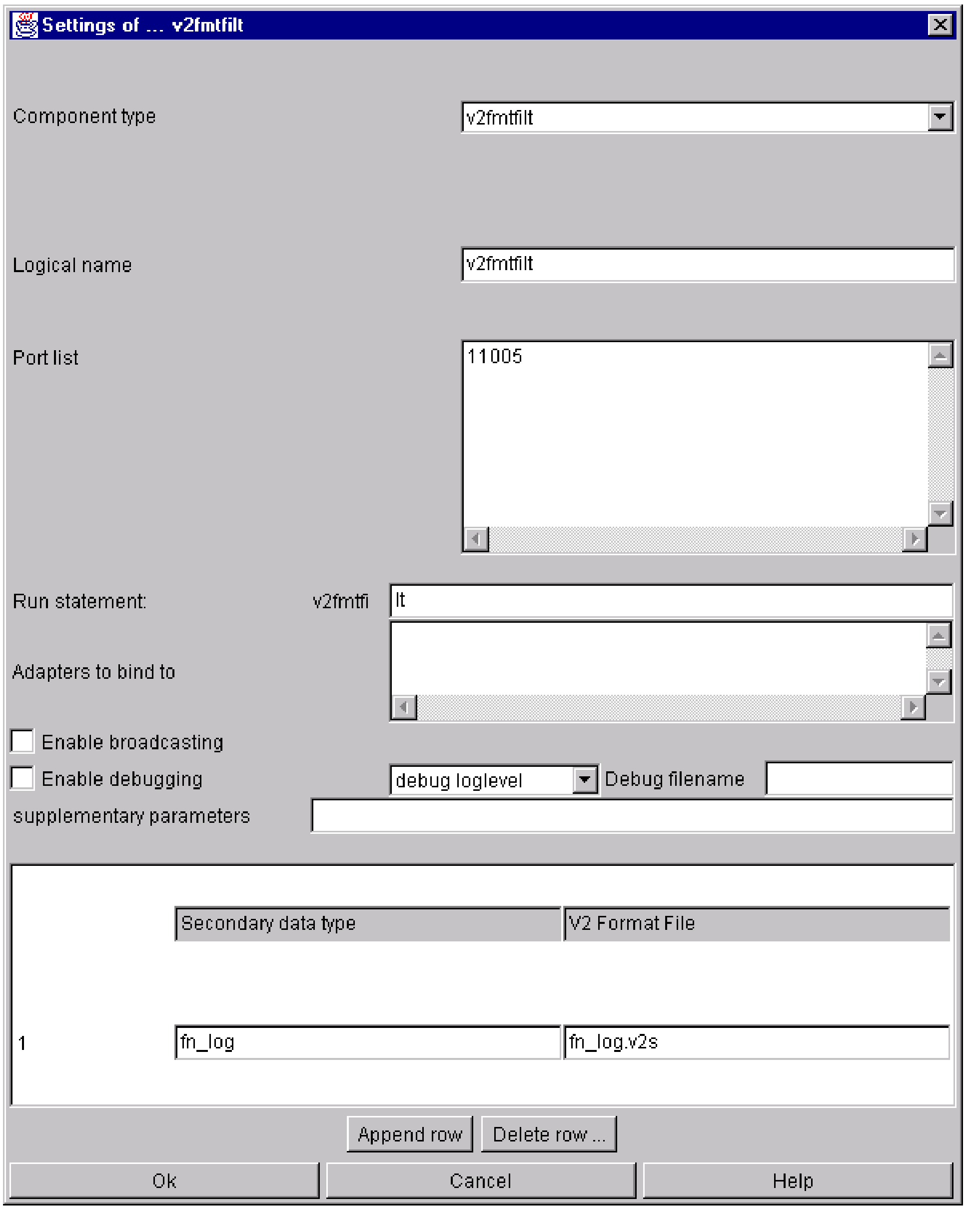
tecfmtfilt command line parameters

These parameters can be set in the field supplementary parameters

Parameter	**example**	**description**	**default value**
`-Q <size in bytes>`	-Q 2000000	Sets the size of the static buffer used for parsing. Increase this value if you get the message `not enough quickmem` in the debug file.	1048576

v2fmtfilt

The v2 format filter window (`v2fmtfilt`) is used for describing all complex data flows, which cannot be described with a conventional Tivoli `.fmt` file. This includes multi-line logfile formats or those formats which can only be described using complex expressions.



The Settings of v2fmtfilt dialog

In contrast to the standard Tivoli format descriptions, the CALA v2 format makes it possible to implement format descriptions of almost any level of complexity.

v2fmtfilt specific parameters and their setting in the

configuration file

The configuration of `v2fmtfilt` is identical to the configuration of *`tecfmtfilt`* the difference is the format of the input files (which is Tivoli `.fmt` for `tecfmtfilt` and `.v2s` for `v2fmtfilt`).

```
v2fmtfilt=run!v2fmtfilt -P 11005,port!11005,targets!msgclsfsrv,formatlis
t!fn_log;fn_log.v2s,conf!port;run;targets;formatlist
```

Example 9-20. An example configuration line for `v2fmtfilt`

formatlist instruction

The filter-specific parameter formatlist has already been described with filter *`tecfmtfilt`* .The difference with the formatlist definition for `v2fmtfilt` is the syntax the format files use. Format files used with the `v2fmtfilt` have to be in v2 format, while format files used with the *`tecfmtfilt`* have to be in Tivoli file format.

```
formatlist!<secondary type>;<name of v2 file>{;<secondary type>;<name of
 v2 file>}
```

Figure 9-16. Format of formatlist instruction

```
formatlist!fn_log;fn_log.v2s
```

Example 9-21. Example formatlist instruction

The primary data type for the `v2fmtfilt` filter is always `v2` .

Note: The syntax of CALA v2 format files is explained in the appendix.

Note: Events which are assigned a classname starting with *DISCARD are discarded by this component.

Note: Since CALA 2.03 v2 format files need to be saved in UTF-8 encoding if they contain Non-ASCII-127 characters.

v2fmtfilt command line parameters

These parameter can be set in the field supplementary parameters

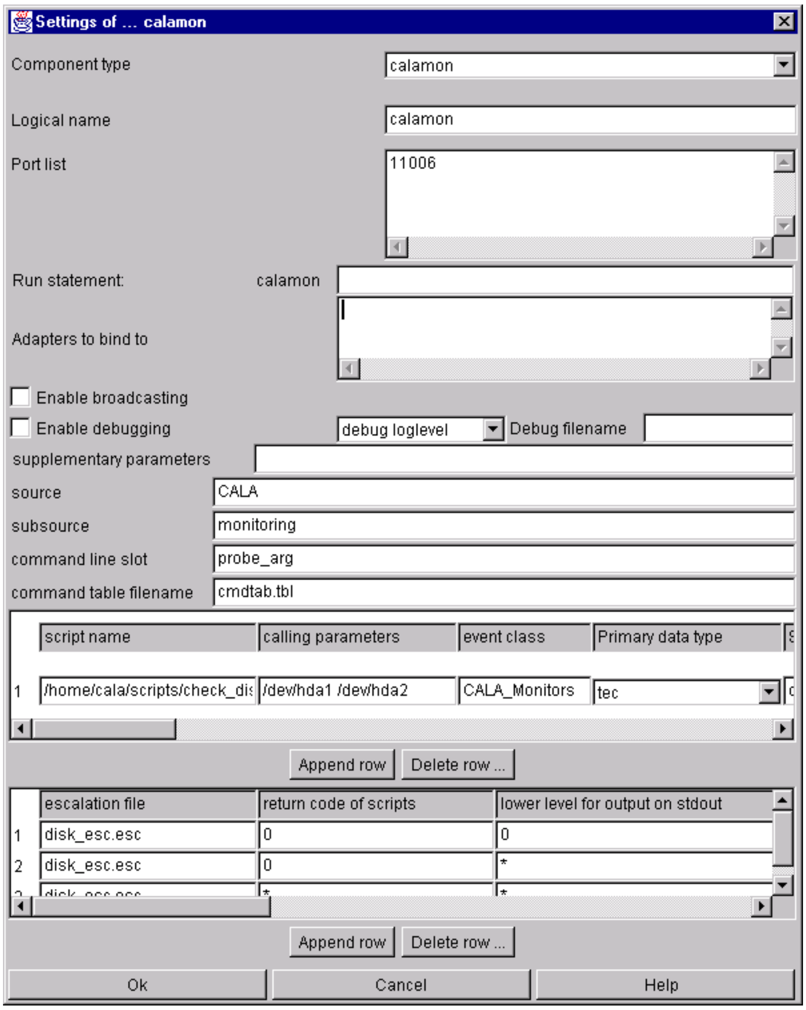
Parameter	example	description	default value
`-Q <size in bytes>`	-Q 2000000	Sets the size of the static buffer used for parsing. Increase this value if you get the message `not enough quickmem` in the debug file.	1048576
`-D`	-D	If this parameter is given, each event contains the following fields (which are taken from the events timestamp or from current time if no timestamp is given): • `YEAR`: 4 digits • `MONTH`:2 digits • `DAY`:2 digits • `HOUR`:2 digits • `MINUTE`:2 digits • `SECOND`:2 digits	disabled

calamon

`calamon` is CALA's monitoring engine implementation.

> **Note:** There is a configuration GUI for command tables called Monitoring Manager , refer to the Monitoring Manager User's Guide for more information of this product.

The Settings of calamon window defines the name and the settings of the command table file which contains the parameters for the processes to be started.



The Settings of calamon dialog

There are two tables with parameters, which must all be set to a value (empty parameters are not allowed).

The first table configures the scripts or programs to be started and their parameters. It contains the following columns:

script name

path and name of the script to be started

command line parameters

parameters which are passed to the script

primary data type , secondary data type and event class

type of event to be created

stdout field

FIR field to receive the script s output to stdout

stderr field

FIR field to receive the script s output to stderr

return code field

FIR field to receive return value of the script

comment prefix

prefix which marks a line of the scripts output as comment (e.g. #)

comment field

FIR field to receive comment lines (which are removed from stdout field)

escalation field

FIR field to receive escalation level (is set from escalation file)

escalation file

name of escalation file (see escalation table description below)

the execution times specification

The execution times specification is similar to the unix crontab, it uses the following columns

execution months

month may be given numeric (from 1 to 12) or as three letter appreviations (e.g. 11,Jan-Mar to allow execution in November, January, February and March)

execution days of month

e.g. 1,15 to allow scripts execution each 1st and fifteenth

execution days of week

the days of week are given as three letter appreviations or numeric values (0 to 7, 0 and 7 mean sunday).

execution hours

24 lesson format, e.g. 23-1 to allow execution from 11 p.m. to 1 a.m.

execution minutes

minutes from 0 to 60

execution seconds

seconds from 0 to 60

execution period

length of period in seconds or in format DD:HH:MM[:SS] (days:hours:minutes[:seconds])

message template

a template for the message to be written into the message slot (may contain links to other fields)

message slot

slot the name of the message slot

The script is run periodically within the specified execution times.

The message template can contain links to other fields (e.g. the stdout and stderr fields). Links to other fields are indicated by writing the field name enclosed with < and >.

```
001  The disk <stdout> is <rc>% full. You may run in difficulties.
```

Example 9-22. An example message template

The escalation settings are given in the lower table which contains the followings parameters:

escalation file

name of escalation file (the same escalation file may occur multiple times here, but should only occur in lines which are following each other)

return code of script*

the script s return code

lower level output on stdout*

lower limit of escalation level or alpha-numeric value of stdout

upper level output on stdout

upper limit of escalation level (used only for numeric values of lower/upper level output on stdout)

lower level output on stderr* and upper level output on stderr

see above, but output on stderr is used

value of escalation field

value the escalation field is set to

action

either DISCARD (no event is created), SEND (create an event and send it to the targets) or SENDFIRST (create an event only if the escalation value changes)

Fields marked with a * may also contain wildcards with special meanings:

*

matches any or no output

+

matches any output

!

matches no output

When using non-numeric monitors, only the lower values are checked. The escalation files are checked top down, the first matching line is used.

calamon specific parameters and their setting in the configuration file

Because the configuration of calamon may be very complex, it is moved to calamon specific files. Therefore its configuration line in logctlsrv.conf is very simple:

```
calamon=run!calamon -P 11006,port!11006,targets!msgclsfsrv,cmdtab!cmdtab
.tbl,source!CALA,sub_source!cala_mon,cmdline_slot!cmdline,conf!port;run
;targets;cmdtab
```

Example 9-23. An example configuration line for calamon

cmdtab instruction

The cmdtab instruction sets the name of the calamon command table file. The filename is given in the GUI s entry field command table filename

```
cmdtab!<name of command table file>
```

Figure 9-17. Format of cmdtab instruction

```
cmdtab!cmdtab.tbl
```

Example 9-24. Example cmdtab instruction

Note: The format of the command table file is described in the appendix.

`source` instruction

The source instructions set s the value of the source slots of the created FIRs. If source is not configured, the source slot is set to `FSM CALA` .

sub_source instruction

Like the source instruction sets the value of the source slot, the sub_source instruction sets the value of the standard T/EC field sub_source . If not configured, sub_source is set to the logical name of the calamon process.

cmdline instruction

This instruction defines the name of the field to receive the command line (program + arguments) called for the monitor. If unset, this information is written into the slot `cmdline` .

calamon command line parameters

This parameter can be set in the field supplementary parameters

parameter	**example**	**description**	**default value**
-H `<hostname>`	-H foo.bar.com	Changes the hostname written into the events `HOSTNAME` field.	tcp/ip hostname
-T `<seconds>`	-T 120	Sets the timeout for monitor scripts/programs.	60
-O	-OUTF-8	Specifies the default character set to be used for passing parameters to the monitor and to parse the input streams.	the system default encoding

Structure of FIRs created by calamon

The following table shows the fields of a calamon created FIR:

slot name	**value**
`source`	from source instruction
`sub_source`	from sub_source instruction
cmdline (configurable)	command line (program + arguments)
`date`	monitors execution time
`hostname`	name of the host running calamon
`origin`	ip address of the host running calamon
rc	monitors return code

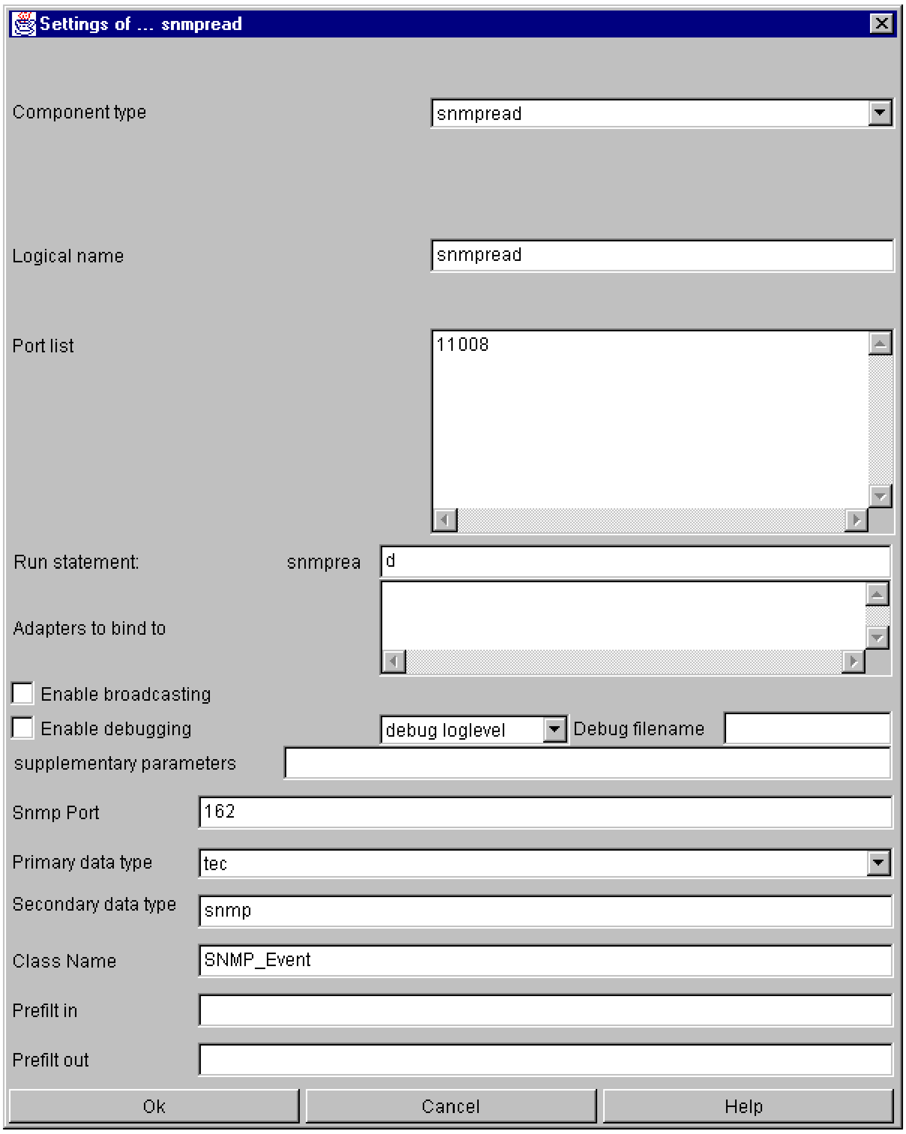
slot name	value
stdout	monitors output to stdout
stderr	monitors output to stderr
comment	monitors output to stdout lines beginning with comment prefix
msg	message generated from message template
severity	events severity according to escalation table

The names of the fields `rc`, `stdout`, `stderr`, `comment`, `severity` and `msg` are defined in the command table. If `stdout` and `stderr` are written into the same slot, `stderr` is redirected to `stdout`.

Events can also be suppress depending on their return code or output to stdout or stderr (see documentation of escalation table above).

snmpread

`snmpread` is the CALA component to receive SNMP traps and forward them as CALA events (FIRs).



The Settings of snmpread dialog

snmpread specific parameters and their setting in the configuration file

This is the configuration line created from the settings window

```
snmpread=run!snmpread -P 11008,port!11008,
targets!msgclsfsrv,type!tec;snmp,
```

```
class!SNMP_Event,conf!port;run;targets;type;class
```

Example 9-25. An example configuration line for `snmpread`

type instruction

The type instruction specifies the primary and secondary data type assigned to received events. Its values are taken from the primary data type and secondary data type entry fields. (Either both or none has to be given.)

```
type!<primary type>;<secondary type>
```

Figure 9-18. Format of `type` instruction

```
type!tec;snmp
```

Example 9-26. Example `type` instruction

This defines received SNMP events to be of primary type `tec` and of secondary type `snmp`.

The type instruction is optional. If it is not given, the default values for primary (`cala`) and secondary type (`snmpreader`) are used.

class instruction

This instruction defines the class received SNMP events will get. This value is taken from the entry field Name of class . This instruction is optional, if it is not given (no value in entry field), the default class `CALA_SNMP` is used.

```
class!<class name>
```

Figure 9-19. Format of class instruction

```
class!SNMP_Event
```

Example 9-27. Example class instruction

prefilt_in and prefilt_out instructions

These instructions sets pre-filters for the SNMP reader.

Pre-filters are used to discard events before sending them to any other process. The in-filter specifies events that should not be discarded, the out-filter specifies events that should be discarded.

Pre-filters are optionally. If no filter is set, all events are send to the target processes.

```
prefilt_in!<filter_file>
prefilt_out! <filter_file>
```

Figure 9-20. Format of prefilt_in and prefilt_out instructions

```
prefilt_in!snmp_in.flt
prefilt_out!snmp_out.flt
```

Example 9-28. Example prefilt_in and prefilt_out instructions

The pre-filter keys are the received SNMP fields (description see below). For further information about pre-filters, refer to the pre-filter section in the *ntevtlogread* description.

Pre-filters are optional. Every event is sent to the targets if no pre-filter is set.

Snmpread command line parameters

These parameter can be set in the field supplementary parameters

parameter	example	description	default value
`-p <port no.>`	-p 20015	Sets the tcp port on which to listen for SNMP events. (This parameter can be set with the CALA-GUI entry field `Snmp port`)	162 (the default SNMP port)

Snmpread generated Events

Events (FIRs) generated by the CALA component `snmpread` process contain the following fields:

field name	description
`ORIGIP`	IP address of trap sender (IP address or resolved hostname)
`SENDOID`	OID of sending system
`COMMUNITY`	community string
`TRAPTYPE`	trap type
`SPECTYPE`	spectype
`OID<no.>`	OID of Variable `<no.>`
`TYPE<no.>`	Type of Variable `<no.>` Possible values are: `ASN_OCTET_STRING`, `ASN_INTEGER` and `ASN_BIT_STRING`
`VALUE<no.>`	Value of Variable `<no.>`

`<no.>` is the number of the SNMP event variable. These variables are numbered starting from `1`.

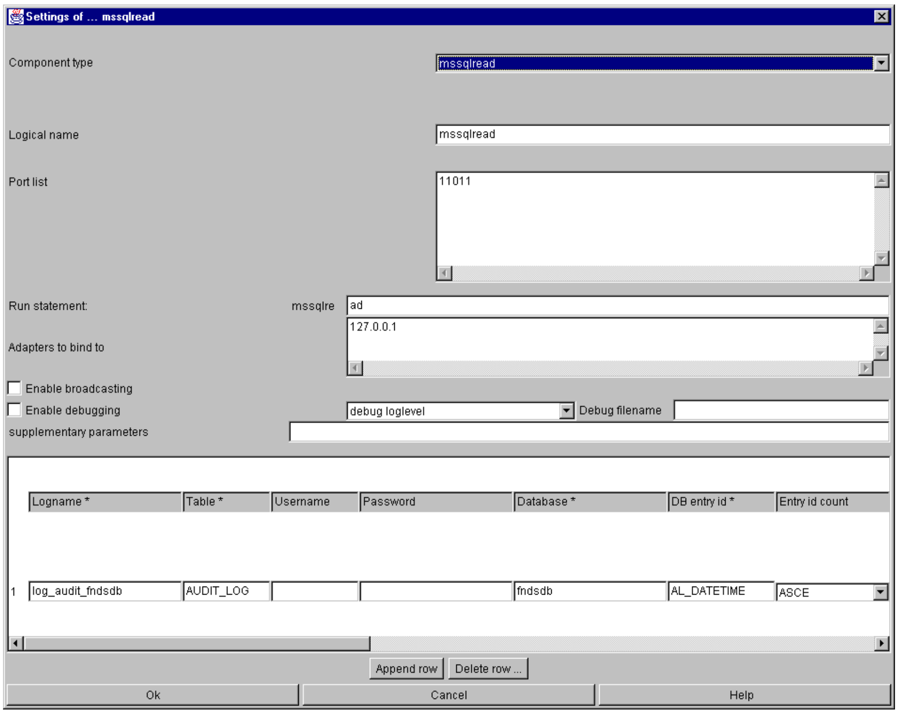
E.g. the first variable `OID` will be stored in the field `OID1`, while the `TYPE` of the third variable will be stored in field `TYPE3`.

mssqlread and oracleread

The MS SQL and ORACLE readers are designed for reading log entries from a database. The configuration of these readers is completely identical.

The `mssqlread` is used to read events from a MS SQL database (MS SQL-Server >= 7) and is only available for the Microsoft Windows platforms.

For reading ORACLE databases , the `oracleread` can be used. It is available for Windows 2000/2003/Xp Professional, AIX and Solaris. It supports oracle databases with version 8.0 or higher.



The Settings of mssqlread dialog

For each table to be read, a database log type has to be defined. When using the CALA configuration GUI, each database log type is displayed as a row in the table on the bottom of the window.

There are several parameters to be set for each database log type, those marked with an asterisk (`*`) are mandatory.

mssqlread/oracleread specific parameters and their setting in the configuration file

These are the configuration lines created from the settings window:

```
mssqlread=run!mssqlread -P 11011 -AB 127.0.0.1,port!11011,targets!msgclsfsrv,
db_log_types!log_audit_fndsdb,conf!port;run;targets;db_log_types
```

```
log_audit_fndsdb=table!AUDIT_LOG,database!fndsdb,db_entry_id!AL_DATETIME;ASCE,
map!AL_PROCESSID;pid;AL_STATUS;AL_STATUS;AL_WORKSTATN_ADDR;workstation;AL_EVENT_PARAM1;msg;
AL_EVENT_PARAM2;PARAM2;AL_EVENT_PARAM3;PARAM3;AL_EVENT_PARAM4;PARAM4;AL_USER;USER,
copy_unmapped!0,timestamp!AL_DATETIME,type!tec;FNDS_MSSQL,defaultclass!FNDS_AUDITLOG_Error,
classmap!ds_class.map;AL_EVENT_ID,pollinterval!30
```

Figure 9-21. Some example configuration lines for `mssqlread` (identical for `oracleread`)

db_log_types instruction

The db_log_types instruction contains a list of all configured database log types to be monitored by this server. Each db_log_type needs to be configured in a own line.

```
db_log_types!<type>{;<type>}
```

Figure 9-22. Format of db_log_types instruction

```
db_log_types!log_audit_fndsdb
```

Example 9-29. Example db_log_types instrcution

This defines the database log type labeled log_audit_fndsdb.

The following instructions are set in the database log type configuration line:

dbuser instruction

The dbuser instruction gives a database user and password to be used when connection to the database. The password is encrypted and can only be configured using the CALA configuration GUI.

```
dbuser!<user>[;<password>]
```

Figure 9-23. Format of dbuser instruction

```
dbuser!tec;11161219140600
```

Example 9-30. Example dbuser instruction

The dbuser parameter is optional. If no user is specified, the reader tries to connect the database without any user and password (system user authentification).

database instruction

This instruction defines to database to be used.

```
database!<database-name>
```

Figure 9-24. Format of database instruction

```
database!fndsdb
```

Example 9-31. Example database instruction

table instruction

The name of the database table is given with this instruction.

```
table!<table-name>
```

Figure 9-25. Format of table instruction

```
table!AUDIT_LOG
```

Example 9-32. Example table instruction

db_entry_id instruction

The db_entry_id instructions sets the table column which is used for entry identification and if the order. The values in this field must be unique for each entry and they must either be descending or ascending.

The first time the SQL reader is started, it must read the whole table to find the newest entry (the one with the highest or lowest value in the id-field).

To find new events, the following SQL statement is used (line 001 shows the statement used for descending, line 002 the statement used for ascending tables):

```
001 select * from table where id-field < id-of-last-entry
002 select * from table where id-field > id-of-last-entry
```

Figure 9-26. SQL statement used to find events

```
db_entry_id!<id-field>;[DESC|ASCE]
```

Figure 9-27. Format of db_entry_id instruction

```
db_entry_id!AL_DATETIME;ASCE
```

Example 9-33. Example db_entry_id instruction

map instruction

To map database field names to event slot names, the map instruction is used. It takes pairs of parameters, each consisting of the database field name and the FIR field name.

```
map!<db-field>;<fir-field>{;<db-field>;<fir-field>}
```

Figure 9-28. Format of map instruction

```
map!AL_PROCESSID;pid;AL_STATUS;AL_STATUS;AL_WORKSTATN_ADDR;workstation
```

Example 9-34. Example map instruction

This parameter is optional. If it is not set, the database field names are used as FIR field names.

copy_unmapped instruction

The copy_unmapped instruction is a boolean instruction which only takes one of the values `0` or `1`.

It defines whether or not mapped database fields (fields, which names do not have a map entry, see above) should be copied into the resulting FIR (FIR field name = database field name) or should be discarded.

```
copy_unmapped![0|1]
```

Figure 9-29. Format of copy_unmapped instrucion

```
copy_unmapped!0
```

Example 9-35. Example copy_unmapped instrucion

The copy_unmapped parameter is optional, it s default value is `1`.

defaultclass instruction

This instruction sets the default class to be used if no class mapping is defined or no matching class is found in the classmap file.

```
defaultclass!<class-name>
```

Figure 9-30. Format of defaultclass instruction

```
defaultclass!FNDS_AUDITLOG_Error
```

Example 9-36. Example defaultclass instruction

This instruction is optional, if it isn't set, the default class is set to `MSSQLREAD_Base` (`mssqlread`) or `ORACLE_Base` (`oracleread`).

classmap instruction

The classmap instruction takes two arguments: a filename and a database field.

The classmap file is an ASCII file containing one mapping instruction per line. A mapping consists of the database fields value and the class name separated by white spaces.

```
classmap!<map-file>;<db-field>
```

Figure 9-31. Format of classmap instruction

```
classmap!ds_class.map;AL_EVENT_ID
```

Example 9-37. Example classmap instrucion

```
LOGON APP_Logon
```

Example 9-38. An example mapfile mapping the class of the created FIR to `APP_Logon` if the map field's value is `LOGON`:

The classmap instruction is optional. The default class (given with the default class instruction) is used if no configuration is given.

type instruction

The type instruction specifies the primary and secondary data type assigned to created events. Its values are taken from the primary data type and secondary data type entry fields. (Either both or none has to be given.)

```
type!<primary type>;<secondary type>
```

Figure 9-32. Format of type instruction

```
type!tec;FNDS_MSSQL
```

Example 9-39. Example type instruction

The type instruction is optional. If it is not given, the default values for primary (`tec`) and secondary type (`mssql`/`oracle`) are used.

prefilt_in and prefilt_out instructions

These instructions sets pre-filters for each database logfile.

Pre-filters are used to discard events before sending them to any other process. The in-filter specifies events that should not be discarded, the out-filter specifies events that should be discarded.

Pre-filters are optionally. If no filter is set, all events are send to the target processes.

```
prefilt_in!<filter_file>
prefilt_out! <filter_file>
```

Figure 9-33. Format of prefilt_in and prefilt_out instructions

```
prefilt_in!sql_in.flt
prefilt_out!sql_out.flt
```

Example 9-40. Example prefilt_in and prefilt_out instructions

The pre-filter keys are the database fields. For further information about pre-filters, refer to the pre-filter section in the *ntevtlogread* description.

Pre-filters are optional. Every event is sent to the targets if no pre-filter is set.

timestamp instruction

There are two possibilities to use the timestamp expression.

- If the database table contains a numerical timestamp field or one from any date or time type, only the fieldname has to be given to this instruction.
- If the timestamp is split into several fields or is given within any text field, a extended usage of timestamp is needed. The field names and text position of the date parts have to be given.

```
timestamp!<db-field>
timestamp!<date-part-id >;<db-field>;<text-position>{;<date-part-id >;<db-field>;<text-position>}
```

Figure 9-34. Format of timestamp instruction

date-part-id can be one of $YEAR, $MONTH, $DAY, $HOUR, $MINUTE and $SECOND.

The text position argument is given similar to the text position in message map description, see for details.

```
timestamp!date
timestamp!AL_DATETIME
timestamp!datestr;$YEAR;F0L3;datestr;$MONTH;F4L5;datestr;$DAY;F6L7;timestr;$HOUR;F0L1;timestr;$MINUTE;F2L3;timestr;$SECOND;F4L5
```

Example 9-41. Examples for the timestamp instruction

The second sample is a definition for a table having two columns for the timestamp: the datestr column containing the date in the format YYYYMMDD and timestr column with the time in the format HHMMSS. (E.g.: datestr= 20020415 and timestr=084933 for 8:49:33 on april 15th 2002)

mssqlread and oracleread command line parameters

These parameter can be set in the field supplementary parameters

parameter	example	description	default value
-D <db-host>	-D foo.bar.com	Sets the name of a remote databserver. .	localhost
-H <hostname>	-H thud.grunt.net	Changes the value of the hostname field in the created events.	tcp/ip hostname
-L <timeout>	-L 60	Set the timeout for database login (in seconds).	60
-E	-E	Skip old events: Discard all events found when opening the database the first time.	disabled

jdbcread

The jdbc reader implements a generic database reader component similar to `mssqlread` and `oracleread`. It uses the java jdbc interface to access the database and can therefore be used to access any database for which a jdbc driver is available.

jdbcread specific parameters and their setting in the configuration file

The configuration of the `jdbcread` is very similar to the configuration of `mssqlread` and `oracleread`.

There are few differences:

- Additional command line parameters for the java virtual machine (e.g. classpath settings) can be passed before the component parameters are given. This parameters must at least add the `calaJNI.jar` file to the classpath. End the java parameters with two dashes.
- The database name consists of three parts:
  - the `JDBCRead` java classname (`de/cenit/eb/sm/cala/jdbc/reader/DefaultCalaJDBCRead` by default)
  - the jdcb driver class
  - the jdbc database URL

  These parts are separated by colons.

```
de/cenit/eb/sm/cala/jdbc/reader/DefaultCalaJDBCRead:com.mysql.jdbc.Drive
r:jdbc:mysql://localhost/fndsdb
```

Example 9-42. An example database string for `jdbcread`

```
jdbcread=run!jdbcread -Djava.class.path=calaJNI.jar;mysql-connector.jar
-- -P 11011 -AB 127.0.0.1,port!11011,targets!jdncread,db_log_types!log_
audit_fndsdb,conf!port;run;targets;db_log_types

log_audit_fndsdb=table!AUDIT_LOG,database!de/cenit/eb/sm/cala/jdbc/read
er/DefaultCalaJDBCRead:com.mysql.jdbc.Driver:jdbc:mysql://localhost/fnd
sdb,db_entry_id!AL_DATETIME;ASCE,map!AL_PROCESSID;pid;AL_STATUS;AL_STAT
US;AL_WORKSTATN_ADDR;workstation;AL_EVENT_PARAM1;msg;AL_EVENT_PARAM2;PA
RAM2;AL_EVENT_PARAM3;PARAM3;AL_EVENT_PARAM4;PARAM4;AL_USER;USER,copy_un
mapped!0,timestamp!AL_DATETIME,type!tec;FNDS_MSSQL,defaultclass!FNDS_AU
DITLOG_Error,classmap!ds_class.map;AL_EVENT_ID,pollinterval!30
```

Example 9-43. An example `jdbread` configuration using a mysql connector for connection to the local database `fndsdb`

The jdbc reader needs a java 1.4 or higher virtual machine to be in the library path.

For further configuration information refer to section *mssqlread and oracleread*.

msgclsfsrv

The processing server `msgclsfsrv` is the core component of CALA. Due to its central function it features some specific configuration parameters, which are explained in the following chapter.

To provide better understanding of these causal relationships, a few of the key terms and their functions are explained first.

Comment: Several `msgclsfsrv` processes can be implemented in parallel manner on a system. Since version 2.1, this no longer requires a unique name for the program binary.

Definition MessageMap File

General: The value/information defined in a Message Map file is used to manipulate/process events (FIRs). This means that a kind of linear control function is implemented with the Message Map definitions.

The structure of a Message Map file is defined with a Message Classification Type (MCT), i.e. the format is not firmly specified.

Definition RulesMap File

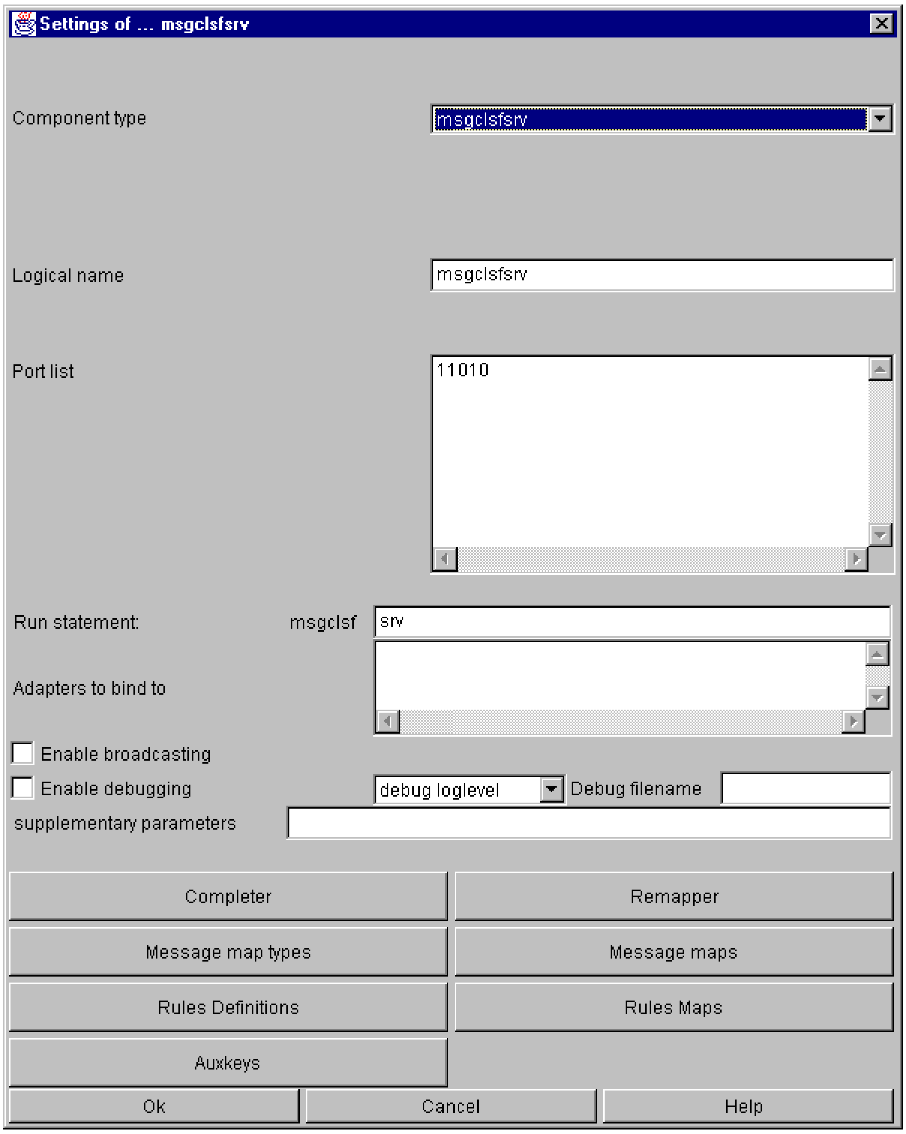
General: A Rules Map file is a description of rules for event handling. Rules are used to handle correlations between correlating events and to perform actions/modifications on events

in dependency of previous by received events.

The structure of a Rules Map file is defined with a Rules Map Type (RMT), i.e. the format is not firmly specified.

The large number of possible parameters available for the Message Classification Server (`msgclsfsrv`) configuration makes it necessary to have a hierarchical structure in the configuration window.

The basic msgclsfsrv window

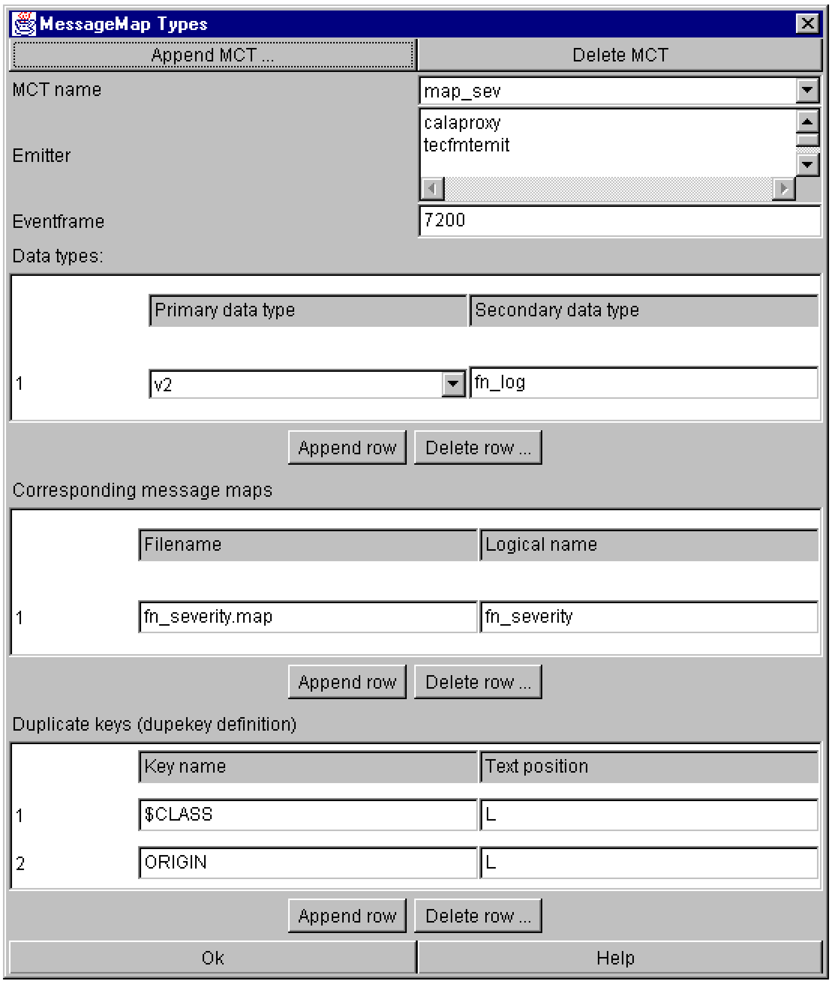


The Settings of msgclsfsrv dialog

General settings for `msgclsfsrv`, such as logical name and ports can be configured in the basic window.Further configuration can be set in the sub-windows.

The Message Map Types window

In order to set up MCT definitions (Message Classification Types), the appropriate field must be selected. At this point, the Message Classification Type (MCT) window opens.



The MessageMap Types dialog

Definition of MessageMap Classification Type (MCT)

A message map classification type (MCT) is a logical unit for message mapping. It is used to process data streams (specified by primary and secondary data type) for a list of emitters.

All configured MCTs are working parallel, this means, that each incoming event is put into every MCT which is defined for its stream. The MCTs will process the event and send it to the emitters defined on it. If no MCT is found for an event, the event is not modified and sent to all targets of `msgclsfsrv`.

A MCT definition can use one or more message map files which are given in the table Corresponding message maps . For more information about message map files see *The Message Map definition window*.

MCT parameters and their setting in the configuration file

Each MCT is configured in a line of its own within the configuration file `logctlsrv.conf` . The message classification servers configuration line only contains references to the MCT definitions.

```
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!tecfmtemit;cmdemit; calaproxy;reportemit;snmpemit;smtpemit,types!map_sev,conf!port;run; targets;types
map_sev=type!v2;fn_log,handledby!calaproxy;tecfmtemit,msgmaps!fn_severity.map;fn_severity,eventframe!7200,dupekey!$CLASS;L;ORIGIN;L
```

Example 9-44. These are the configuration lines created for the message map classification type:

`types` instruction

The types instruction contains a list of all message map classification types to be used by the message classification server

```
types!<mct-name>{;<mct-name>}
```

Figure 9-35. Format of `types` instruction

```
types!map_sev
```

Example 9-45. Example `types` instruction

MCT configuration line

```
<mct-name>=type!<primary type;secondary type>,handledby!<emitter name>{;<emitter name>}, msgmaps!<msgmap filename;logical name>{;<msgmap filename>;<logical name>},eventframe!<seconds>, dupekey!<field name;text position>{;<field name;text position>}
```

Figure 9-36. Format of mct configuration line

```
map_sev=type!v2;fn_log,handledby!calaproxy;tecfmtemit,msgmaps!fn_severity.map;fn_severity,eventframe!7200,dupekey!$CLASS;L;ORIGIN;L
```

Example 9-46. Example mct configurationline

type instruction

This describes the data type combination (primary / secondary) which is applied to this MCT definition. These parameters are set by the entry fields Primary data type and Secondary data type

```
001  type!<primary type>;<secondary type>
```

Figure 9-37. Format of mct type instruction

```
001  type!v2;fn_log
```

Example 9-47. Example mct type instruction

handledby instruction

The handledby instruction describes the following component(s) which contain(s) results of this type as a target. If no handledby parameter is defined, all FIRs are propagated to all defined emitters. The emitters for the MCT are taken from the textbox Emitter

```
001  handledby!<emitter name>{;<emitter name>}
```

Figure 9-38. Format of mct `handledby` instruction

```
001  handledby!calaproxy;tecfmtemit
```

Example 9-48. Example mct handledby instruction

Explanation: The FIRs of this type are propagated to `calaproxy` and `tecfmtemit` .

msgmaps instruction

The msgmaps instruction gives a list of message map file to be used from this MCT. Each message map file must be specified by its filename and a logical identifier, which is used to define the file s format later.

In the settings window the message map files are set in the table corresponding message maps

```
001  msgmaps!<msgmap filename>;<logical name>{;<msgmap filename>;<logical nam
...  e>}
```

Figure 9-39. Format of mct msgmaps instruction

```
msgmaps!fn_severity.map;fn_severity
```

Example 9-49. Example mct msgmaps instruction

Explanation: The above example defines the Message Map fn_severity whose Message Map data are stored in the `fn_severity.map` file.

eventframe instruction

The event frame describes a time frame. Events received during this time frame are checked for duplicates.

The default value is 3600 seconds = 1hr. Event frame settings can be indicated separately for every MCT (Message Classification Type). If no event frame is given, or event frame is set to 0 seconds, duplicate detection is disabled.

```
eventframe!<seconds>
```

Figure 9-40. Format of mct eventframe instruction

```
eventframe!7200
```

Example 9-50. Example mct eventframe instruction

Example explanation: This event frame setting defines a time interval of 2 hours (7200 seconds) for duplicate detection purposes.

dupekey instruction

A dupekey is a key which is used to recognize duplicate events. It is created from one or more parts of an event. The event fields used within this key are defined in the table Duplicate keys (dupekey definition)

```
dupekey!<fieldname>;<text position>{;<fieldname>;<text position>}
```

Figure 9-41. Format of mct dupekey instruction

```
dupekey!$CLASS;L;ORIGIN;L
```

Example 9-51. Example dupekey instruction

The dupekey instruction defines the field names (slots) which should be used for duplication detection purposes. Sub-parameter `text-position` defines which part of the key field should be used for duplication detection purposes.Further information on the text position can be found in chapter *msgclsfsrv Text Formatting*.

The example defines a combination of internal slot `$CLASS` and the slot `ORIGIN` as a duplication detection key. L means the entire field contents (starting from the left) are used.

If the dupekey is omitted, duplicate detection is disabled unless the `$DUPEKEY` field is mapped in the FIR during the classification process. In this case, the key referenced by the `$DUPEKEY` is used. This must be entered in the configuration list and in the `AUXKEYS` list (see *Auxkeys definition window*).

```
mqlogmct=type!v2;MQ,handledby!tecfmtemit,msgmaps!suppressed.map;suppress
map1;mq.map;mqmap1;severity.map;redefsevmap1,dupekey!errcode;L;msg;L
```

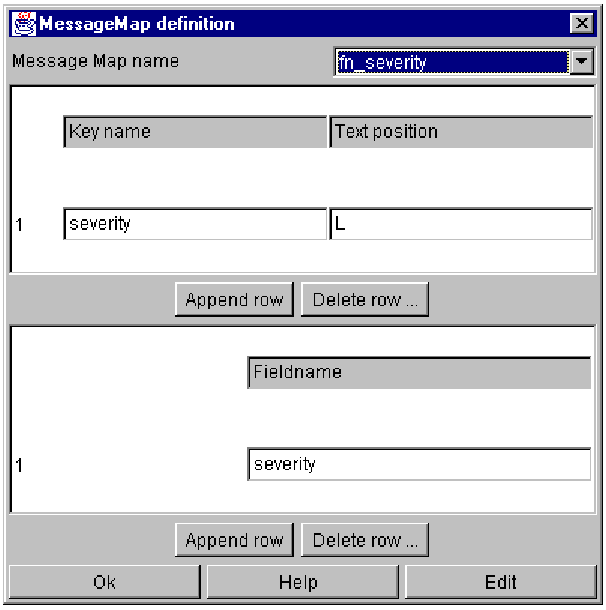
Example 9-52. Another example of a complete MCT definition

Explanation of the example:

- One MCT definition is defined with the name `mqlogmct`
- Primary type is `v2`, secondary type is `MQ`.
- Data is sent to `tecfmtemit` for further processing. (handleby sub-parameter)
- The Message Map definitions `suppressmap1` (described by the file `suppressed.map`), `mqmap1` (described by the file `mq.map`) and `redefsefmap1` (described by the file `severity.map`).
- The slots `errcode` and `msg` are used for this MCT type for duplicate detection and designator designates the use of the complete string (from left) for duplicate detection

The Message Map definition window

Once the MCT definition window has been closed by pressing the OK button, the Message Map file relating to the MCT must be defined. This configuration is performed using sub-menu Message maps



The message map definition dialog

This window defines the format of the message map files. To edit the message map file contents, press the Edit button. Message Map parameters and their setting in the configuration file

A format description must exist for every Message Map file stored in the configuration file `logctlsrv.conf` .

```
<log. msgmap name>=key!<key name>;<text position>{;<key name>;<text posi
tion>}, fields!<field>{;<field>}
```

Figure 9-42. Format of message map definition

```
fn_severity=key!severity;L,fields!severity
```

Example 9-53. Example message map definition

The slot names following the keyword key (at least 1 slot) are related to the MessageMap assignment. The slot names which follow the keyword fields are mapped using the values for the relevant line/column.This means that all slot names can be indicated as key and as field as well.

Explanation of the example

The format of the Message Map definition with the name `fn_severity` contains the key `severity` and the `severity` field is also used as a mapable field.

> **Note:** This Message Map definition is used for implementing non-Tivoli severity values on severity values used by Tivoli.

The file which describes this format possesses has following format:

```
INFO HARMLESS
WARN WARNING
ERROR CRITICAL
Error CRITICAL
Notice HARMLESS
Warning WARNING
```

Example 9-54. An example mapfile for the above map definition

The content of the slot severity for all events manipulated with this Message Map is processed in accordance with this conversion table. E.g.: If the `severity` field on an event being processed contains the value `Notice` , the contents of the severity slot will be mapped to `HARMLESS` after processing.

The CALA-processing server defines a few internal slots which are required for internal processing, but which can also be forwarded as external slots (to the T/EC). The most important internal slot is `$CLASS` which represents the event class name of the T/EC (Class).

Default Mapping

If the map file contains a line with the special string `__DEFAULT_KEY__` as key, the mapping defined in this line is used for events which do not match for other mappings.

> **Note:** The default entry can be used anywhere in the map file, if several occurences are found, the latest one is used.

Deleting slots

Event fields can be deleted by setting it s value to `*`. To delete a complete event, map it s class to `**.` This works with message maps, rules maps and re-mapping.

Special slots for duplicate detection

To implement duplicate detection, 5 reserved internal field names (internal slots) are implemented, `$SEVFLD`, `$ESCAT`, `$ESCLEV`, `$CLASS` and `$DUPEKEY`.

These slots have the following meaning:

field name	description
$SEVFLD	Describes the name of the field which implements the severity field.
$ESCAT	Describes the number of identical events which are suppressed using duplicate detection.
$ESCLEV	Describes the value of the severity level $ESCAT for identical events and escalates the $ESCAT+1 event to this level.
$CLASS	The class of the event (FIR)
$DUPEKEY	Name of a duplicate key, see also section *Auxkeys definition window*
$ESCCNT	Escalation counter holds the no. of escalations that occurred (can be used in further processing).

Another example for a complete message map definition

```
mqmap1=key!errcode;L,fields!$CLASS;$SEVFLD;severity;$ESCAT;$ESCLEV
```

Example 9-55. Another example for a complete message map definition

Explanation of the example

- The Map Definition `mqmap1` message uses the `errcode` field for duplicate detection. The entire contents of this field are considered (text position `L`)
- The first field in the field instruction `$CLASS` (internal slot, see *above*) is mapped with this Message Map in relation to the errcodes (key field is `errcode`).
- The second field, `$SEVFLD`, describes the name of the severity field (the severity field can be any field within the FIR).
- The third field is an assignment for the field `severity`. In this example the field `severity` is also used for escalation, so the mapping is overwritten if an escalation occurs.
- The fourth field, `$ESCAT`, contains the number of identical events (identical in terms of the defined dupekey fields in the MCT definition) which have to be suppressed.
- The last defined field `$ESCLEV` defines the value of the field with which the field described in `$ESCFLD` is mapped if an escalation occurs. An escalation occurs, if (within one event frame) more than in `$ESCAT` defined events of the same type arrived.

```
AMQ9001 AMQ_CALA_AMQ9001 severity MINOR 10 CRITICAL
```

Figure 9-43. Excerpt from the related message map file

Using this message map, events are processed as follows:

- If a FIR is received, whose `errcode` slot contains the value `AMQ9001`, the Eventclass (`$CLASS` slot) is mapped to `AMQ_CALA_AMQ9001`.

- The field which describes the severity is called `severity` (T/EC standard).
- Default severity is mapped to `MINOR`
- The 2nd to 9th event with `errcode` set to `AMQ9001` will be suppressed.
- If 10 events are identified as identical within the defined period of time (refer to *event frame instruction*), the severity level is increased to `CRITICAL` and a new event is initiated (FIR).

Operations on FIR fields per Message Maps

A normal message map contains just strings, which were assigned to the specified FIR fields. For some applications, it may be useful to do some arithmetic operations on fields. If the mapping value has one of the following pre- or postfixes, a special action is performed.

Prefix/Postfix	Action
+	string concatenation
++	addition
--	subtraction
**	multiplication
//	division

All these operations can be used as postfix or prefix. Usage as prefix (e.g. `++5`) means

<current value> <operation> <value in mapfile>

while the operator given as postfix means

<value in mapfile> <operation> <currentvalue>

E.g. if the value of the field `counter` is `5`, and the mapping value for this field is --3, the resulting value will be `2` (= 5-3). If the mapping value is 8--, the result will be `3` (= 8-5).

All operators except the string concatenation operator (+), only work with numeric values. If any operation value is not numeric or a division by zero occurs, the field will be set to the string `NaN` (*N*ot *a* *N*umber).

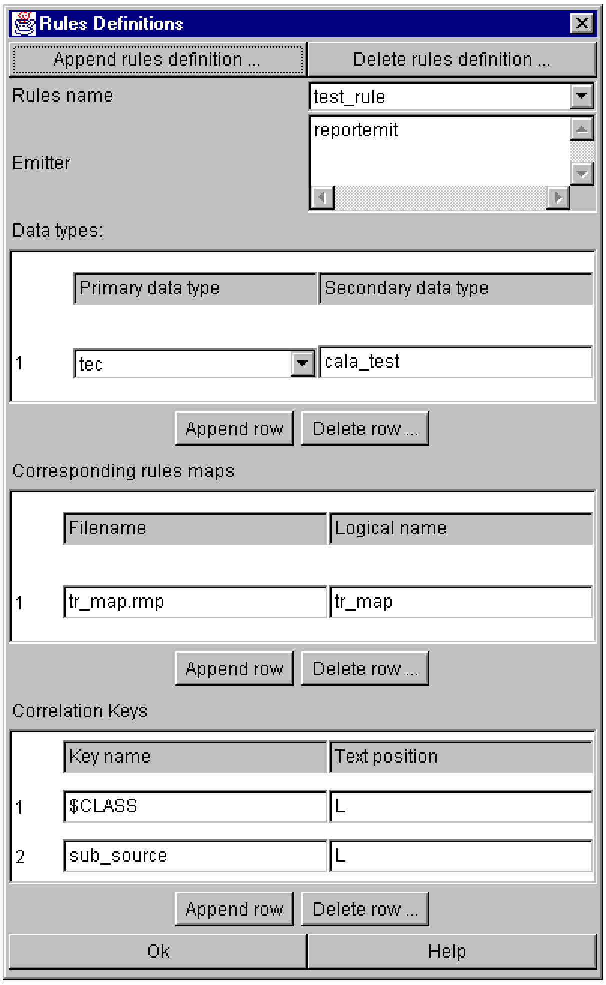
Instead of provoding a fixed value, it is also possible to give a reference to any other field of the FIR. A reference is given with the prefix & followed by the field's name. If no operation is defined, the reference just copies the value of the referenced field.

If any of these operators appears in a string, which should be assigned to the field, this string must be written in quotes. If a quote appears in a quoted string, it has to be escaped with backslashes (\).

Some examples:

- ++5 increases the field's value by `5`.
- 5++ increases the field's value by `5` (same as above).
- --1 decreases the field's value by `1`.
- 10-- sets the field's value to `10` minus its old value.
- 2** doubles the field's value.
- ++&count increases the field by the value stored in the field named `count`.
- &count is replaced by the value of the field count.
- PREFIX+ adds the string `PREFIX` to the beginning of the field's value.
- +POSTFIX appends the string `POSTFIX` to the end of the field's value.
- "--&count" sets the field's value to the value of the `count` field decreased by one.

The Rules definition window



The rules definition dialog

Definition of Rules Map Type (RMT)

A rules map type is a logical unit to handle correlation between events. Like message map types, rules map types are used to process data streams for one or more emitters, but a RMT can also handle several data streams having different primary and secondary data types.

RMT parameters and their setting in the configuration file

The configuration of rules map types is similar to configuration of message map types. This is the configuration line of the example above:

```
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!tecfmtemit;cmdemit;calaproxy;reportemit;snmpemit;smtpemit,types!map_sev,rules!test_rule,conf!port;run;targets;types;rules
test_rule=for!reportemit,type!tec;cala_test,corrkey!$CLASS;L;sub_source
```

```
;L,rulesmaps!tr_map.rmp;tr_map
```

Example 9-56. An example msgclsfsrv configuration with a rules map type definition

rules instruction

The rules instruction lists the RMTs used by the message classification server.

```
rules!<rmt name>{;<rmt_name>}
```

Figure 9-44. Format of rmt rules instruction

```
rules!test_rule
```

Example 9-57. Example Format of rmt rules instruction

RMT configuration line

```
<rmt-name>=for!<emitter>{;<emitter>},type!<primary type>;<secondary type
>{;<primary type>;<secondary type>},rulesmaps!<rulesmap filename>;<logi
cal name>{;rulesmap filename>;<logical name>}, corrkey!<field>;<text po
sition>{;<field>;<text position>}
```

Figure 9-45. Format of rmt configuration line

```
test_rule=for!reportemit,type!tec;cala_test,corrkey!$CLASS;L;sub_source;
L, rulesmaps!tr_map.rmp;tr_map
```

Example 9-58. Example rmt configuration line

RMT configuration line parameters

for instruction

The for instruction sets the targets the rule is to be used for. If no for parameter is defined, this rule is used for every target. The target emitters are listed in the textbox Emitters.

```
for!<emitter name>{;<emitter name>}
```

Figure 9-46. Format of rmt for instruction

```
for!reportemit
```

Example 9-59. Example rmt for instruction

Explanation: This rule is processed for each event queued for the emitter `reportemit`.

type instruction

This describes the data type combinations (primary/secondary) which are applied to this RMT definition.

Events which do not match any of these types are passed through the rules processing. Unlike the type instruction of message map types, the type instruction for rules map types is able to process more than one data stream therefore the primary and secondary data types are not given in text fields but the table Data types.

```
type!<primary type>;<secondary type>{;<primary type>;<secondary type>}
```

Figure 9-47. Format of rmt type instruction

```
type!tec;cala_test
```

Example 9-60. Example of rmt type instruction

rulesmaps instruction

The rules map instruction gives a list of rules map files to be used with this RMT. Each rules map file must be specified by its filename and a logical identifier, which is used to define the files format later. (See table Corresponding Rules Maps in settings window.)

```
rulesmaps!<rulesmap filename>;<logical name>{;<rulesmap filename>;<logic
al name>}
```

Figure 9-48. Format of rmt rulesmap instruction

```
rulesmaps!tr_map.rmp;tr_map
```

Example 9-61. Example rmt rulesmap instruction

Explanation: The above example defines the Rules Map `tr_map` whose Rules Map data are stored in the rules map `tr_map.rmp` file.

corrkey instruction

The corrkey instruction defines the field names (slots) which should be used for correlation detection purposes. Corrkeys are configured and used like dupekeys , refer to the *dupekey* instruction explained *above*.

```
corrkey!$CLASS;L;sub_source;L
```

Example 9-62. Example of rmt corrkey instruction

Another example of a complete RMT definition

```
samplerule1=for!tecfmtemit,type!v2;MQ;tec;ntevtlog,rulesmaps!rulesmap1.r
mp;rulesmapfmt1;rulesmap2.rmp;rulesmapfmt2,corrkey!$CLASS;L;msg;L
```

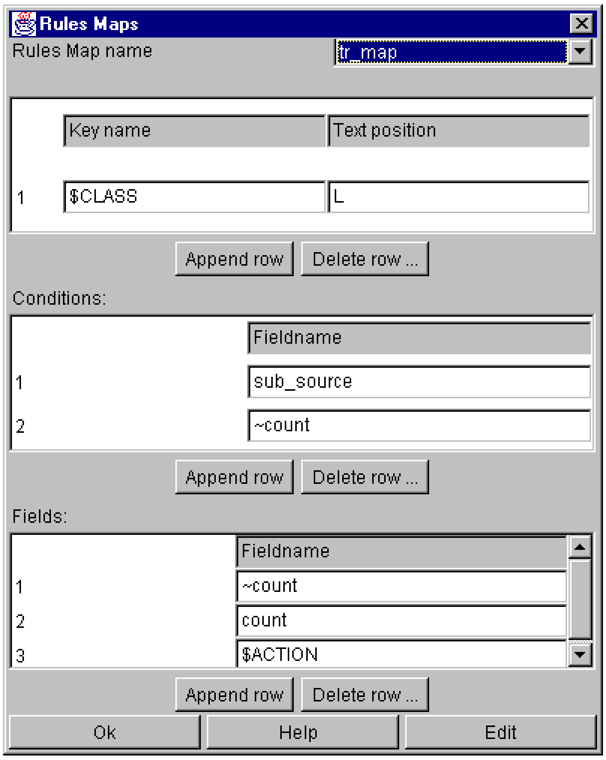
Example 9-63. Another example of a complete RMT definition

Explanation of the example:

- One RMT definition with the name `samplerule` is defined.
- Each event which is queued for `tecfmtemit` is passed through this rule
- This RMT is used on events with primary type `v2`, secondary type `MQ` or primary type `tec` and secondary type `ntevtlog`
- The rules map definitions are rules `mapfmt1` (described by the file `rulesmap.rmp`), `rulesmapfmt2` (described by the file `rulesmap2.rmp`)
- The event's class and the slot `msg` are used for this RMT type for correlation detection. Designator designates the use of the complete string (from left) for correlation detection.

The Rules maps window

To specify the format of a rules map file, open the rules maps window. This window looks very similar to the message map definition window, but has an additional table Conditions. To load the rules map file into an editor, press the Edit button.



The rules maps dialog

Rules Map Parameters and their setting in the configuration file

A format description must exist for every Rules Map file stored in the configuration file `logctlsrv.conf` .

```
<log. rulesmap name>=key!<field>;<text position>{;<field>;<text position
}, conditions!<field>{;<field>},fields!<field>{;<field>},ext_conditions
!<field>;<format>{;<field>;<format>}
```

Figure 9-49. Format of rules map definition

The new extended conditions parameter (`ext_conditions`) works like the `conditions` parameter with the enhancement, that only parts of a field can be used for a condition. This for example is useful, if only a part of a field is interessting. For a definition of the text format see *msgclsfsrv Text Formatting* .

If ext_conditions and conditions are both used in the same rules map, the entries for the extended contion fields have to appear after the conditions entries.

```
tr_map=key!$CLASS;L,conditions!sub_source;~count,fields!~count;count;$ACTION
```

Example 9-64. Example rules map definition

The slot names following the keyword key (at least 1 slot) are related to the Rules Map assignment.

The slot names given as conditions (keyword conditions) are checked before executing the rule. If the rule is executed (all conditions are fulfilled), the slot names which follow the keyword fields are mapped using the values for the relevant line/column. This means that all event slot names can be used for key, for condition and for field.

To understand how the rules engine works, some further definitions are needed.

Definition Base Event

Any received event can be kept in memory to be compared with the new received events for correlation handling. These events are called base events.

A fileservers client has to be monitored. If the fileserver is not reachable, an event of class FILESERVER_DOWN is received. If the fileserver is reachable again, an event FILESERVER_UP is expected.

When receiving a FILESERVER_DOWN, a base event is created, which will be deleted if a FILESERVER_UP event occurs.

If more than one clients send a FILESERVER_DOWN, the base event can be modified to hold e.g. a list of all clients which tried to access the server.

Example 9-65. Example of rules engine usage with base events

Slots of the base event can also be used in the conditions and fields statements with using a ~ as prefix. (E.g. to check against the field count of the base event, ~count should be written.)

Reserved fieldnames and their meaning

The rules engine defines a few internal slots which are required for internal processing. Most of them can also be forwarded as external slots (to the T/EC).

The slots that are used internal only are `$TIMER` and `$ACTION` . They should only be set by the rules map, but not be used in the correlation key, the conditions statement or as a reference.

field name	description

field name	description
$TIMER	If this field is set by a rules map to any positive value secs, a timer is started for the base event. If no correlating event is received within secs seconds, the timer is stopped and an event of class _TIMER_<class of base event> is created, which can be processed by the rules map. The timer is also stopped, if any correlating event is received. Setting $TIMER to any value <= 0 deactivates it.
$ACTION	This fields specifies which action on the event storage has to be done, for further information see below.
$CREATION_TIME	This slot is set only in base events and contains the time,the base event was created (in seconds since 1.1.1970).
$TIME_NOW	This slot contains the current time in seconds since 1.1.1970.
$CORRKEY	Name of the correlation key to use (usage is analog to $DUPEKEY in message maps)

By setting the field `$ACTION`, the base event storage can be handled. If set to one of the following fields, the described action will be started:

_ CREATE:BASE _

copies the received event into the event storage (to be a new base event), no event will be sent to the targets.

_ DISCARD:BASE _

deletes the correlating base event from the storage (no event will be sent to the targets)

_ DISCARD:CURRENT _

deletes the currently received event and leaves the event storage untouched. (no event will be sent to the targets)

By setting $ACTION to one of the following values, any modification of the event storage is possible and a FIR will be sent to the targets:

_CREATEANDSEND:<which> _

like _ CREATE:BASE _ but also sends an event to the targets

SENDANDDISCARD:<which>

like _ DISCARD:BASE_ but also sends an event to the targets

SEND:<which>

sends an event to the targets (does not create or delete any base event)

`<which>` can be set to `BASE` to send the base event, or to `CURRENT` to send the current received event. If `<which>` is neither set to `BASE` nor to `CURRENT`, its value is interpreted as a slot within the currently received (and modified) event. This slot must contain a FIR-string in the format:

```
<field>=<value>{;<field>=<value>}
```

The created FIR has the same class and data types like the FIR it was created from (the currently received one). To change the class or data types, the special fields `$CLASS`, `$PRITYPE` and `$SECTYPE` can be used.

Condition values

Instead of mapping fields (which are used like message map mapping fields), condition fields are only compared with the event s fields the value of the fir field is not modified. The following operators (given as prefix with the fields value) can be used (single or combined):

>

is true if the event field s value is greater than the following one (for numbers only)

<

is true if the event field s value is lower than the following one (for numbers only)

=

is true if the event field s value is the same as the following one (for numbers and strings)

!

inverts the following operator, if ! is given without any value, it returns true, if the event field does not exist

*

is true for all values if the event field exists

If no operator is given, a string comparison is performed.

<10 is true if the value of the FIR field is lower than 10

!>12 is true if the value of the FIR field is not greater than 12 (same as <= 12)

!=4 is true if the value of the FIR field is not 4 (same as <>4)

!=four is true if the value of the FIR field contains not the string four

=four is true if the value of the FIR field contains the string four

four is true if the value of the FIR field contains the string four

Example 9-66. Some example conditions

Rules Map Example

Now we have the background information to understand the rules example. Here s the configuration line again:

```
tr_map=key!$CLASS;L,conditions!sub_source;~count,fields!~count;count;$ACTION
```

Example 9-67. Example rules map definition

The rules map file may have the following entries (# prefixes comment lines)

```
001 # key condition condition field field special field
002 #$CLASS (key) sub_source ~count ~count count $ACTION
003 CALA_Testevent ascfileread ! 1 ~count _ CREATEANDSEND:CURRENT_
004 CALA_Testevent ascfileread 1 2 ~count _ DISCARD:BASE_
005 CALA_Testevent * ! 1 ~count _ CREATEANDSEND:CURRENT_
006 CALA_Testevent * * ++1 ~count _ SEND:CURRENT_
```

Example 9-68. Example rules map file

Explanation of this example:

- The only class handled by this rules map is `CALA_Testevent` . (See first column. Such events are generated when calling **logctlcmd test**.)
- The lines 003-004 implement a filter for test events generated from `ascfileread`: every second event is suppressed.
- Line 003 means: If event is from class `CALA_Testevent` and `sub_source` is `ascfileread` and the field `count` is not set in the base event (which is true if no base event is currently created) then create a base event and set its field `count` to `1`, copy the base event's `count` field to the currently received event and send this event to the target.
- Line 004 means: If event is of class `CALA_Testevent` and `sub_source` is `ascfileread` and field `count` of the base event is set to `1`, then discard both, the base event and the currently received one. (The settings of the `count` fields don t have any effect in this case).
- The two lines 005-006 implement a counter for test events. The field `count` will be set to the number of test events which occurred for the same `sub_source`.

Remember that base events are found with the correlation key, which is defined to $CLASS;L;sub_source;L in this example (see description of Rules definition window).

What happens if any test events from ascfileread arrive?

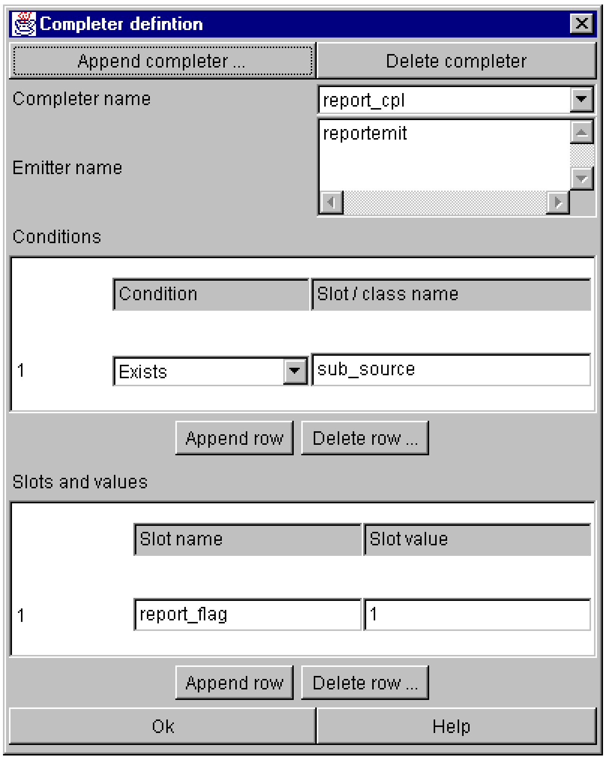
The first test event from `ascfileread` creates a base event and is forwarded to the emitter (it's `count` field is set to `1`). The base event is created with the correlation key `CALA_Testeventascfileread`.

When a second test event from `ascfileread` is received, the base event is found with count=1 and therefore, both events (the arrived and the base event) are discarded.

The third event will be treated like the first, because no base event with the correlation key `CALA_Testeventascfileread` will be found . The fourth one gets the same processing like the second, and so on.

Completer definition window

A `completer` delivers final completion of event processing, independently of primary or secondary data type.



The completer definition dialog

Completers are used to complete events, i.e. slot contents are mapped, deleted or created. Completer instructions are performed after processing of message and rules maps.

Completer Parameters and their setting in the configuration file

These are the configuration lines created for the completer definition shown above:

```
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!tecfmtemit;cmdemit
;calaproxy;reportemit;snmpemit;smtpemit,completers!report_cpl,types!map
_sev,rules!test_rule,conf!port;run;targets;completers;types;rules
report_cpl=for!reportemit,fill!report_flag;1,if!sub_source
```

Example 9-69. An example msgclsfsrv configuration using a completer

completers instruction

The `completers` instruction holds a list of completers used by the message classification server.

```
completers!<completer name>{;<completer name>}
```

Figure 9-50. Format of completers instruction

```
completers!report_cpl
```

Example 9-70. Example completers instruction

Completers configuration line

```
001 <completer name>=for!<emitter name>{;<emitter name>},fill!<slot name>;<value>{;<slot name>;<value>},unless!<slot name>{;<slot name>}
002 <completer name>=for!<emitter name>{;<emitter name>},fill!<slot name>;<value>{;<slot name>;<value>},if!<slot name>{;<slot name>}
```

Figure 9-51. Format of completers configuration line

The first format (line 001) maps one or more slots with the defined value(s) provided that one or more slots do not exist. The second format (line 002) maps one or more slots with one or more defined values provided that one or more slots exist.This mechanism is typically used when setting default slots or when deleting slots which are not required.

Any desired number of if and unless instructions can be used combined in a completer instruction.

```
report_cpl=for!reportemit,fill!report_flag;1,if!sub_source
```

Example 9-71. Example completers configuration line

Explanation of the example:

Completer report_cpl maps the slot report_flag to if a slot sub_source exists.

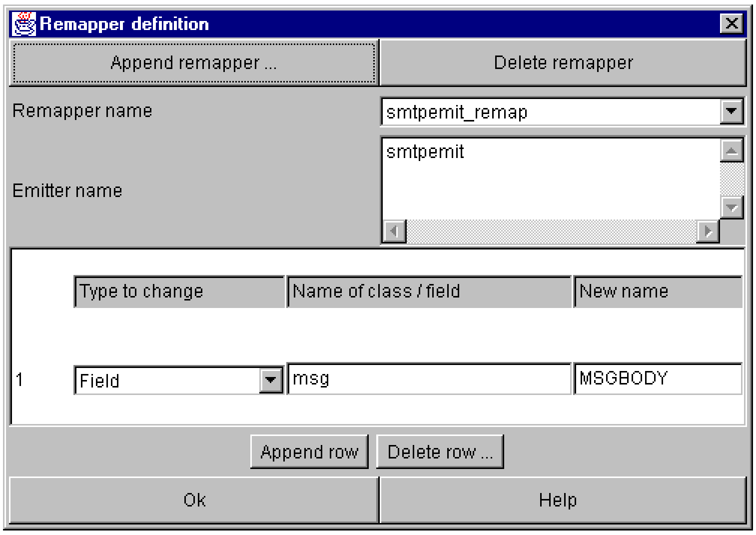
```
generalcpl1=for!tecfmtemit,fill!source;CALALOGS,unless!source
```

Example 9-72. Another completer example

Explanation:

Completer generalcpl1 maps the source slot with the CALALOGS value provided that this slot does not exist.

Remapper definition window



The remapper dialog

Remappers are used to re-map class names and field names, although this does not apply to their contents. The Remapper works in the same way as the completer on all events (FIRs).

Remapper parameters and their setting in the configuration file

The following configuration lines are created for the remapper definition in the window above:

```
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!tecfmtemit;cmdemit
;calaproxy;reportemit;snmpemit;smtpemit,completers!report_cpl,remappers
!smtpemit_remap,types!map_sev,rules!test_rule,conf!port;run;targets;com
pleters;remappers;types;rules
smtpemit_remap=for!smtpemit,fieldalias!msg;MSGBODY
```

Example 9-73. An example msgclsfsrv configuration using a remapper

remappers instruction

The remappers instruction holds a list of remappers used by the message classification server.

```
remappers!<remapper name>{;<remapper name>}
```

Figure 9-52. Format of remappers instruction

```
remappers!smtpemit_remap
```

Example 9-74. Example remappers instruction

Remappers configuration line

For each remapper, there is a remapper configuration line.

```
<remapper name>=for!<emitter name>{;<emitter name>},fieldalias!<old field name>;<new field name>{;<old field name>;<new field name>},classalias!<old class name>;<new class name>{;<old class name>;<new class name>}
```

Figure 9-53. Format of remapper configuration line

```
smtpemit_remap=for!smtpemit,fieldalias!msg;MSGBODY
```

Example 9-75. Example remapper configuration line

Both field- and class-alias are optional but at least one alias has to be defined. The example remapper renames the field `msg` to `MSGBODY` before sending the event to the emitter `smtpemit`.

```
remapclass=for!reportemit,classalias!Logfile;CALA_Logfile
```

Example 9-76. Another example for a remapper configuration line

Explanation of the example

The Remapper `remapclass` re-maps the class name logfile to the `CALA_Logfile`.

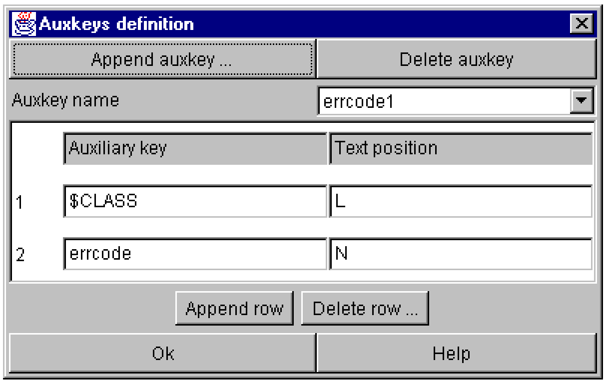
Auxkeys definition window

The auxkeys parameter defines an additional (list of) key(s) used for duplicate or correlation detection. Auxkeys are used to specify different dupe- or corrkeys for each map or rule. To use an auxkey as dupekey, a map has to set the field $DUPEKEY with the auxkeys label. A rules map must set the $CORRKEY field with the auxkeys label for this.

This is an example message map which sets different dupekeys for each map (assuming that the aukeys `dupekey_calatest` , `dupekey_diskspace` and `dupekey_su_failure` are defined in `logctlsrv.conf` - see below):

```
#$CLASS (key) $DUPEKEY
CALA_Testevent dupekey_calatest
Solaris_Disk_Space dupekey_diskspace
Solaris_Su_Failure dupekey_su_failure
```

Example 9-77. An example message map using auxkeys



The auxkeys definition dialog

Auxkeys parameters and their setting in the configuration file

```
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!tecfmtemit;cmdemit;calaproxy;reportemit;snmpemit;smtpemit,completers!report_cpl, remappers!smtpemit_remap,types!map_sev,rules!test_rule,auxkeys!errcode1;location,conf!port;run;targets;completers;remappers;types;rules;auxkeys
errcode1=$CLASS;L;errcode;N
location=$CLASS;L;location;L
```

Example 9-78. An example msgclsfsrv configuration using auxkeys

auxkeys instruction

The auxkeys instruction holds a list of auxkeys used by the message classification server and its subcomponents.

```
auxkeys!<key name>{;<key name>}
```

Figure 9-54. Format of auxkeys instruction

```
auxkeys!errcode1;location
```

Example 9-79. Example auxkeys instruction

Auxkeys configuration line

Each auxkey has to be defined in a configuration line of its own.

```
<key name>=<field>;<text position>{;<field>;<text position>}
```

Figure 9-55. Format of auxkeys configuration line

Further information on the text position parameter can be found in the annex.

```
errcode1=$CLASS;L;errcode;N
location=$CLASS;L;location;L
```

Example 9-80. Examples auxkeys configuration line

The example defines two auxkeys `errcode1` and `location` which can be used within message maps rules maps for duplication or correlation detection.

the msgclsfsrv flowlimiter

The msgclsfsrv flowlimiter is a component of the msgclsfsrv which can be used to control the number of events send to the targets. It detects event storms and stops event forwarding until the event occurance has reached a normal value.

```
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!reportemit,flowlimiters!reportemit;fl_reportemit,conf!run;port;targets;flowlimiters

fl_reportemit=eventquota!10;100;30,eventperiod!300,unblock!30;50;60,blockedevent!fl_event,unblockedevent!fl_event2,logfile!fl_reportemit.fir
fl_event=$PRITYPE=tec;$SECTYPE=CALA_FLOWLIMITER;$CLASS=FLOWLIMITER_BLOC
```

```
K;msg=FLOWLIMITER is blocking some events: $FLOWLIMITER_BLOCKED_INFO
fl_event2=$PRITYPE=tec;$SECTYPE=CALA_FLOWLIMITER;$CLASS=FLOWLIMITER_UNBLOCK;msg=FLOWLIMITER unblocked: $FLOWLIMITER_BLOCKED_INFO
```

Example 9-81. An example msgclsfsrv configuration using auxkeys

flowlimiters instruction

The `flowlimiters` instruction configures the targets to be supervised by the flowlimiter. A flowlimiter entry consists of two entries: the name of the target and a symbolic name of the corresponding flowlimiter configuration. Several entries are separated by semicolons.

```
flowlimiter!<target>;<flowlimiter_name>{;<target>;<flowlimiter_name>}
```

Figure 9-56. Format of flowlimiter instruction

```
flowlimiters!reportemit;fl_reportemit
```

Example 9-82. Example flowlimiter instruction

Note: When using flowlimiters for several targets, one flowlimiter should be defined for each of this targets. Using the same flowlimiter (the same flowlimiter name) for serveral targets is not supported.

flowlimiter configuration line

A flowlimiter configuration line is identified by the flowlimiters name. It contains the configuration of the flowlimiter e.g. the number of events which are allowed to pass in a specified period.

```
<flowlimiter-name>=eventquota!<eventcount/stream>;<eventcount/all>;<secs.>{;<eventcount/stream>;<eventcount/all>;<secs.>},eventperiod!<secs.>,unblock!<eventcount/stream>;<eventcount/all>;<secs.>,blockedevent!<event_name>,unblockedevent!<event_name>,logfile!<logfile>
```

Example 9-83. Format of the flowlimiter configuration line

```
fl_reportemit=eventquota!10;100;30,eventperiod!300,unblock!30;50;60,blockedevent!fl_event,unblockedevent!fl_event2,logfile!fl_reportemit.fir
```

Example 9-84. An example flowlimiter configuration line

`eventquota`

The `eventquota` instruction defines the conditions for detecting an event storm. It takes three arguments:

- the time period for the limits described below
- the maximum number of events for a single stream
- the maximum number of events overall streams

These three arguments are separated by semicolons and may be repeated to define several time periods.

```
eventquota!<secs.>;<eventcount/stream>;<eventcount/all>{;<secs.>;<eventc
ount/stream>;<eventcount/all>}
```

Figure 9-57. Format of `eventquota` instruction

```
eventquota!130;0;100
```

Example 9-85. Examples `eventquota` configuration line

The example defines a time period of 30 seconds and a limits of 10 events for a single datastream and 100 events overall.

If more than 10 events for a single data stream arrive within a period of 30 seconds, the 11th and all following events are blocked by the flowlimiter.

The overall limit is hit, if there have been 100 events sent within the last 30 seconds. The 101st event and all following will be blocked.

Blocked events are written to a logfile configured in the `logfile` instruction. If an event storm has detected, an information event configured by the `blockedevent` instruction is generated.

`eventperiod`

The `eventperiod` statement specifies the period of informational events.

```
eventperiod!<secs.>
```

Figure 9-58. Format of `eventperiod` instruction

```
eventperiod!300
```

Example 9-86. Examples `eventperiod` configuration line

The example eventperiod configuration specifies a period of 300 seconds. As long as any datatype is blockedby the flowlimiter, an informational event is created each 300 seconds.

`unblock`

The `unblock` instruction is the counterpart of the `eventquota` instruction. It defines the conditions to unblock a data stream and resume event delivering. It takes the same arguments as the `eventquota` instruction, but only a single triplet.

- the time period for the limits described below
- the maximum number of events for a single stream
- the maximum number of events overall streams

```
unblock!<secs.>;<eventcount/stream>;<eventcount/all>
```

Figure 9-59. Format of `unblock` instruction

```
unblock!60;30;50
```

Example 9-87. Examples `unblock` configuration line

If a single datastream has been blocked, it is unblocked if within 60 seconds not more than 30 events arrive for this datastream.

If a general block occured (all datastreams are blocked), it is unlocked if not more than 50 events are procssed within 60 seconds.

`blockedevent` and `unblockedevent`

The `blockedevent` and `unblockedevent` statements specify the events the be sent, if an event storm starts or ends. They both just take one argument: the identifier of the event configuration. The event configuration is specified in an own line.

```
blockedevent!<event_name>,unblockedevent!<event_name>
```

Figure 9-60. Format of `blockedevent` and `unblockedevent` instructions

```
blockedevent!fl_event,unblockedevent!fl_event2
fl_event=$PRITYPE=tec;$SECTYPE=CALA_FLOWLIMITER;$CLASS=FLOWLIMITER_BLOCK;msg=FLOWLIMITER is blocking some events: $FLOWLIMITER_BLOCKED_INFO
fl_event2=$PRITYPE=tec;$SECTYPE=CALA_FLOWLIMITER;$CLASS=FLOWLIMITER_UNBLOCK;msg=FLOWLIMITER unblocked: $FLOWLIMITER_BLOCKED_INFO
```

Example 9-88. Examples `blockedevent` and `unblockedevent` configurations

The event definition is configured in an own line, starting with the event name and following the format:

```
<event_name>=<slot_name>;<slot_value>{;<slot_name>;<slot_value>}
```

Figure 9-61. Event definition

The special string `$FLOWLIMITER_BLOCKED_INFO` in a slot value is replaced with additional information about blocked/unblocked data streams.

> **Note:** Since the flow limiter is the last component of msgclsfsrv, the created events do *not* pass any mappers, remappers or completers.

`logfile`

The `logfile` statement specifies the name of the file to receive blocked events. Each event is written to an own line, event format is the same as described above (`blockedevent` and `unblockedevent` statements).

```
logfile!<logfile_name>
```

Figure 9-62. Format of `logfile` instruction

```
logfile!blocked_events.dmp
```

Example 9-89. Examples `logfile` configuration line

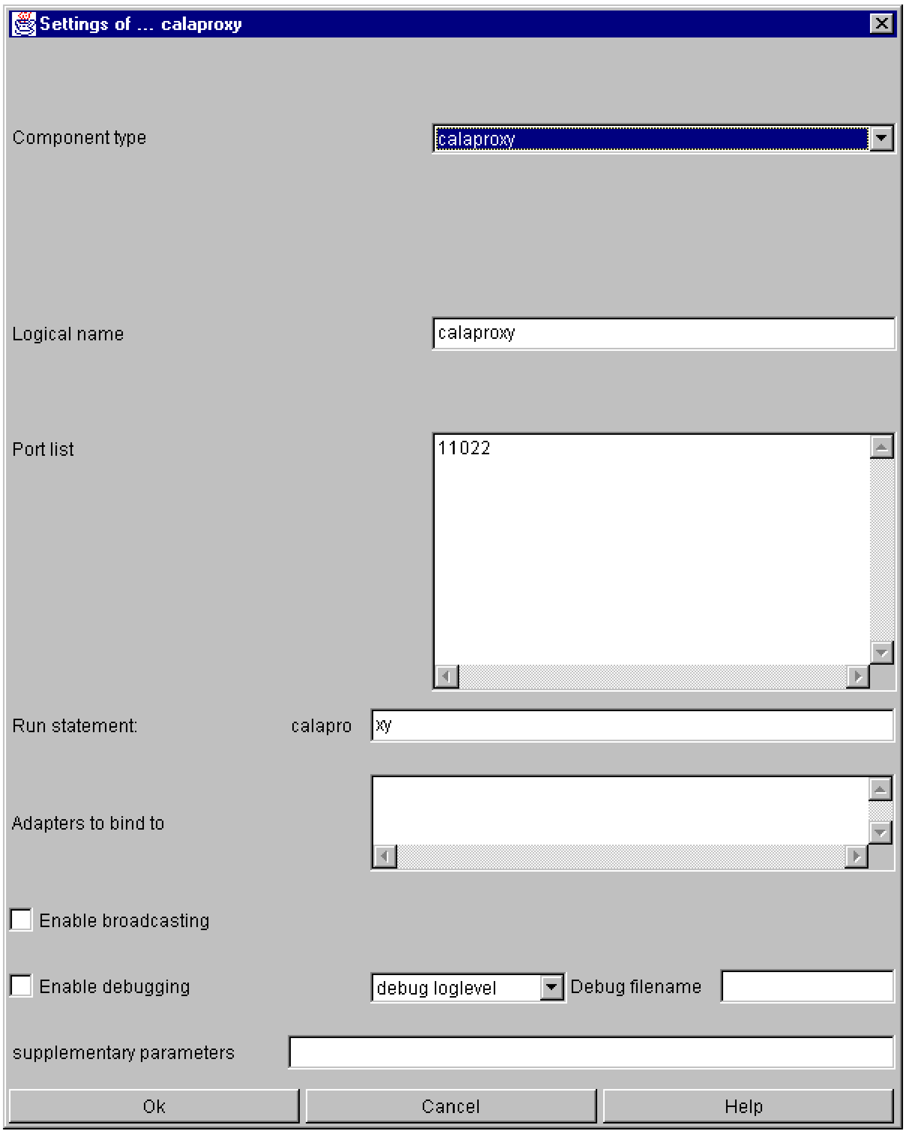
This example defines the events to be written to a file `blocked_event.dmp` within the cala directory.

msgclsfsrv command line parameters

There are currently no additional command line parameters for `msgclsfsrv`.

calaproxy

The `calaproxy` is a simple server process to forward events to one or more targets. It was designed for use in a DMZ (Demilitarized Zone).



calaproxy specific parameters and their setting in the configuration file

This is the configuration line created from the settings window:

```
calaproxy=run!calaproxy -P 11022,port!11022,targets!remote_emit, conf!port;run;targets
```

Example 9-90. An example configuration line for calaproxy

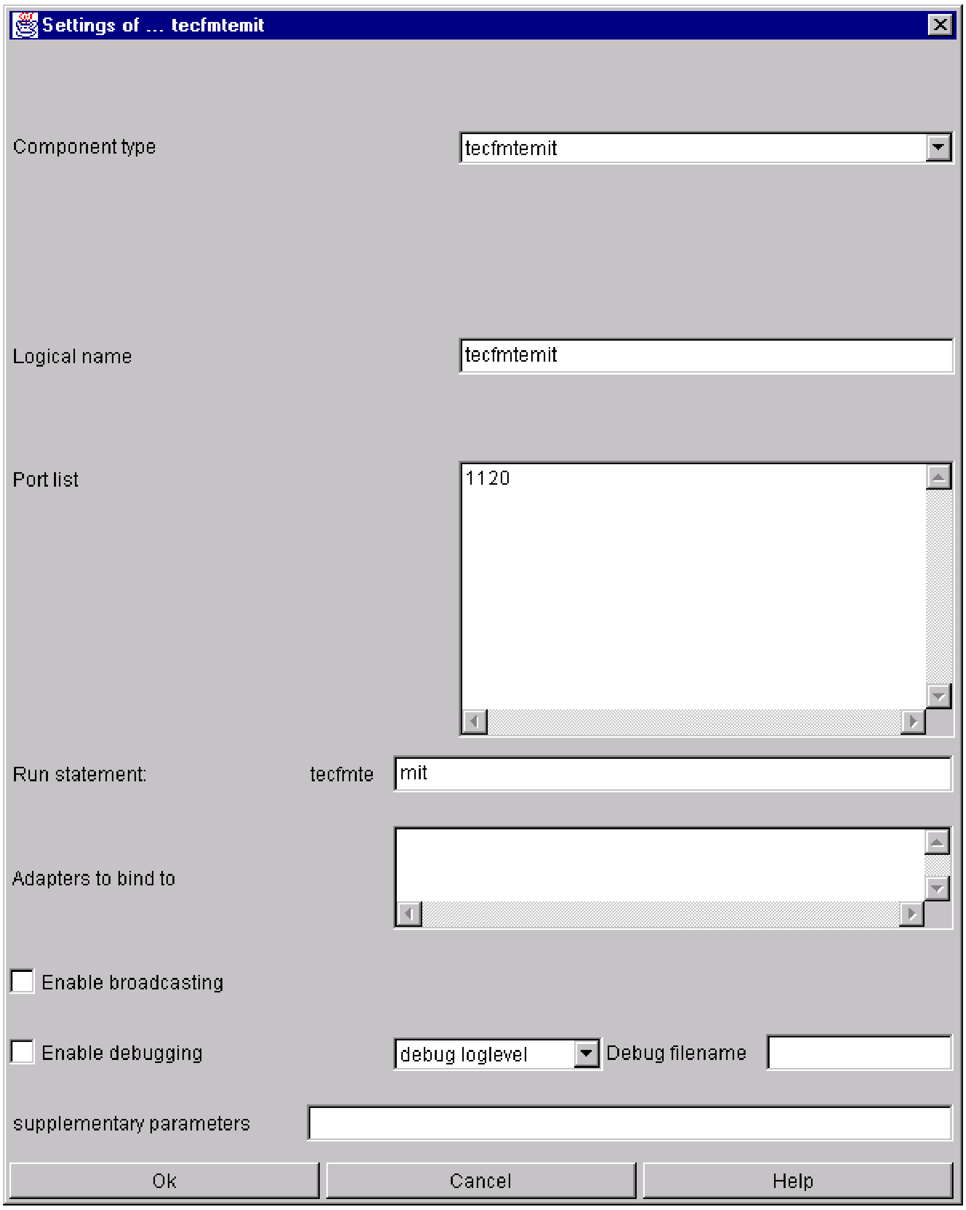
There are no special parameters for `calaproxy`. For description of standard parameters see *Common settings*.

calaproxy command line parameters

There are currently no additional command line parameters available for `calaproxy`.

tecfmtemit

The T/EC format emitter `tecfmtemit` converts events from CALA's FIRs format into a string format which is understood by Tivoli components. The T/EC interface server `tecifcsrv` will send this string to the T/EC.



The `tecfmtemit` settings dialog

tecfmtemit specific parameters and their setting in the configuration file

This is the configuration line created from the settings window:

```
tecfmtemit=run!tecfmtemit -1120,port!1120,targets!tecifcsrv, conf!port;r
un;targets
```

Example 9-91. An example configuration line for `tecfmtemit`

There are no special parameters for `tecfmtemit`, for description of standard parameters see chapter *Common settings*.

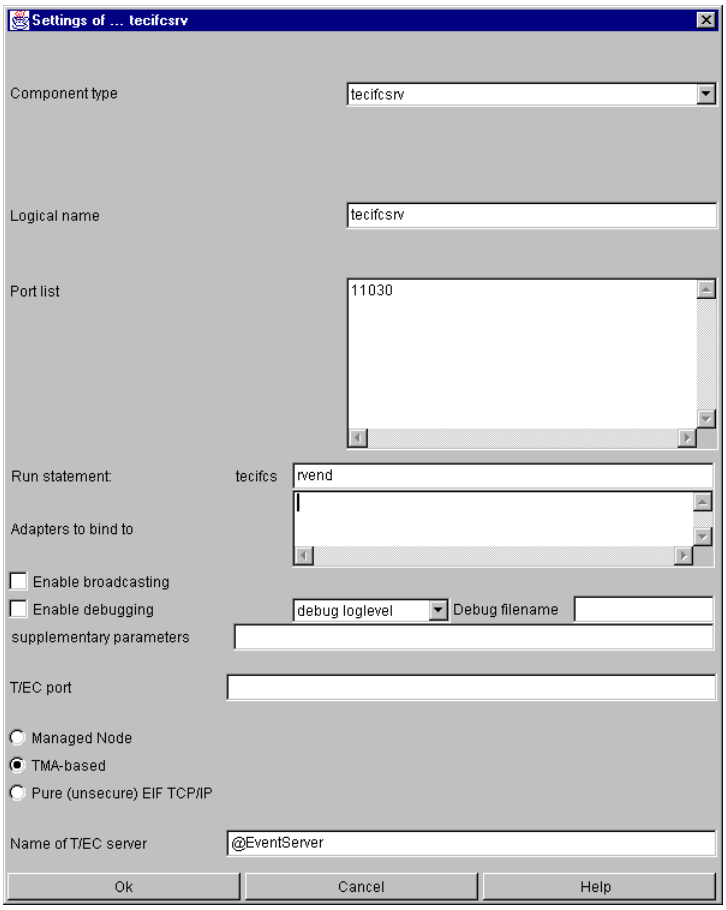
tecfmtemit command line parameters

There are currently no additional command line parameters available for `tecfmemit`.

tecifcsrv

The T/EC interface server sends events which are in string format to the Tivoli enterprise console. FIRs are converted into the string format using the T/EC format emitter `tecfmtemit` (see *tecfmtemit*).

This window defines all parameters for the CALA-T/EC communication component tecifcsrv. Component tecifcsrv exists as a Secure Version (Tivoli ManagedNode communication), an Endpoint-Version (Tivoli TMA communication) and as a purely un-secure TCP/IP Version (EIF communication).



The `tecifcsrv` settings dialog

Note: The run instruction for NT systems must contain parameters -p <T/EC-Port>.Normally, the Default T/EC port on NT systems is set/mapped to 5529 (-p 5529).

tecifcsrv specific parameters and their setting in the configuration file

This is the configuration line created from the settings window:

```
tecifcsrv=run!tecifcsrvend -P 11030 -h @EventServer,port!11030,conf!port
;run
```

Example 9-92. An example configuration line for `tecifcsrc`

There are no special parameters for `tecfmtemit`, for description of standard parameters see chapter *Common settings*.

tecifcsrv command line parameters

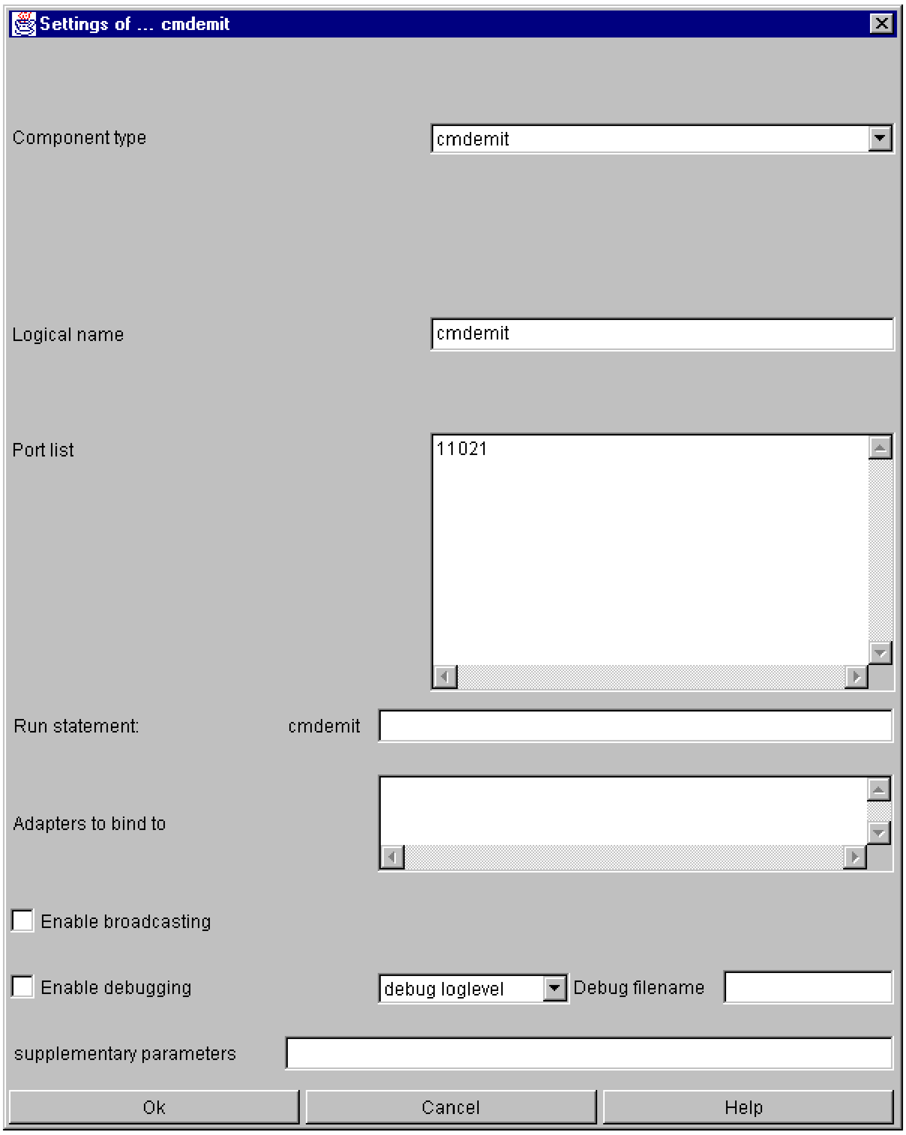
parameter	example	description	default value
-p <port no.>	-p 5529	Specifies the TCP-port the T/EC server is listening on. Attention: This parameter must be given on NT system, on Unix it is optional.	not set
-h <hostname>	-h tiv1.cenit.de	IP-Address or name of the host running the T/EC server.	the tcp/ip hostname
-c	-c	Set communications mode to connectionless communication. (Should be used if only few events are send to T/EC, for details see the Tivoli documentation for Event Integration Facility.)	useconnection oriented communication
-C	-C	Switches usages of one way and connectionless connections. (For details see the Tivoli documentation for Event Integration Facility)	not set

parameter	example	description	default value
`-l <filename>`	-l if_cala.conf	Sets the name of the Tivoli T/EC adapter configuration file. (For details on the T/EC adapter configuration files see the Tivoli documentation for Event Integration Facility.)	tecad_cala.conf

cmdemit

The component cmdemit is a task engine. It is used to execute various programs (binary programs, scripts, batch programs, etc.). Any number of parameters can be issued.

The programs to be executed are taken from some special slots within the received event (see below). These slots can be set using one of the filter components (`tecfmtfilt`, `v2fmtfilt`) or the message classification server.



The `cmdemit` settings dialog

cmdemit specific parameters and their setting in the configuration file

This is the configuration line created from the settings window:

```
cmdemit=run!cmdemit -P 11021,port!11021,conf!port;run
```

Example 9-93. An example configuration line for cmdemit

There are no special parameters for cmdemit. For description of standard parameters see *Common settings*.

cmdemit command line parameters

parameter	example	description	default value
-m<no.>	-m1	Sets the maximum number of child processes allowed to run parallel. To get all child processes called serially, specify 1.	10
-k<secs.>	-k60	Specifies the timeout (in seconds) for child processes to terminate. If a child process doesn t terminate within the given time, it is killed.	60

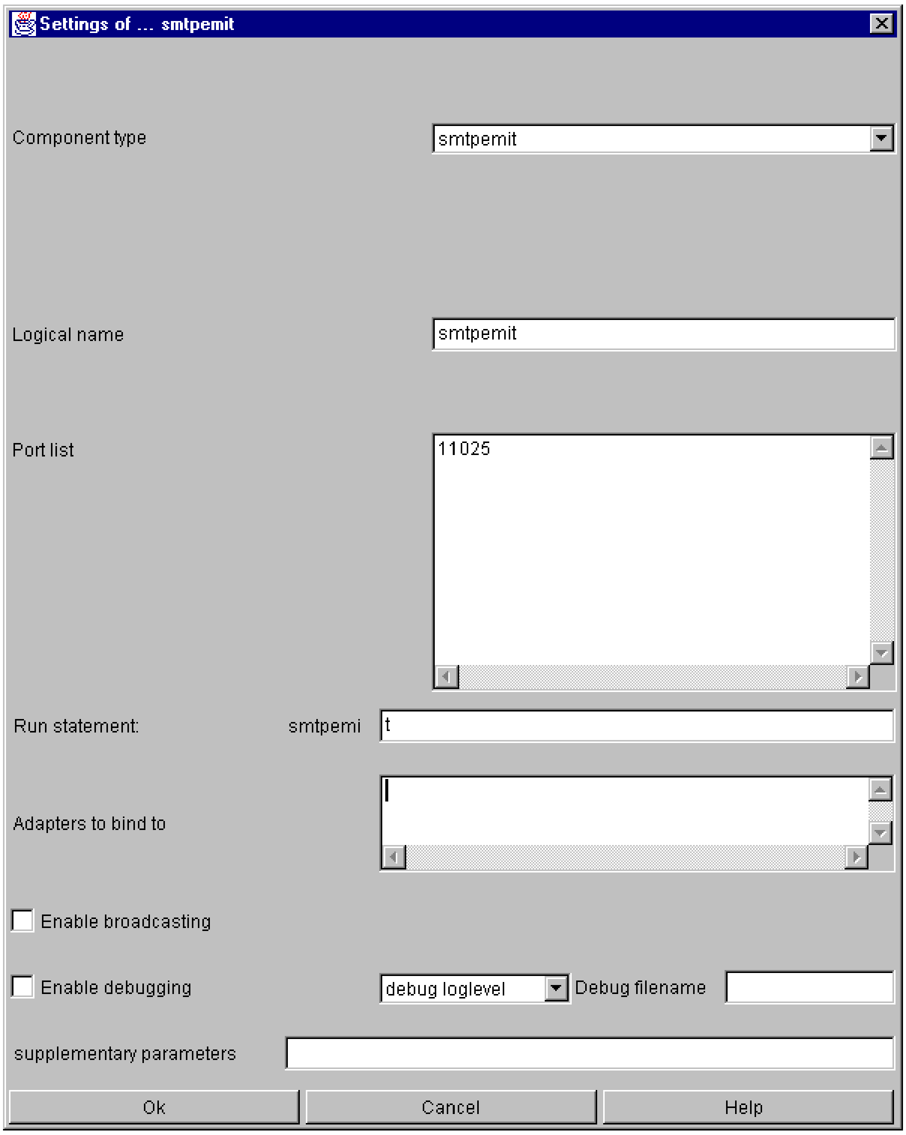
cmdemit input events

Events sent to the command emitter need some special fields to be set.

field name	description
$COMMAND	Name of binary or script to be executed.
$NUMARGS	Number of arguments specified within this FIR.
$ARG<no.>	Argument <no.>, where <no.> is the argument number. The first argument is put in a slot $ARG0, the second one in $ARG1 and so on.

smtpemit

This component was created for enabling CALA to send emails from incoming events.



The settings of smtpemitdialog

smtpemit specific parameters and their setting in the configuration file

This is the configuration line created from the settings window:

```
001  smtpemit=run!smtpemit -P 11025,port!11025,conf!port;run
```

Example 9-94. An example configration line for `smtpemit`

There are no special parameters for `smtpemit`. For description of standard parameters see *Common settings*.

smtpemit command line parameters

There are currently no additional command line parameters available for the mail emitter.

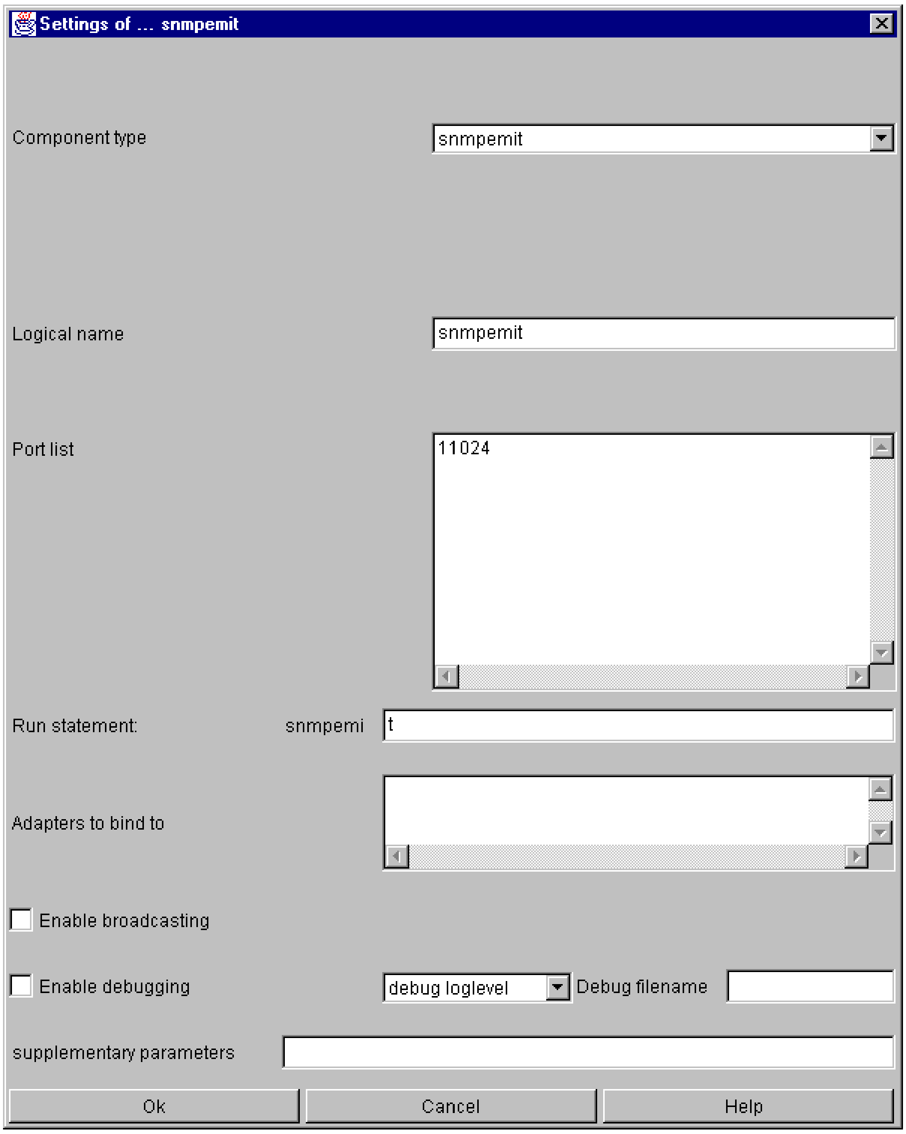
smtpemit input events

Events sent to the mail emitter need some special fields to be set.

field name	**description**
`SERVER`	hostname or IP address of host running any SMTP server like sendmail.
`PORT`	(optional) port no. on the target system (default: 25 (SMTP))
`CLIENT`	(optional) Hostname or IP address of sending system (default: local address)
`SENDER`	sender's email address
`SUBJECT`	email subject
`RECIPIENT`	email address to deliver email to (several adresses separated by comma are supported)
`MSGBODY`	message text
`TIMEOUTSEC`	(optional) timeout for TCP/IP requests to SMTP-server (seconds) (default: 30)
`TIMEOUTUSEC`	(optional) timeout for TCP/IP requests to SMTP-server (milliseconds) (default: 30)

snmpemit

The SNMP trap emitter `snmpemit` was designed for creating SNMP traps from CALA events (FIRs).



The settings of snmpemit dialog

snmpemit specific parameters and their setting in the configuration file

This is the configuration line created from the settings window:

```
snmpemit=run!snmpemit -P 11024,port!11024,conf!port;run
```

Example 9-95. An example configuration line for `snmpemit`

There are no special parameters for `snmpemit`. For description of standard parameters see *Common settings*.

snmpemit command line parameters

parameter	example	description	default value
-a	-a	replaces all characters >= ascii 126 with underscores (some snmp consoles can handle such characters)	not set
-w	-w	replaces all whitespace characters (tabs, linefeed, newline etc.) with space characters (some snmp consoles can handle such characters)	not set

snmpemit input events

Events sent to the snmp emitter need some special fields to be set.

field name	description
`SNMP_VERSION`	(optional) valid values are SNMPv1, SNMPv2c, SNMPv2c_INFORM, SNMPv3 and SNMPv3_INFORM (default: SNMPv1)
`TARGIP`	Hostname or IP address of host to receive SNMP trap
`TARPORT`	(optional) port no. on the target system (default: 162)
`TRANSPORT_PROTOCOL`	(optional) the ip transport protocol, valid values are tcp and udp (default: udp)
`ORIGIP`	(optional) Hostname or ip address of sending system (default: local address)
`NUMVARS`	number of variables attach to this trap
`OID<no.>`	the object id of variable no. <no.> (mostly a subtree of SENDOID)
`TYPE<no.>`	the type of variable no. <no.> which must be either STRING or NUMERIC.
`VALUE<no.>`	the value of variable no. <no.>

`<no.>` specifies a number between `1` and `$NUMVARS`. For each variable the three fields `OID`, `TYPE` and `VALUE` followed by the variables number have been defined.

Depending on the used SNMP version there are additional fields needed:

SNMPv1

field name	description
`SENDOID`	(optional) the object id of the sending system (default: .1.3.6.1.4.1.8235 iso.org.dod.internet.private.enterprises.cenit)
`COMMUNITY`	the SNMP community
`TRAPTYPE`	(optional) the type of this trap
`SPECTYPE`	(optional) the specific type of this trap

SNMPv2c

field name	description
`COMMUNITY`	the SNMP community

SNMPv3

field name	description
`SECURITY_LEVEL`	the security level, valid values are noAuthNoPriv, authNoPriv and authPriv
`AUTH_ENGINE_ID`	the id of autoritative security engine (hexadecimal no. starting with 0x)
`CONTEXT_ENGINE_ID`	(optional) the id of the context engine (default: same as AUTH_ENGINE_ID)
`AUTH_USER`	the security name for SNMPv3 authentification
`AUTH_PASSPHRASE`	the authentification passphrase
`AUTH_PROTO`	(optional) the authentification protocol valid values are MD5 and SHA (default: MD5)
`PRIV_PASSPHRASE`	the privacy passphrase
`PRIV_PROTO`	(optional) the privacy protocol, the only valid value is DES (default: DES)
`CONTEXT_NAME`	(optional) the destination context name (default: empty)
`ENGINE_BOOTS`	(optiona) engine boots count (default: 1)
`ENGINE_TIME`	(optional) the engine time (default: current time)

Depending on the selected security level, the fields beginning with AUTH_ and PRIV_ are needed or not.

Hint for SNMPv2c and SNMPv3 users

When using SNMPv2c and SNMPv3 the first two variables are defined to have special meanings as described below. (For details refer to RFC 2576.)

The first variable is called sysUpTime and contains the system uptime, it is from type `TIMETICKS` and has the OID `1.3.6.1.2.1.1.3.0`. If `VALUE` is set to 0, the `snmpemit` sets it to the current time..

```
OID1=1.3.6.1.2.1.1.3.0
TYPE1=TIMETICKS
VALUE1=0
```

Example 9-96. Setting `sysUpTime` to current time

The second variable, the `snmpTrapOID` contains the traps oid which is the `enterprise OID` followed by `.0`. and the trap type or one of the following pre-defined values:

- 1.3.6.1.6.3.1.1.5.1 (coldStart)
- 1.3.6.1.6.3.1.1.5.2 (warmStart)
- 1.3.6.1.6.3.1.1.5.3 (linkDown)
- 1.3.6.1.6.3.1.1.5.4 (linkUp)
- 1.3.6.1.6.3.1.1.5.5 (authenticationFailure)
- 1.3.6.1.6.3.1.1.5.6 (egpNeighborLoss)

The snmpTrapOID variable is from type OBJID.

```
OID2=.1.3.6.1.4.1.8235.0.1
TYPE2=OBJID
VALUE2=0
```

Example 9-97. Setting the `snmpTrapOID` variable

mysqlemit

The MySQL emitter is a component for writing events into a MySQL database. The CALAGUI doesn't currently support `mysqlemit`, so it has to be configured manually or using the CALA Configurator (see *CALA Configurator Basics* and *Details*).

mysqlemit specific parameters and their setting in the configuration file

The `mysqlemit` supports several parameters for database and eveet configuration, this is a sample configuration line:

```
mysqlemit=run!mysqlemit P 22012 -Hmysqlserv,port!22012,database!cala,dbu
ser!calaweb;0e0e1908150e1300,tableconf!db_id;default;%s_new;%s_history,
ok_status!0,dbfields!$HOSTNAME;hostname;k+;$area;area;k+;$info;info;k*;
$CTIME;date;dao;msg;msg;ma;status;status;s;$ORIGIN;adapterhost;;$LOGFIL
ENAME;source;;value;value;a;$CTIME;since;d;$mode;mode;a,logfile_type!$m
ode;2,conf!run;port;database;dbuser;ok_status;dbfields;tableconf;logfil
e_type
```

Example 9-98. An example `mysqlemit` configuration line

database instruction

This defines the database to be used

```
database!<db-name>
```

Figure 9-63. Format of <code>database</code> instruction

```
database!cala
```

Example 9-99. Example <code>database</code> instruction

dbuser instruction

The dbuser instruction sets the database user and password. The password has to be given encrypted, use the Monitoring Manager to encrypt the password. (Use the password dialog from the Tools menu, for ruther information refer to the CalaMoMa User's Guide.)

```
dbuser!<user>;<password>
```

Figure 9-64. Format of the dbuser instruction

```
dbuser!calaweb;0e0e1908150e1300
```

Example 9-100. Example the dbuser instruction

tableconf instruction

This configures the table(s) to be used.

The `mysqlemit` can be configured to use different database tables depending on event slots. The first parameter of this instruction defines the FIR field to be used for the table name assembling.

The second parameter is a default value for events which don t have set that field.

The third parameter is a mask string to be used to create the table name. It may contain a `%s` which would be replaced by the value taken from the FIR field. If no `%s` is used within this string, the first two fields of this instruction are ignored.

The last parameter is a mask for another table the history table. The `mysqlemit` uses two tables: one table for current events, and another, a history table, for out-of-date events. Monitor events are first written into the current events table. If a new value from the same monitor is received, the old event is moved to the history table and the current event table receives the new one.

```
tableconf!<fir-field>;<default>;<mask new-table>;<mask history table>
```

Figure 9-65. Format of tableconf instruction

```
tableconf!db_id;default;%s_new;%s_history
```

Example 9-101. Example tableconfinstruction

ok_status instruction

This instruction tells the `mysqlemit` which status value means that everything is ok.

This is used when overwriting events in the current events table and moving them to the history table. If the status is `ok`, an event is only moved to the history table, when its status switched to `not ok`, no matter if any message field changes or not. (That is because status changes aren’t from historical interest as long as the main status is `ok`.)

If an event is in the `not ok` status, it s also moved if any of the message fields change, because this could be needed for problem diagnostics.

```
ok_status!<value>
```

Figure 9-66. Format of `db_status` instruction

```
001  ok_status!0
```

Example 9-102. Example `db_status` instruction

`dbfields` instruction

This instruction maps event fields to database fields and classificates them.

Each field mapping consists of three parameters:

- the FIR field
- the corresponding database field (row)
- the field classification flags

The following table shows the available field classification flags an their meanings. Each field may have set none, one or several flags. Be aware, that the commas have to be configured, even if there is no flag to set for a field.

flag	**name**	**description**
`k`	is keyfield	This is a database keyfield, used to identify corresponding events.
`s`	is statusfield	This is a status field, see description of the ok_status instruction for special handling of status fields.
`m`	is message field	This is a message field, see description of the ok_status instruction for special handling of message fields.
`d`	is date field	This field contains a data, which needs special handling when writing it to the database.
`l`	lower case field	This field's name will allways be set to lower case spelling.
`a`	update always	This field should always be updated in the database, even if neither status nor message fields changed.
`o`	overlay	If an event is moved to the history database, the overlay fields from the new (replacing) event is copied into the old event. (Useful to save the end-time of a status.)
`S`	update if status changes	Specifies the field only to be updated if the status changes.
`O`	update if status changes to `ok`	Specifies the field only to be updated, if the status switches to `ok` (overrides `S`).
`+`	remove dummy keyfield	FSM internal: this is a keyfield for dummy entries
`*`	remove dummy empty	FSM internal: this is empty for dummy entries

```
001
```

```
dbfields!<fir-slot>,<db-field>,<flags>{,<fir-slot>,<db-field>,<flags>}
```

Figure 9-67. Format:

```
dbfields!$HOSTNAME;hostname;k+;$area;area;k+;$info;info;k*;$CTIME;date;
dao;msg;msg;ma;status;status;s;$ORIGIN;adapterhost;;$LOGFILENAME;source
;;value;value;a;$CTIME;since;d;$mode;mode;a
```

Example 9-103. Example:

mysqlemit command line parameters

parameter	example	description	default value
-H <hostname>	-H mysqlserv	sets the database server	localhost
-L <timeout>	-L 30	sets the timeout for SQL operations (in seconds)	infinite

jdbcemit

The jdbc emitter implements a generic database emitter component similar to `mysqlemit`. It uses javas' jdbc interface to access the database and can therefore be used to access any database for which a jdbc driver is available.

jdbcemit specific parameters and their setting in the configuration file

The configuration of the `jdbcemit` is very similar to the configuration of `mysqlemit`.

These are few differences:

- Additional command line parameters for the java virtual machine (e.g. classpath settings) can be passed before the component parameters are given. This parameters must at least add the `CtkDB.jar` file to the classpath. End the java parameters with two dashes.
- The database name consists of three parts:
  - the `jdbcemit` java classname (`de.cenit.eb.sm.ctk.db.DefaultCTKDatabaseDriver` by default)
  - the jdcb driver class
  - the jdbc database URL

  These parts are separated by colons.

```
de.cenit.eb.sm.ctk.db.DefaultCTKDatabaseDriver:com.mysql.jdbc.Driver:jdbc:mysql://localhost/cala
```

Example 9-104. An example database string for `jdbcread`

This example uses mysqls' jdbc driver to connect to the database. The jdbc driver class is `com.mysql.jdbc.Driver`, the database url is `jdbc:mysql://localhost/cala`. The java interface of jdbcemit is implemented in the `de.cenit.eb.sm.ctk.db.DefaultCTKDatabaseDriver` class.

```
jdbcemit=run!jdbcemit -Djava.ext.dirs=../tools/de.cenit:../tools/com.mysql -- -SR -P 23848 -Hlocalhost,port!23848,database!de.cenit.eb.sm.ctk.db.DefaultCTKDatabaseDriver:com.mysql.jdbc.Driver:jdbc:mysql://localhost/cala,dbuser!webtpladmin;00001204190d081409081e00,tableconf!customer;filenet;%s_new;%s_history,ok_status!0,dbfields!$HOSTNAME;hostname;k+;$area;area;k+;$info;info;k*;$CTIME;date;dao;msg;msg;ma;status;status;s;$ORIGIN;adapterhost;;$LOGFILENAME;source;;error_id;error_id;a;$unit;unit;;value;value;a;$CTIME;since;d;$mode;mode;a,logfile_type!$mode;2,conf!run;port;database;dbuser;ok_status;dbfields;tableconf;logfile_type
```

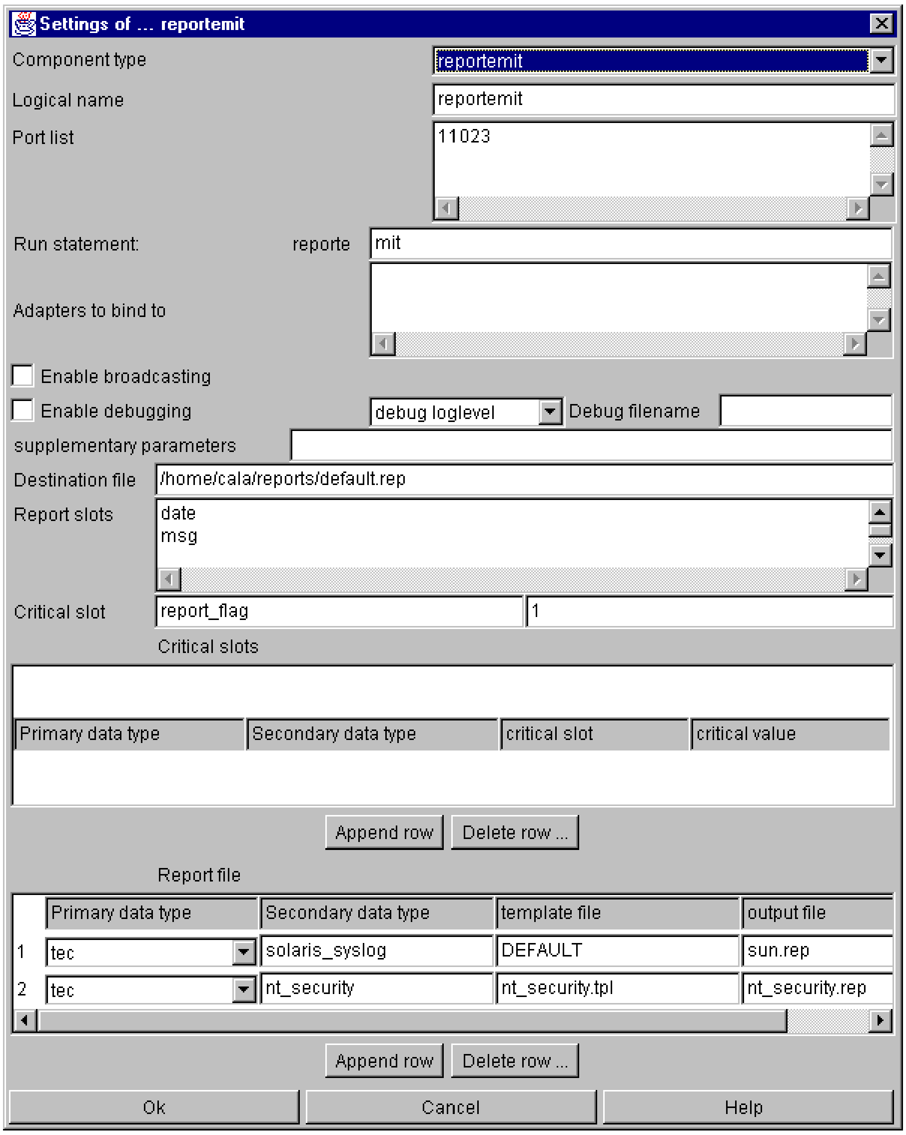
Example 9-105. An example `jdbread` configuration using a mysql connector for connection to the local database `fndsdb`

The jdbc emitter needs a java 1.4 or higher virtual machine to be in the library path.

For further configuration information refer to section *mysqlemit*.

reportemit

The reportemit component is used for creating reports from received events.



The settings of reportemit dialog

reportemit specific parameters and their setting in the configuration file

The configuration line for the report emitter looks like this

```
reportemit=run!reportemit -P 11023,port!11023,dest_file!/home/cala/repor
ts/default.rep,report_slots!date;msg,critical_slot!report_flag;1,report
_file!tec;solaris_syslog;DEFAULT;sun.rep;tec;nt_security;nt_security.tp
l;nt_security.rep,conf!port;run;dest_file;report_slots;critical_slot;re
```

```
...  port_file
```

Example 9-106. An example configuration line for `reportemit`

dest_file instruction

This defines a default report file each data stream not given in the template table will be reported to this file.This file is written in a format similar to the output of the Tivoli tool `wtdumprl`. The destination file is taken from the text field Destination file

```
001  dest_file!<filename>[:<encoding>]
```

Figure 9-68. Format of dest_file instruction

```
001  dest_file!/home/myuser/reports/default.rep
```

Example 9-107. Example dest_file instruction

The example tells `reportemit` to write its events to `/home/myuser/reports/default.rep` unless the template table specifies another file.

The name of the output file may contain various `%` expressions for date and time. (A new file is created if one parameter changes.) The following expressions are possible:

% expression	description
`%a`	name of the day of the week (abbreviation, 3 letters)
`%A`	complete name of the day of the week
`%b`	name of the month (3 letters)
`%B`	complete name of the month
`%d`	day in the month
`%H`	hour (00-23)
`%I`	hour (00-12)
`%j`	day of the year
`%m`	month as a number
`%M`	minute
`%U`	week in the calendar year (00-53), Sunday being the first day in each week
`%w`	day of the week as a number (0=Sunday)
`%W`	week in the calendar year (00-53), Monday as the first day in each week
`%y`	year, two-digit
`%Y`	year, four-digit
`%%`	The % character

For a list of supported encoding refer to *Supported character sets*.

```
dest_file!/home/cala/reports/report_%Y%m%d.rep:UTF-16
```

Example 9-108. Example dest_file instruction using % expressions

This example will create a new report file each day, which will be named report_<4 digit year><month><day>. E.g. on May 8th 2001 incoming events would be reported to the file report_20010508.rep.

The file will be written UTF-16 encoded.

report_slots instruction

The report_slots instruction is optional. If set, it specifies which FIR fields should be reported. If the report_slots instruction is not set, all slots will be reported. The report slots are taken from the text area Report slots.

```
report_slots!<field name>{;<field name>}
```

Figure 9-69. Format of report_slots instruction

```
report_slots!date;msg
```

Example 9-109. Example report_slots instruction

critical_slot instruction

To specify whether a FIR should be reported by reportemit, there s the possibility to configure a critical slot. If both field name and value are given, the reportemit reports only FIRs having the specified value within that field, if only a field name is given, the FIR is reported if the field exists.

The critical_slot instruction is optional. If it is not given, all received FIRs are reported.The values of critical_slot can be given in the two text fields Critical slot.

```
critical_slot!<field name>[;<field value>]
```

Figure 9-70. Format of critical_slot instruction

```
critical_slot!report_flag;1
```

Example 9-110. Example critical_slot instruction

The example specifies to report only events which have the field report_flag set to 1.

critical_slots instruction

The critical_slots instruction is an extension of the critical_slot instruction. It defines different critical slots for one or more data types.

```
critical_slots!<primary data type>;<secondary data type>;<field name>;<field value>{;<primary data type>;<secondary data type>;<field name>;<field value>}
```

Figure 9-71. Format of critical_slots instruction

```
critical_slots!tec;solaris_syslog;severity;FATAL
```

Example 9-111. Example critical_slots instruction

There are some field values with special meanings:

*

slot exists or doesn’t exist (disables the critical slot for this data type)

+

slot exists

-

slot doesn’t exists

If there is no critical_slots configuration for any data type, the critical_slot configuration is used as default configuration.

report_file instruction

For special reports, there is the possibility to use template files for each data stream (given by primary and secondary data type). The data for template file handling is given in the template table at the bottom of the window.

```
report_file!<primary type>;<secondary type>;<template filename>[:<encoding>];<report filename>{;<primary type>;<secondary type>;<template filename>;<report filename>}
```

Figure 9-72. Format of report_file instruction

```
report_file!tec;solaris_syslog;DEFAULT;sun.rep;tec;nt_security;nt_security.tpl;nt_security.rep
```

Example 9-112. Example report_file instruction

If `DEFAULT` is given as template filename, the default output format (like Tivoli `wtdumprl`) is used. If no type within the template table matches, the default report file is used (see instruction dest_file).

Template files

A template file is a text file containing text and slot tags. A slot tag is the field name, enclosed in < and >.

```
This report for class <$CLASS> was created at <date>.
```

Example 9-113. A sample report template:

```
This report for class CALA_Testevent was created at Mon Mar 12 10:16:01
2001.
```

Example 9-114. An example result for the above template

reportemit command line parameters

parameter	example	description	default value
-O	-OUTF-16	Specifies the default character set to be used for writing report files.	UTF-8

parameter	example	description	default value
-L[type][replacestring]	-LW	Specifies the handling of newline characters, supported values are: W Replace line breaks in slot values with Windows style newline characters. w Same as *W* but used for all line breaks in the reportfiles. U Replace line breaks in slot values with unix style newline characters. u Same as *U* but used for all line breaks in the reportfiles. R Replace line breaks in slot values with a space character. If R is followed by additional characters, these characters are used to replace newline characters. (E.g. -LR*NL* would replace all line breaks with the string *NL*) r Same as *R* but used for all line breaks in the reportfiles.	unset (don't replace any newline chars)

javasrv

`javasrv` is a generic CALA component to start CALA components implemented in java. It therefor needs a java 1.4 virtual machine in the library path. java servers can be FIR generating or FIR processing or both, depending on the implementation.

javasrv specific parameters and their setting in the configuration file

This is a sample configuration line for javasrv running the pchread component:

```
pchread=run!javasrv de/cenit/eb/sm/fnpch/calamanager/FnManagerCalaSrv -D
java.ext.dirs=../tools/de.cenit:../tools/com.filenet:../tools/org.apach
e --11024,port!11024,targets!msgclsfsrv,xmlconf!javasrv_pchread.xml;jav
asrv_pchread.xsd,conf!run;port;targets;xmlconf
```

Example 9-115. An example configuration line for `javasrv`

xmlconf instruction

This value of this parameter is passed to the java component. In most cases it would contain the name of an xml configuration file and the components xsd file, but the sense of this parameter may vary in future components.

See the describtion of the java component for details.

```
xmlconf!<filename of xml file>;<filename of xsd file>
```

Figure 9-73. Format:

```
xmlconf!javasrv_pchread.xml;javasrv_pchread.xsd
```

Example 9-116. Example:

javasrv command line parameters

The commandline parameters of `javasrv` differ from the commandline parameters of other CALA components, its commandline uses the following format:

```
javasrv <Classname of java component> <java vm parameters> -- <cala parameters> <-H hostname>
```

Figure 9-74. Format of `javasrv` commandline

parameter	example	description
`<Classname>`	de/cenit/eb/sm/fnpch/calamanager/FnManagerCalaSrv	specifies the main class of the java component
`<java vm parameters>`	-Djava.ext.dirs=../tools/de.cenit	parameters to be passed to the java virtual machine
`--`		Signals the end of java vm parameters. The following parameters are interpreted as cala parameters.
`<cala parameters>`		see *Common settings* for a list of general parameters
`-H hostname`	-H myserver	sets the hostname

javasrv/pchread

`pchread` is one implementation of a java component. It is used to read events from a FileNet Listener and is an FIR generating component.

Classname

`de/cenit/eb/sm/fnpch/calamanager/FnManagerCalaSrv`

Jarfile

`fn_pch_calamanager.jar`

Dependencies

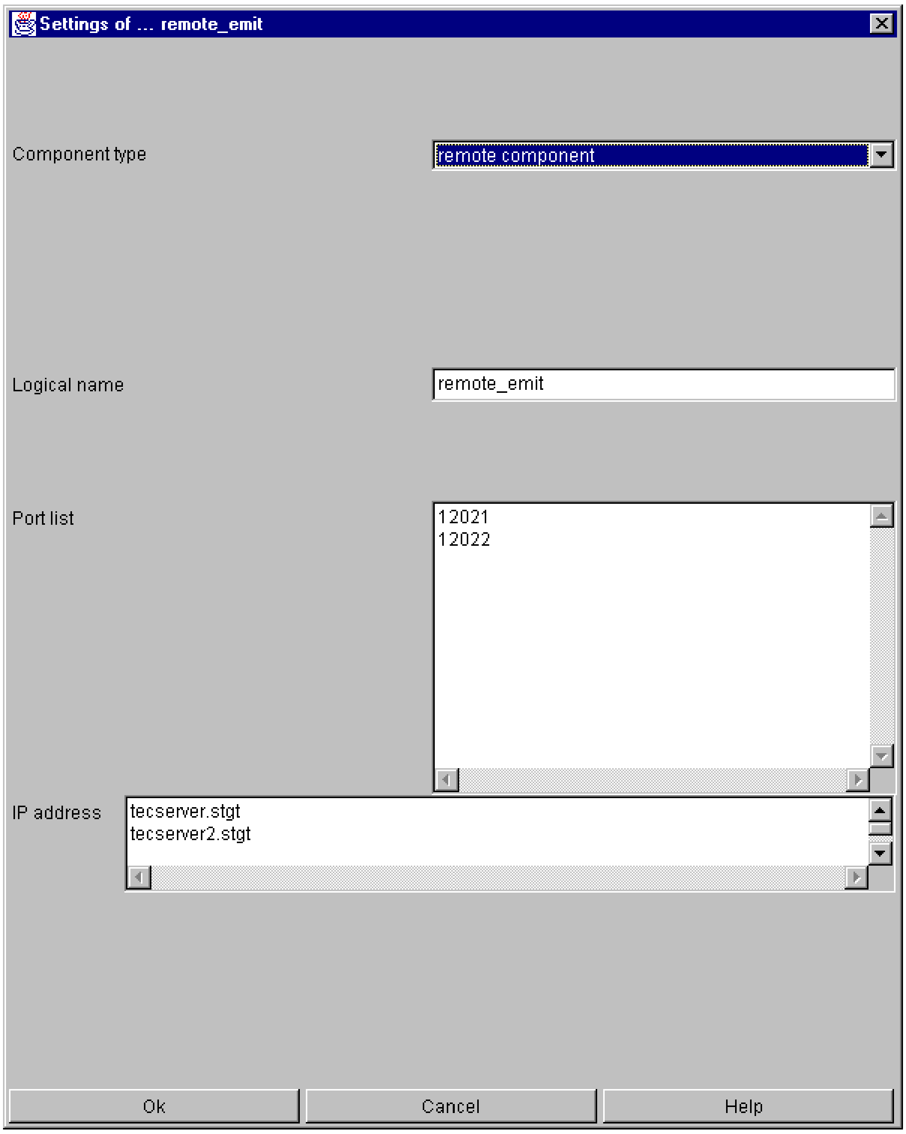
fn_pch_utils.jar, cala_jni.jar, utils.jar, pwdcrypt.jar, mgrlib.jar[1], log4j.jar

See *Pchread XML Configuration* for a description of the `pchread` xml file.

1. Depending on the installed FileNet software and version, the filename of the mgrlib.jar may be extended with a version number, e.g. mgrlib-3.5.jar.

remote component

This virtual component is used in every client configuration as a place holder for definition of parameters for the server(s) to which the last configured client component is to sent the architecture.



The settings of remote_component dialog

remote component specific parameters and their setting in the configuration file

As the remote component is only virtual and represents any unknown remote component, there are no additional parameters. The settings window above created the following configuration line:

```
remote_emit=ip!tecserver.stgt.cenit.de;tecserver2.stgt.cenit.de, port!12021;12022,conf!port;ip
```

Example 9-117. An example configuration line for a remote component

Chapter 10. Security

This chapters deals with security problems and shows how CALA can be configured to prevent unauthorized persons from reading events sent over un-secure connections and to avoid invaders to affect CALA's functionality.

For further information about CALA security mechanism see also the CALA Security White paper shipped with FileNet System Monitor.

Encrypted Communication

To protect CALA processes from being abused by crackers, there are additional encryption features which should be used on un-secure connections.

The CALA processes exchange data over the TCP/IP protocol, as local well as remote. For local communication the loopback device is used, so there is no traffic on the network.

For remote connection, there are 4 security levels available:

0

no encryption (by default is used for internal connections over loopback device only)

1

Vigenere encryption (default)[1]

2

RSA encryption (safe but slow)[2]

3

one-time-pad encryption

The encryption keys can be generated by the tool crypttool which is part of the CALA distribution. It can generate different keys for client and servers, the key length is freely definable.

The one-time-pad encryption algorithm

The one-time-pad encryption (encryption level 3) uses a combination of the Vigenere encryption and the RSA encryption algorithm.

At connection establishment, the client generates some random data and sends it to the server, using RSA encryption. With this initial data, both processes are able to calculate a large number of temporary encryption keys.

The following communication packages are transmitted Vigenere encrypted, each package using a number of the temporary keys (depending on its size). After a key is used, it is discarded. If no more keys are available, a new initialization package is send to the server.

1. The Vigenere algorithm is a simple encryption algorithm . It is explained at several internet sites, e.g. http://raphael.math.uic.edu/~jeremy/crypt/vignere.html (http://raphael.math.uic.edu/~jeremy/crypt/vignere.html)
2. For more information about RSA see http://www.rsa.com.

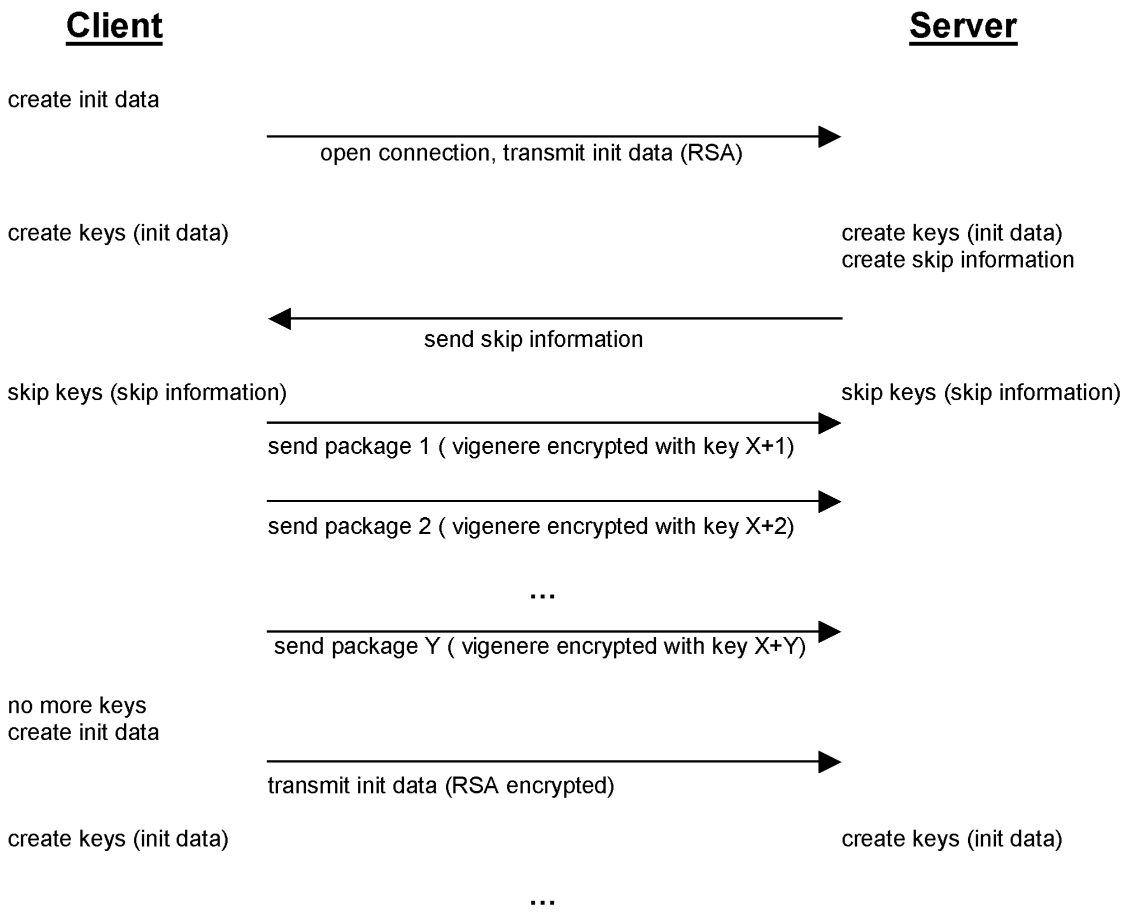


Figure 10-1. The one-time-pad communication schema

To ensure that a cracker cannot attack the system by replaying old TCP/IP packages, the server sends a random skip information after creating the keys. This tells the client not to start with the first generated key, but with a later one.

When replaying old TCP/IP packages, this will cause the server to be unable to decrypt the received packages, because the client uses the wrong (old) skip information.

Configuring encryption

Encryption configuration is done by setting command line arguments for the CALA components. In CALAGUI these arguments can be added in the text field supplementary parameters

The following parameters are supported:

parameter	example	description	default value

parameter	example	description	default value
`-YK<keyfile>`	-YKcala_key	This parameter specifies the name of the key file containing the encryption keys to use. If the key file name contains a , this is replaced by the hostname or IP-address of the machine connected to. In this way you can have different key files for different connections. By default the key is searched in `.cala_key` within the CALA directory. If no key file is found, a standard key is used. This parameter has to be set on the client side as well as on the server side.	-Y.cala_key
`-YL<level>`	-YL1	This is a parameter to be set on the client side. It specifies which encryption level is to be used for outgoing connections (see list of possible connection levels above). By default a encryption level of 1 is used for remote connections. (Local connection are always using encryption level 0.) If using an unsecure connection, a encryption level of 3 is recommended.	-YL1

parameter	example	description	default value
-YA<level>	-YA1	This server-side parameter controls the minimum encryption level a client has to use when connecting to the server. Clients using a lower connection level are refused. The minimum encryption level for accepting connections is 1 by default, for servers with un-secure connection, a level of 3 is recommended.	-YA1
-YE	-YE	Enables the creation of encryption error events (see *CALA created events*)	not set

On critical connection which may be attacked by hackers the usage of encryption level 3 is recommended.

The crypttool

Encryption keys can be generated using the tool `crypttool`.This is the usage screen (call **crypttool -?** to get this):

```
        *******************************************************************
        **                                                               **
        **      crpt_tool is part of the CENIT Advanced Logfile Adapter  **
        **                                                               **
        **  version: 2.02-035   - generation date: Oct 13 2005 17:04:24  **
        **                                                               **
        **               (c) 1999-2005 CENIT AG Systemhaus               **
        **                                                               **
        *******************************************************************

unknown paramter: "-?"

usage: ./crypttool.exe [-c <filename>] [-s <filename>] [-a <filename>] [-v <length>] [-r <length>]

        -c, --client-key: filename of client key-file to write
        -s, --server-key: filename of server key-file to write
        -a, --all: filename of complete key-file to write - default: '.cala_key'

        -v, --vigenerekey-len: length of vigenerkey (byte) - default: 2000
        -r, --rsakey-len: length of rsa-key (decimal places) - default: 154, maximum: 154
```

Figure 10-2. The crypttool usage screen

By default, the program generates a file named `.cala_key` containing three keys:

- a vigenere key for encryption and decryption (the same key is used for both operations), 2000 bytes long
- a RSA key for encryption, using 154 decimal places
- a RSA key for decryption, using 154 decimal places

To get higher security, the RSA keys for encryption and decryption should be splitted. The client processes only need to encrypt the data, the servers only need to decrypt it.

> **Note:** The `crypttool` uses system specific random functions to create the keys, so if called two times with the same parameters, different keys are generated. If using separate keys for clients and servers, ensure these keys are generated during the same program run.

The crypttool supports the following arguments:

parameter	example	description	default value
-c <keyfile>	-c client_key	A client key (encryption only) is created and saved into the given file.	unset
-s <keyfile>	-s server_key	A server key (decryption only) is created and saved into the given file.	unset
-a <keyfile>	-a complete_key	A complete key (encryption and decryption) is created and saved into the given file.	-a .cala_key
-v <bytes>	-v 8000	Sets the length (in byte) of the vigenere key to create.	-v 2000
-r <length>	-r 100	Sets the length (in decimal places) of the prime numbers used by the RSA key to create. (Maximum: `154`)	-r 154

Note that the maximum length for RSA keys is `154` decimal places. If a higher value is given, the program creates a key with the maximum length of `154`. (For comparison: a 512 bit number has up to 155 decimal places.)

Supervision of connections

For CALA processes which are reachable over un-secure connections, there are several features to tell CALA to create events if there are problems with any client. For detailed description of the created events see *CALA created events* .

Encryption error events

The generation of encryption error events is enabled by default. This feature can be disabled with the parameter **-**`YE` on the servers command line.

An encryption error event is generated when a client sends an encrypted package that cannot be decrypted with the servers key. The server rejects the faulty package and closes the connection to the client.

Connection accepted event

When the parameter `-CAE` is set, the concerned server sends a connection accepted event each time a remote client connects to it. This behavior is disabled by default.

Accept timeout events

To prevent CALAfrom being blocked by an invader who opens many connections but sends no data, the parameter `-CAT` can be set.

This parameter takes one argument: the maximum time between a (remote) client connect and the reception of the first data package (in seconds). The argument may be a negative number, which means that no event is generated when the connection is closed (the timeout is set to the absolute value of the argument). If -CAT0 is configured, this feature is disabled.

The default setting for this feature is -CAT-30 which means, that a client has to send the first data package within 30 seconds after it connected. If no package is received within this time, the connection is close, but no event is generated.

Connection lost events

A CALA process creates a connection lost event, if the connection to a (remote) client broke down. This can be harmless e.g. if the administrator stopped CALA on this machine, but it can also be a sign for network problems or a crash of the client machine.

The parameter `-CLE` tells a CALA server to create connection lost events, this feature is disabled by default.

CALA communication over firewalls

To enable CALA to communicate over firewalls, there are some preconditions that have to be full filled.

The illustration shows an example client/server configuration with firewall.

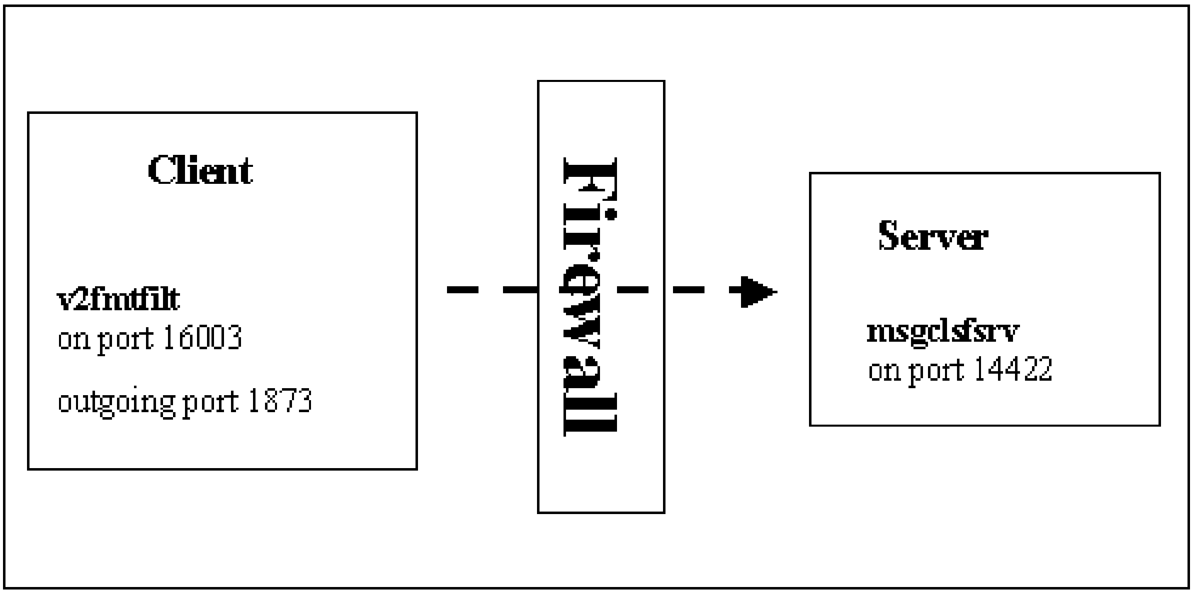


Figure 10-3. CALA sending events over a firewall

The following preconditions have to be fulfilled:

- The client must have permission to connect from local port `1873` to port `14422` on the server.
- The firewall must allow bi-directional communication (the client sends data to the server, the server sends acknowledges to the client).

CALA is able to communicate over firewalls using network address resolution (NAT).

CALA communication over DMZ

To bridge over a demilitarized zone (DMZ), a `calaproxy` can be installed between the two firewalls. Both communication lines need to full fill the preconditions mentioned above.

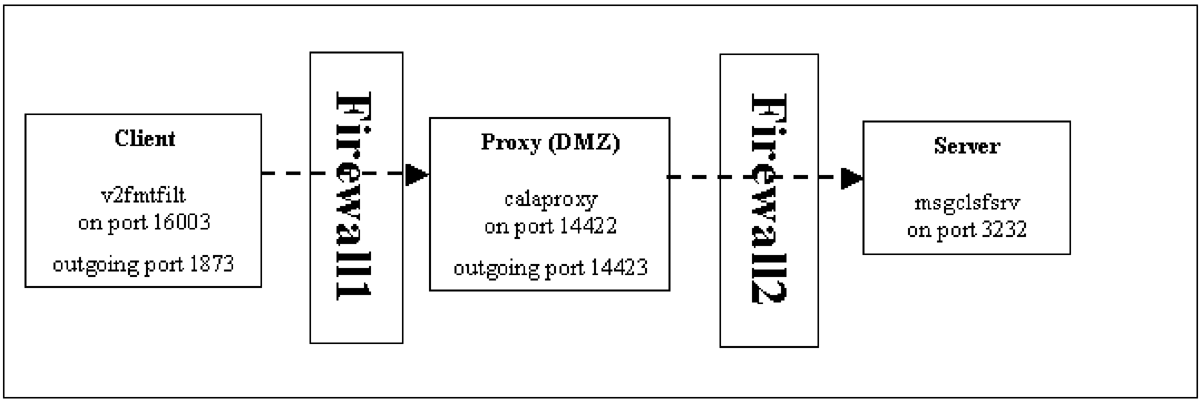


Figure 10-4. CALA sending over a DMZ

Configuration of `Firewall1`:

- Client must be allowed to connect from local port `1873` to port `14422` on `Proxy`.
- Bi-directional communication on the connected socket must be allowed.

Configuration of `Firewall2`:

- Proxy must be allowed to connect from local port `14423` to port `3232` on Server
- Bi-directional communication on the connected socket must be allowed.

Example configuration for ip chains on linux

This is an example configuration for a linux firewall allowing CALA communication from client `10.0.1.1`, port `1873` to server `192.168.1.2`, port `1422`.

The linux firewall has two network connections:

- eth0 is connected to the client s network 10.0.1.x
- eth1 is connected to the server s network 192.168.1.x

```
# 10.0.1.1 -> 192.168.1.2:6192 (data)
ipchains -I input 1 -i eth0 -p tcp -s 10.0.1.1 1873 -d 192.168.1.2 1422 -j ACCEPT
ipchains -I output 1 -i eth1 -p tcp -s 10.0.1.1 1873 -d 192.168.1.2 1422 -j ACCEPT

# 192.168.1.2:6192 -> 10.0.1.1 (acknowledges)
ipchains -I input 1 -i eth1 -p tcp -d 10.0.1.1 1873 -s 192.168.1.2 1422 -j ACCEPT
ipchains -I output 1 -i eth0 -p tcp -d 10.0.1.1 1873 -s 192.168.1.2 1422 -j ACCEPT

# forwarding eth0->eth1, eth1->eth0
ipchains -I forward 1 -p tcp -s 10.0.1.1 1873 -d 192.168.1.2 1422 -j ACCEPT
ipchains -I forward 1 -p tcp -d 10.0.1.1 1873 -s 192.168.1.2 1422 -j AC
```

```
...  CEPT
```

Example 10-1. Example ip chains rules

Revert connections: Servers connecting to clients

Some Firewalls allow connections only be initiated from the internal (private) network and deny all connections initiated from the internet.

If a CALA client runs for example on a Webserver in a DMZ and should report events to a CALA server behind such a firewall, there is need for a new communication mechanism. This it the mechanism of demand clients and tagets. The Firewall allows only connection initiated from the internal Network, so the server must connect to the webserver which then sends the data over the established connection.

To implement this behavior, both, the client (Webserver) and the server (Server) need a special CALA configuration, which is explained in the next sections.

clients waiting for servers to connect

The client opens tcp port and allows servers to connect to. Until a server has connected, events are cached, so nothing is lost. After a server has connected and the connection is established, the clients sends the events in real time to the server, just like it does with normal targets.

A server which is expected to connect to a client is called a demand target and needs a special configuration in the `logctlsrv.conf` file. The targets entry which is used for normal targets is replaced by the demand_targets entry like described below.

```
demand_targets!<local network device>;<local port>;<server ip>;<server p ↙
ort>;<cryptlevel>;{<local network device>;<local port>;<server ip>;<ser ↙
ver port>;<cryptlevel>}
```

Figure 10-5. Format of demand_targets instruction

These are the parameters:

local network device

: the local network device (ip address) on which the client is listening for incoming requests from demand servers, this should be the address of the network card connected to the internal (private) network, specify `*` to use all network cards.

local port

: the local port on which the client is listening for incoming request

server ip

: ip address of servers allowed to connect to this client, the ip address may contain the `*` wildcard to allow a range of ip addresses to connect (or only `*` to allow all servers) server port

: the port the server is connection from (or `*` to allow all server ports)

cryptlevel

: the encryption level used to communication with the server

```
demand_targets!10.0.1.2;14423;10.0.3.*;3233;3
```

Example 10-2. Example `demand_targets` instruction

The example client listens on port `14423` of the network device `10.0.1.2` and accepts connection from all servers from the network `10.0.3` from their local `3233` ports. Connection with encryption level `< 3` are denied.

servers connecting to clients

In contrast to a normal server, which waits for clients and needs no special configuration about this, a server which should connect to a client needs some further configuration: the demand_clients statement which configurates the clients to connect to.

```
demand_clients!<client ip>;<client port>;<local network device>;<localpo
rt>;<poll interval>;<cryptlevel>{;<client ip>;<client port>;<local netw
ork device>;<localport>;<poll interval>;<cryptlevel>}
```

Figure 10-6. Format of demand_clients instruction

A demand client needs the following paramters:

client ip

: the ip address of the client to connect to

client port

: the port on the client to connect to

local network device

: the local network device to use to connect to the client (`*` may given to use all available network devices)

localport

: the local port to connect to the client (`*` may be given to use any port)

poll interval

: the poll interval (in seconds), if the connection to the client should not be hold, if a poll interval of `0` is given, the connection will be hold.

cryptlevel

the minimum encryption level to be used with the client

If a poll interval is given, the server connects to the client, retrieves all cached messages and disconnects after the reception of all these messages and will reconnect after the specified time to retrieved messages again.

```
demand_clients!10.0.1.2;14423;10.0.3.1;3233;120;3
```

Example 10-3. Example demand_clients instruction

This would be a valid configuration for a communication with a client configured the section above. The server connects via it s network device `10.0.3.1` from local port `3233` to the client `10.0.1.2` on port `14423` to receive messages. Encryption level `3` is used. The server disconnects after retrieving all messages and reconnects `2` minutes later to receive new messages (if there are any).

A sample client/server configuration using demand clients

This is a sample configuration using the demand clients feature:

A CALA client on `10.0.1.2`, running `calamon` and a CALA server on `10.0.3.1` running a `msgclsfsrv` and a `reportemit`.

The server pools for new events every 120 seconds using the outgoing port 3233 to connect to port 14423 on the client machine (10.0.1.2).

```
# Configuration for CALA client 10.0.1.2
# waiting for the server to poll for events

serverlist=calamon

calamon=run!calamon -P 14422 T 10 AB 127.0.0.1,port!14422,cmdtab!cmdtab
.tst,demand_targets!10.0.1.2;14423; 10.0.3.*;3233;3,conf!run;port;cmdta
b;demand_targets
```

Example 10-4. A client configuration using demand targets

```
# Configuration for CALA server 10.0.3.1
# polling the client for events
serverlist= msgclsfsrv,reportemit

msgclsfsrv=run!msgclsfsrv -P 3232 AB 127.0.0.1,port!3232,targets!report
emit,demand_clients!10.0.1.2;14423;10.0.3.1;3233;120;3,conf!port;run;ta
rgets;demand_clients

# new column
reportemit=run!reportemit -P 3234 AB 127.0.0.1,dest_file!test.rep, port
```

```
...  !3234, conf!port;run;dest_file
```

Example 10-5. A server configuration: using demand clients

Appendix A. The v2 format

The v2 format is the description language for complex logfile formats which do not comply with the logfile standard (single-line entries, fixed format).

The v2 format is capable of describing formats which

- possess a multi-line sentence format
- possess a sentence format which cannot be defined in advance without ambiguity, or which contains a repetitive sentence format

Storage form

Format files for V2FMTFILT must be saved as a file.The filename can have any extension, although the extension ".v2s" is recommended.

Identifiers

Identifiers are class names and variables (slots) in the V2S format.

Identifiers in V2S must start with a letter and can contain any sequence of alphanumeric characters.Valid characters include uppercase and lowercase letters as well as digits and the underscore

Identifiers are used directly by V2FMTFILT to set up FIRs (Filter Input Records). Variables with designators starting with a leading underscore are treated as temporary and do not occur in the resulting FIRs.

General design of the v2 format

A V2 format file contains three main sections:

- a header
- definitions of global variables
- declarations of sub expressions and classes

Comments

Comments are allowed before, between and after the sections and between expressions in the declarations section.

There are two different comment types supported, similar to comments in C/C++.

```
/* <comment> */
// <comment terminated by new line>
```

Example A-1. Example of comments in v2s

Header

The header has the format:

```
SPEC <name>
```

Figure A-1. Format of v2s spec expression

`<name>` is any identifier for the format specification.

```
SPEC SNA
```

Example A-2. Example of a v2s spec expression

The header information is obligatory.

Global Variables

Global variables are definitions of general FIR slots, which should occur in each created FIR. The class definition may overwrite or delete this slot.

```
GLOBAL BIND <slot name> TO "<string>"
```

Figure A-2. Format of v2s global bind expression

```
GLOBAL BIND source to cala
```

Example A-3. Example of a v2s global bind expression

The definition of global variables is optional.

Automatically assigned variables

There are some variables, which are automatically assigned by the parser. This variables can also be used in the class finalization.

> **Note:** Fields starting with $ should be accessed in a read only manner only.

field name	description
$HOSTNAME	name of the host the event occurred on

field name	description
$ORIGIN	ip address of the host the event occurred on
$ADAPTER_HOST	name of the host, which read the event
$LOGFILENAME	name of the logfile the event was read from
hostname	name of the host the event occurred on
origin	ip address of the host the event occurred on
adapter_host	name of the host, which read the event

Variables to set timestamp

The event's timestamp is initialized with the current time (the time when the v2 format filter starts parsing the event). Using the following fields, the timestamp can be adjusted.

field name	description
DAY	day of month (1-31)
HOUR	hour (0-24 or 0-12 in 12-hour mode, see below)
MINUTE	minute (0-60)
SECOND	second (0-60)
TIME_POSTFIX	Setting `TIME_POSTFIX` to any value switches to 12-hour-mode. Sets time to P.M. if `TIME_POSTFIX` is set to any value starting with P or p.

Classes and sub-expressions

Classes

Every format description file must contain a series of class definitions. These definitions were processed top-down at parsing time. This means that if more than one definition matches, the first one is taken.

```
IF expression CLASS name [ FINALIZATION: BIND <slot> TO <any sequence of V2S expressions > ]
```

Figure A-3. Format of v2s class expression

```
IF (
SUBEXPRESSION TIMESTAMP
"myprocess shut down"
```

```
) CLASS MYPROCSHUTDOWN
```

Example A-4. Example of a v2s class expression

A classname may occur several times in one format description file.

Sub-expressions

Sub-expressions can be defined and called in the same way as macros.

The declaration must take place in the File-Scope, i.e. at the same level as the classes are defined. Ideally, all SUBEXPRESSION definitions should be defined before the list of class definitions.

```
SUBEXPRESSION name ( expression )
```

Figure A-4. Format of v2s subexpression definition

```
SUBEXPRESSION IPADDREXPR ( GROUP BIND IPADDR %d { '. %d } )
```

Example A-5. Example of a v2s subexpression definition

The "Macro" is called using

```
SUBEXPRESSION name
```

Figure A-5. Format of v2s subexpression call

```
"Text" SUBEXPRESSION IPADDR "Text"
```

Example A-6. Example of a v2s subexpression call

Expressions

Matching types

Character Match (individual characters)

Syntax < 'x > defines a character match.

<x> can be any character for which a match is to be found.

Example:

'A matches the letter A

This syntax makes it possible to match up special characters.

Character Match (individual characters by ASCII code)

Syntax '\x defines a character match. `x` is the decimal ASCII code of the character to be found.

```
\65
```

Example A-7. Example v2s character match: matching the letter `A`

This syntax makes it possible to match up special characters.

Multi match (multiple match)

Syntax %x [BIND field] matches a sequence of characters and links the result to a specified field (slot).

```
%a BIND FIELD1
```

Example A-8. Example v2s: matching a sequence of alphanumeric chars

If the field name starts with a leading underscore, the field is for local use only and does not appear in the resulting event. Nevertheless it can be used in the finalization section of the class.

Multi match type d (decimal match)

A sequence with at least one decimal numeral is matched.

e.g. %d BIND NUMBER23

Multi match type a (alphanumeric match)

A sequence of at least one alphanumerical character is matched. Alphanumeric characters include letters `A-Z`, `a-z`, as well as digits `0-9`.

Multi match type w (white space match)

A sequence with at least one white space character (space or tab key, ASCII characters `SPC` and `HT`, code 32 or 9) is matched.

Multi match type n (new line match)

Precisely one line feed is matched. (`LF`, ASCII-Code 10): where necessary, a `CR` (ASCII code 13) is skipped for this.

Multi match type b (blank line match)

Matches precisely one blank line. This can contain any number of `SPC` and `HT` characters.

Character Match s<number>

Using the notation %`<number>`s, it is possible to read out a definable number of characters. This makes it possible to disassemble an input string into any number of sub-sections, e.g. to generate a standard time format out of any given time stamp.

Multi match type s (string match)

There are six operational modes:
%s TERM 'x
%s TERM 'x BIND field
%s TERM \x
%s TERM \x BIND field

The first format matches all characters up to the specified terminator (not including this character), and links the result to a field when necessary. The character can be given as the character itself (x) or as it s ASCII code (\x).

%s TERM WHITESPACE
%s TERM WHITESPACE BIND field

The second format matches all characters up to the next white space character (`SPC`, `HT`, `CR` or `LF`), and links the result to a field if necessary.
%s TERM NEWLINE
%s TERM NEWLINE BIND field

The third format matches all characters up to the first line break (UNIX and DOS/Windows line breaks) and links the result to a field if necessary.
%s TERM BLANKLINE
%s TERM BLANKLINE BIND field

The fourth mode matches all characters up to the first blank line and links the result to a field when necessary.
%s TERM termination string
%s TERM termination string BIND field
%s TERM (alt. term. string 1 | alt. term. string2
%s TERM (alt. term. string 1 | alt. term. string2) BIND field

The fifth format matches all characters up to the first occurrence of the given termination string and links the result to a field if necessary. It is also possible to give a list of alternative termination strings, which means: match the characters up to the first occurrence of one of the given strings.
%s TERM SUBBEXPRESSION subexpr
%s TERM SUBBEXPRESSION subexpr BIND field

The sixth format matches all characters up to the next occurrence of subexpr (not including this subexpression) and links the result to a field if given.

To ensure the match is successful, **at least 1** character must be matched.

Multi match type S

This special type of string match behaves in the same way as the standard multi-match type **s** with one exception: processing of the string stops at the end of the first line.

%S TERM `<term expression>`
%S TERM `<term expression>` BIND field

Note: The implementation of this match has changed from CALA version 1.1b to CALA version 2.1

Old implementation (<= CALA 1.1b): Match the string up to the termination condition or if this condition is not fulfilled until the line ends, match the rest of the line.

New implementation (>= CALA 2.1): Match if the termination condition can be fulfilled within the current line.

Constant string match

By specifying

```
"any text"
```

Figure A-6. Format of v2s constant string match

(any text in double quotes), precisely that section of text is matched.

You can also specify a list of alternative strings to match:

```
(" alt string1" |  "alt string2" |  "alt string 3" )
```

Figure A-7. Format of v2s constant string match with alternatives

Escape sequences have not yet been implemented. In an instance of this kind, the special character must take the form of a character match (‘<any character>).

Subexpression match

The following line calls a subexpression match

```
SUBEXPRESSION name
```

Figure A-8. Format of v2s subexpression match

The sub-expression indicated is matched (refer to subexpression section).

Mandatory, optional and repetitive expressions

Mandatory expression

The use of parentheses (round brackets) around any code group (<your code>) indicates that an expression is mandatory.

This means that all matches enclosed in brackets must be performed.

```
001  ( ’A %d )
```

Example A-9. Example for a mandatory v2 expression group

Tihs matches the letter A and one or more numerals.

Examples:

Expression	Source	Match
(’A %d)	A1PQR	A1
(’A %d)	A2324 XYZ	A2324

Optional expression

The use of square brackets around any code group [<your code>] indicates that an expression is optional.

This means that all matches enclosed in these brackets should be made either 0 or 1 time.

```
001  ’A %d [ ’. %d ]
```

Example A-10. Example for an optional v2 expression group

matches letter A and one or more digits and optional a following dot and another sequence of digits.

Expression	Source	Match
’A %d [’. %d]	A1PQR	A1
’A %d [’. %d]	A1.24XYZ	A1.24

Optional repetitive expression

The use of curly brackets around any code group { <your code> } indicates that an expression is optional, and can be repeated several times.

```
’A %d { ’. %d }
```

Example A-11. Example for an optional-repetitive v2 expression group

matches letter `A` and one or more digits as well as (optional and repetitive) a following dot and another sequence if digits.

Expression	Source	Match
’A %d { ’. %d }	A1PQR	A1
’A %d { ’. %d }	A1.24.35XYZ	A1.24.35

Group binding

An expression can be started with a group statement: GROUP BIND field

This binds all characters matched by this expression to a field. If a group statement is used within any expression, it must be set in parenthesis.

```
( GROUP BIND IPADDR %d { ’. %d } )
```

Example A-12. Example for a v2s group bind expression

matches a series of numerals interspersed with dots, assigning the field `IPADDR`.

If the field name starts with a leading underscore, the field is for local use only and does not appear in the resulting event. Nevertheless it can be used in the finalization section of the class.

Example of format file sna.v2s

This section provides a description of an SNA server error logfile:

```
SPEC SNA

SUBEXPRESSION TIMESTAMP (
   %d BIND HOUR
   ':
   %d BIND MINUTE
   ':
   %d BIND SECOND
   " "
   %a BIND TIMEZONE
   "  "
   %d BIND DAY
   " "
   %a BIND MONTH
   " "
   %d BIND YEAR
)

SUBEXPRESSION TIMESTAMPLINE (
   SUBEXPRESSION TIMESTAMP
   " "
   %d BIND CODE1
   '-
   %d BIND CODE2
   '(
   %d BIND CODE3
   '-
   %d BIND CODE4
   ')
   " "
   %a BIND CODE5
   " "
   '(
   %a BIND CODE6
   ')
   %n
)

IF (
   "============ Log file initialised " SUBEXPRESSION TIMESTAMP " ============" %n
) CLASS LOGINIT

IF (
   SUBEXPRESSION TIMESTAMPLINE
   "Abnormal UNBIND request received" %n
   "Sense code" %w '= %w %a BIND SENSECODE %n
   "Local LU name" %w '= %w ( GROUP BIND LOCALLU %a '. %a ) %n
   "Partner LU name" %w '= %w ( GROUP BIND PARTNERLU %a '. %a ) %n
   "Mode name" %w '= %w %a BIND MODENAME %n
   "UNBIND RU :" %n ( GROUP BIND UNBINDRU %a { " " %a } ) %n
) CLASS ABNORMALUNBIND FINALIZATION:
     BIND msg TO "Abnormal UNBIND request received " + SENSECODE,
     BIND SENSECODE TO NOTHING;
```

```
054
```

Example A-13. An example v2s format file

The last class definition described here sub-divides the event into various slots (`SENSECODE`, `LOCALLU`, `PARTNERLU`, `MODENAME` and `UNBINDRU`) and into slots which are defined when sub-expression `TIMESTAMPLINE` is called up. Processing at the end of a class definition (`FINALIZATION`) involves combining slot from text "`Abnormal UNBIND request received`" and the content of the `SENSECODE` slot. The `SENSECODE` slot is deleted afterwards.

Appendix B. The command table file format

The command table file contains a set of parameters for each monitor task. Each of this parameters has to be configured in a separate line. comment lines are prefixed with ## (two #).

The following parameters must be given:

- script name - path and name of the script to be started
- command line parameters - parameters which are passed to the script
- primary data type, secondary data type and event class - type of event to be created
- stdout field - FIR field to receive the script output to stdout
- stderr field - FIR field to receive the script output to stderr
- return code field - FIR field to receive return value of the script
- comment prefix - prefix which marks a line of the script output as comment
- comment field - FIR field to receive comment lines (which are remove from stdout field)
- escalation field - FIR field to receive escalation level (is set from escalation file)
- escalation file - name of escalation file (see escalation file description below)
- the execution times specification (like crontab entries in Unix)
  - execution months
  - execution days of month
  - execution days of week
  - execution hours
  - execution minutes
  - execution seconds

- execution period - length of period in seconds
- message template - a template for the message to be written into the message slot (may contain links to other fields)
- message slot - the name of the message slot

Parameters may be enclosed in double quotes, double quotes within a quoted string have to be masked by backslashes.

Example:

```
##------------------------
/home/cala/scripts/check_disk.sh
"/dev/hda1 /dev/hda2"
tec
calamon
CALA_Monitors
value
$stderr
$return
#
$comment
severity
disk_esc.esc
```

```
1-12
1-31
0-7
0-24
0-60
0-60
"Filespace monitor for <proble_arg> returned <value>. Additional information: <$comment>"
msg
```

Appendix C. msgclsfsrv Text Formatting

A text formatting info string contains of a string of one or more of the following characters:

char	relevance	function
L	Input	align left
R	Input	align right
W<*n*>	Input	field width <n> character
F<*n*>	Input	first character is <n>
T<*n*>	Input	last character is <n>
N	Input	field is numerical
O	Input	field is octal
D	Input	field is decimal
H	Input	field is hexadecimal
h	Output	field is hexadecimal
w<*n*>	Output	field width <n> characters
l	Output	align left
r	Output	align right
i	Output	transform characters to lower case (ignore case feature)

Typical application:

E.g. if you wish to use the first 5 characters of the user field to process a Message Map definition, the declaration takes on the following form:

```
Usermct=key!user!LW5,fields!devision;location
```

Example C-1. Using the first 5 characters of the user field to process a Message Map definition

The Message Map declaration used in the Usermap uses the first 5 characters for division and location

The relevant MessageMap file could possess the following format:

```
Admin SystemsManagement London
```

Example C-2. An example message map for the above definition

All events that contain the value Admin in the first 5 characters of the slot have the division slots mapped with the value SystemsManagement . The location slot is mapped with the value London

Note: The slots entitled location and division are generated if they do not exist.

Some examples how text formatting works

This is the sample string: `0123456789ABCDE`

F5T6NH

The text format string F5T6NH (first 5, last 6, numerical, read as hex) results in `86`

F5T6: `56` (numeric, hex) =: `0x56` = `86` (decimal)

F5T6NDh

The text format string F5T6NDh (first 5, last 6, numerical, read as decimal, output as hex) results in `38`

F5T6: `56` (numeric, decimal) =: `56` = `0x38` (hex)

W3NDh

The text format string W3NDh (field width 3, numerical, read as decimal, output as hex) results in `C`

`W3`: `012` (numeric, decimal) =: `12` = `0xC` (hex)

RW3NH

The text format string RW3NH (align right, field width 3, numerical, read as hex, output as decimal, which is default) results in `3294`

RW3: `CDE` (numeric, hex) =: `0xCDE` = `3294` (decimal)

Appendix D. Pchread XML Configuration

This chapter describes the content of the XML configuration file for the PCH-Reader, a `javasrv` component (see *javasrv* for a description of `javasrv` and `pchread`).

The configuration file defines clusters of hosts and applications, that should be monitored through pchread, and events from these locations, whose values conditionally should be send as FIRs to the CALA server. Beside these, optionally so-called properties can be defined, which are used as a kind of constants or flags in the configuration file.

The toplevel XML tag must be the configuration tag. The file must start like this:

```

<?xml version="1.0" encoding="iso-8859-1"?><configuration xmlns="http://www.cenit.de/eb/sm/xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.cenit.de/eb/sm/xml javasrv_pchread.xsd">
```

Figure D-1. the xml file header

And it must end with:

```

</configuration>
```

Figure D-2. the xml file footer

An example will be installed with the pchread component.

Pchread checks the validity of an XML configuration file against the standard XSD, that is installed with the Pchread component. So the formal definition of the assembly of the XML configuration file is the before mentioned XSD file. The following is a more textual description of the configuration.

Properties

There are two kinds of properties:

1. Flags for pchread.
2. Definition of constants, referenced later in the document.

The Flags are:

name	value	description
`requesthistory`	`false` or `true`	Should pchread ask for old event values, at the first connect to a listener?
`bookkeeping`	`false` or `true`	Should pchread create and maintain a book keeping file?

name	value	description
`de.cenit.eb.sm.cala.utils.logging.Logger.logMethod`	`CENIT`, `LOG4J` or `OFF`	The logging method used by pchread. In a production environment only CENIT or should be used.
`de.cenit.eb.sm.cala.utils.logging.Logger.logLevel` [a]	`NONE`, `INFO`, `WARN`, `FATAL`, or `DEBUG`	The logging level. In a production environment NONE or FATAL should be used. Beware DEBUG will produce a huge amount of output, so the log file will be grow rapidly.

Notes:
a. At the moment, a log level “more sensitive“ than the one set in the `logctlsrv.conf` for the `javasrv` will have no effect. Nevertheless the level must be defined in the XML configuration file. This behavior can change in future releases.

Constants can be defined like this:

```
<property name="a distinct name"value="a fixed value"/>
```

Later on one can reference these values through their distinct names with:

```
$prop.<a distinct name>
```

```
<property name="DefaultPort"value="32775"/>
<application name="Image Services"port="$prop.DefaultPort">
```

Example D-1. An example properties configuration

The prefix `$prop` is a namespace identifier. The `$prop` namespace holds all defined properties and their values.

Request for historic data

The requesthistory flag should be used with care, as it can lead to a heavy load at the remote machine. It also can delay the normal operation of pchread, as some listeners store very old event data. You should also keep in mind, although it is possible to ask for historic data and turn book keeping off, but in a production environment this makes no real sense.

Clusters, Hosts and Applications

The first required tag is the clusters tag. Inside this tag, a logical structure of hosts and applications can be defined. The only allowed subtag is the cluster tag, which must occur at least once. It defines a so-called cluster, which is a set of hosts. Inside the cluster tag there must be at least one host tag, and inside this, there must be at least one application tag.

All three of these tags have the required argument name. The cluster name can be chosen arbitrarily, as it has no real representation outside the pchread, like host and application. The host name must be a valid hostname or IP of a machine that we want to monitor. The application name must be a valid application name. That is the name, the listener, running inside an application at the given host, returns, when asked for the name of its application. All these names are used later on in the events section. The names should not contain the asterisk or the question mark, and case does matter.

Beside the name, the application additionally has the required attribute port and the optional attribute interval. The port attribute defines the port, the pchread should use to connect to the listener at the given port. At the moment, for every application at one host, the port number of the primary listener should be used. So there will be no different ports for the listeners of one host. Hint: Define a property for the port.

Interval defines the amount of seconds, pchread should wait between the request of new event data. As pchread is triggered through javasrv, this interval is only a lower boundary for event requests. If javasrv does not trigger the pchread component, no requests will be sent to the remote listeners at all.

For a detailed description of the FileNet System Monitor framework, and a deeper understanding of the listener/manager mechanism, you should read the appropriate FileNet documentation.

Events

The clusters tag must follow exactly one events tag. In it, there must be at least one event tag.

The event tag defines a rule. A rule, that describes the condition, when pchread should send what kind of data as to the CALA server. If the condition of such a rule is fulfilled, the specified data will be send, and the event will be marked as processed . So pchread will no longer search for another matching rule. In short: The first match serves. That should be considered, when arranging the event tags in the configuration file.

Conditions

The first condition of the event rule is the path, which must be given as a required argument to the event tag. The path can contain the asterisk as a wildcard for an arbitrary number of characters; but only in the part after the hostname. The slash is used as a separator in the path. The format of the event path is:

```
/<cluster name>/<host name>/<application name>[/<PCH class name>/[<event
 specific data>]]
```

Figure D-3. Format of event path

Cluster, host and application name must be defined in the clusters section of the configuration file. It is not an error, if one or all of them are undefined, simply the rule will never match. So you should be aware of typos, and remind, that case does matter. Properties are not allowed inside the path.

The PCH class name can be one of `DISK`, `NETWORK`, `METER` or `USER` . The event specific data is a part, that is typical for every event, and must be known by the editor of the configuration file.

An example of a valid event tag is:

```
/My Cluster/localhost/Image Services/CPU/*
```

Example D-2. An example event tag

It will match any `CPU` event from the `Image Services` application running at the local host.

Optionally, there can be more conditions. They must be defined in the condition subtag of the event tag. If the conditions defined, will evaluate to true , the data will be send. If there is no condition tag, only the path does matter.

Only numerical comparisons can be defined. They will be matched against the actual event value. The comparisons can be logically combined with the boolean operators and, or and xor. These tags do not have any attributes. The comparison tag has two required attributes: `operator` and `compValue`.

Known operators for numerical comparison:

operator	**description**
`eq`	True, if the event value is equal compValue
`ne`	True, if the event value is not equal compValue
`gt`	True, if the event value is greater compValue
`lt`	True, if the event value is less than compValue
`ge`	True, if the event value is greater or equal compValue
`le`	True, if the event value is less or equal compValue

Actions

The required action tag, that follows the optional condition tag, defines the actions that are taken, if the path matches, and the optional conditions will evaluate to true.

The only allowed subtag of the action tag is the assignment tag. There must be at least one assignment tag in side an action tag. The assignment tag has two required attributes: name and value.

The name attribute defines the field name in the FIR that will be sent to the CALA server. The value attribute defines the value of that field. The last attribute can contain defined properties, and references to other namespaces.

Allowed namespaces are

namespace	description
$prop	Properties, defined in the configuration file.
$env	OS environment variables.
$event	Value and path of the actual event.

To reference a variable in a namespace, the namespace and the name of the variable must be concatenated with a single dot . For $event only class, path, when and value are allowed. The first will give the PCH class number of the event, the second the full path of the actual event, the third the timestamp (in milliseconds), and the last the actual event value. For `$env` all defined OS environment variable names are allowed. Beware, it is possible that a Java Security Manager prohibits access to the OS environment.

```
<event path="/My Cluster/localhost/Image Services/*">
   <action>
      <assignment name="$CLASS" value="$event.class"/>
      <assignment name="$PRITYPE" value="v2"/>
      <assignment name="$SECTYPE" value="fnpch"/>
      <assignment name="value" value="$event.value"/>
      <assignment name="msg" value="$event.path"/>
      <assignment name="$area" value="Listener"/>
      <assignment name="$mode" value="1"/>
      <assignment name="severity" value="CRITICAL"/>
      <assignment name="$info" value="$event.path"/>
      <assignment name="$LOGFILENAME" value="$event.path"/>
   </action>
</event>
```

Example D-3. Example of a valid event rule definition

As a default, the `$CLASS` field fill be set to the PCH class name of the event, `$PRITYPE` will be `pch`, and `$SECTYPE` will be `tec`. `$CTIME` will be set to the timestamp (in milliseconds) of the event.

Appendix E. CALA created events

This chapter describes the structure of events created from CALA. Depending on the configuration these events may be generated or not (see also *Security* for details).

There are some common fields available in all CALA generated events:

slot name	value
date	the events creation time
hostname	the name of the host the event occurred on
adapter_host	the name of the host running the CALA component reading the event
origin	ip address of `adapter_host`
primary data type	stream classification primary type (`tec` or `v2`)
secondary data type	stream classification secondary type (application id)
class	event class

CALA Testevent

A test event is generated by calling the `logctlcmd` program with the command test and the logical name of the server which should create a test event.

logctlcmd test ascfileread

Example E-1. Creating a test event for ascfileread

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
primary data type	`tec`
secondary data type	`cala_test`
class	`CALA_Testevent`

Connection Accepted Event

These events are created if any client has connected to a server (see description of parameter *-CAE*).

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
msg	`A client connection has been accepted.`
client	ip address of the client which has connected
primary data type	`tec`
secondary data type	`cala`
class	`CALA_Client_Connect`

Connection Lost Event

These events are created if the connection to any remote client has been lost (see description of parameter *-CLE*).

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
msg	`The connection to the client process has been lost.`
client	ip address of the client which has disconnected
primary data type	`tec`
secondary data type	`cala`
class	`CALA_Connection_Lost`

Accept Timeout Event

An accept timeout event is generated if a remote clients tried to connect, but didn t send any data (see description of parameter CAT).

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
msg	`The client tried to connect, but didn’t send any data. The connection has been closed.`
client	ip address of the client which tried to connect
primary data type	`tec`
secondary data type	`cala`
class	`CALA_Accept_Timeout`

Encryption Error Event

An encryption error event is created if any client sends corrupt encrypted data.

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
msg	`An encryption-error occured while communicating.`
client	ip address of the client which sent corrupt data
primary data type	`tec`
secondary data type	`cala`
class	`CALA_Encryption _Error`

Heartbeat Event

An heartbeat event is send periodically if the parameter *`-ZHEARTBEAT_PERIOD=<secs>`* is given.

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time

slot name	value
hostname	name of the host creating the event
origin	ip address of the host creating the event
primary data type	`tec`
secondary data type	`cala`
class	`CALA_HEARTBEAT_OK`

Status Events (Startup/Shutdown)

The status events are created for remote targets or from the T/EC interface servers, if the parameter *`-ZCREATE_STATUS_EVENTS=1`* is given. The status events are not created for local targets.

Startup event:

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
primary data type	`tec`
secondary data type	`cala`
class	`CALA_STARTUP`

Shutdown event:

slot name	value
source	`Cenit Advanced Logfile Adapter`
sub_source	name of component creating the event
date	event creation time
hostname	name of the host creating the event
origin	ip address of the host creating the event
primary data type	`tec`
secondary data type	`cala`
class	`CALA_SHUTDOWN`

Appendix F. Additional tools

This chapter provides an overview for various scripts and tools included on the FSM CD.

install_cala.sh

General description

install_cala.sh can be used to install CALA, CALAGUI or V2SEdit directly from the installation CD. It can be controlled either interactively or by specifying the appropriate command line parameters.

This script is directly executed by the graphical CALA Installer tool.

> **Note:** On Windows NT, make sure that `gzip.exe` is found in your search path. If you do not have the GNU tools installed, you can find this tool in the directory `/MISC/w32-ix86/tools` on the CD. A Perl Version 4 interpreter needs to be present on the system, too.

`install_cala.sh` requires some external scripts that must be located in the same directory:

- `cala_untar.sh` for unpacking the archive
- `calainst.sh` for the actual CALA installation
- `calainst_std.sh` for standard installations
- `calainst_cfg.pl` for configurator installations

The configuration file `logctlsrv.conf` as well as the files referenced from within the configuration must be present either in the same directory as calainst.sh or in the target directory. If no configuration is available, the binaries will be copied, but CALA will not be started.

For UNIX systems, the template for the start-up script, `cala_rc.templ`, must be located in the same directory as `calainst.sh`, too.

> **Note:** All options can be specified in any order. If you omit any of the options, the script will prompt for a value. A default location will be suggested for all directory parameters.

Parameters

The table shows all supported options. To specify an option, you only need to enter the first letter.

option	description

option	description
-product <prod>	defines the product to be installed Possible values are CALA, CALAGUI, V2SEdit. If no product is specified on the command line, the script will prompt for it.
-srcdir <dir>	directory where the installation archive is located The name of the installation archive depends on the selected product: CALAGUI: `calagui.tar.gz`, V2SEdit: `v2sedit.tar.gz`, CALA: `cala.<interpreter>.tar.gz` Default: current directory
-targdir <dir>	directory where the product must be installed For CALAGUI and V2SEdit, an additional subdirectory is created which is named like the selected product. Default: current directory
-cachedir <dir>	directory where the CALA cache files will be created Default: subdirectory .calacache in the target directory
-jdkdir <dir>	directory where the java executable is located, e.g. `c:\\Program Files\\jdk1.1.8\\bin` or `/usr/java/bin` If no j parameter is specified, the script checks if the environment file `<tool>.env` already exists in the `<tool>` subdirectory of the target directory. If no environment file is found, the script will prompt for the directory.
-untar <Y|N>	Set this option to N if you do not need to unpack the CALA binary archive before the actual installation. Default: Y
-remove	Specify this switch if you want to remove the product. If you uninstall CALAGUI or V2SEdit, the name of the tool is appended to the target directory if you did not specify it (see parameter targetdir). If you uninstall CALA, the script checks for an existing CALA installation by searching the startup script (UNIX) or the registry (Windows). If an installation is found, the corresponding settings for CALA_DIR and CALA_CACHE_DIR will be suggested as default.
-noninteractive	suppress user interaction for missing parameters If this switch is specified, default values will be used without user confirmation.
-hostname <hostname>	use this hostname instead of result of hostname command
-keepmonitors	keep current monitoring settings instead of replacing them from the configuration archive
-debug	activate debugging The debug output is redirected to the file install_cala.dbg in the current directory.
-?	show usage message

The directories specified with the options s, -t and c can be given as relative path definitions. In this case, the directories will be searched relative to the current directory.

Note: On NT you must either use forward slashes '/' or double back slashes '\\' in your path definitions.

If the options s, -t and / or c are not specified on the command line, the script will show the corresponding default value. The user can confirm this value or can specify another directory. This behavior can be turned off by specifying the n switch. In this case, the default values will be used without confirmation.

The following table shows which options are required by which product:

option[a]	**CALA**	**CALAGUI**	**V2SEdit**
-product	X	X	X
-srcdir	(X)	(X)	(X)
-targdir	(X)	(X)	(X)
-jdkdir		(X)	(X)
-cachedir	(X)		
-untar	(X)		
Notes: a. X - required, (X) - optional			

If an optional parameter is not specified, the script will use the defaults listed in the table above.

Installation process

The installation process depends on the selected product.

For CALAGUI and V2SEdit, the corresponding tar.gz file will be unpacked to a subdirectory in the target directory. Both packages already contain a start script that is platform independent.

The setting of the JDK directory will be stored in the environment file <tool> .env. This file is created in the <tool> subdirectory that is created by unpacking the installation archive. On Windows platforms, the file <tool> .env.bat will be created for usage in the DOS start scripts.

For CALA, the existing installation script `calainst.sh` will be called.

cala_untar.sh

General description

cala_untar.sh can be used to unpack the .tar.gz archives for CALA, CALAGUI and V2SEdit.

Note: On Windows NT, make sure that gzip.exe is found in your search path. If you do not have the GNU tools installed, you can find this tool in the directory /MISC/w32-ix86/tools on the CD.

Parameters

The table shows all supported parameters.

option	description
<from_dir>	directory where the installation archive is located
<product>	product (CALA, CALAGUI, V2SEdit)
<interpreter>	interpreter specification if only one archive should be processed. This parameter is optional and applies for processing of CALA archives only.

The binaries are unpacked to the current directory.

Examples

cala_untar.sh /cdrom/CALA/Images cala

unpacks all installation archives into the current directory

cala_untar.sh /cdrom/CALA/Images cala solaris2

unpacks the SOLARIS installation archive and removes the interpreter type "solaris2" from the names of the binaries.

brdcsttool

General description

The broadcast tool is a tool to check the network for CALA broadcast servers. The broadcast tool sends a request into the network and shows a list of all responding servers.

```
brdcsttool <rcv-port> [subnet] <req-port> { [subnet] <req-port> }
```

Figure F-1. Usage of brdcsttool

Parameters

The table shows all supported parameters.

option	description
<rcv-port>	The tcp port on which the brdcsttool listens for server replies.
<subnet>	The subnet to send to broadcast request to.
<req-port>	The port to send the request to.

Examples

```
./brdcsttool 22222 10.0.114.255 16002
queueing request BROADCAST:10.0.114.255:16002
received server location: 10.0.114.201:16002
found any server at port 16002
```

This example shows a successful broadcast request for the subnet 10.0.114.255 on port 16002, the broadcast tools receives the replies on port 22222. Any CALA server (with broadcast enabled) runs on host 10.0.114.201.

If more than one servers are found, all received locations are displayed.

```
./brdcsttool 22222 127.0.0.255 16005
queueing request BROADCAST:127.0.0.255:16005
found NO server at port 16005
```

queueing request BROADCAST:127.0.0.255:16005

found NO server at port 16005

This is a broadcast request with no server found listening on port 16005.

testv2sfile and testfmtfile

General description

These tools can be used for checking v2s format files (testv2sfile) or Tivoli fmt format files (testfmtfilt) for syntactical correctness.

```
testv2sfile <filename> [-options]
testfmtfile <filename> [-options]
```

Figure F-2. Usage of testv2sfile and testfmtfile

Parameters

The table shows all supported parameters.

option	description
<filename>	Name of v2s or fmt-file to be checked.

option	description
-t	show syntax tree
-d	show debug information

Examples

```
./testv2sfile websphere.v2s
*******************************************************************
**                                                               **
**    testv2sfile is part of the CENIT Advanced Logfile Adapter  **
**                                                               **
**  version: 1.01-056   - generation date: May 21 2002 11:16:06  **
**                                                               **
**               (c) 1999-2001 CENIT AG Systemhaus               **
**                                                               **
*******************************************************************
parser initialized ... creating syntax tree ... syntax tree created!
’websphere.v2s’ is a correct v2s-file!
```

```
/.testfmtfile jpo.fmt
*******************************************************************
**                                                               **
**    testfmtfile is part of the CENIT Advanced Logfile Adapter  **
**                                                               **
**  version: 1.01-056   - generation date: May 21 2002 11:16:08  **
**                                                               **
**               (c) 1999-2001 CENIT AG Systemhaus               **
**                                                               **
*******************************************************************
parser initialized ... creating syntax tree ... syntax tree created!
’jpo.fmt’ is a correct fmt-file!
```

Example F-1. Example: calling testfmtfile

sendfir

General description

The sendfir tool creates a CALA event (filter input record) and sends it to a specified component. Please remember, that FIRs cannot be sent to reader or filter components (they would be discarded).

```
sendfir [-H <hostname>] -P <port> [-s] [-f <no.>] ’<fir>’
```

Figure F-3. Usage of sendif

Parameters

The table shows all supported parameters.

option	description
-H <hostname>	Name of the host to send the event to. (optional, default: localhost)
-P <port>	Port of the server to send the event to.
-f <no>	Send the event <no.> times to the server. The event gets an additional field count which holds its sequence number.
-s	show debug information

The FIR has to be given in the following format:

```
label=value{;label=value}
```

Some special values are supported:

$CLASS

sets the FIRs class (default: `Default`)

$PRITYPE

sets the FIRs primary data type (default: `tec`)

$SECTYPE

sets the FIRs secondary data type (default: `cala`)

Examples

```
./sendfir -H localhost -P 16004 ’msg=testmessage’
Sending Event to server ’localhost:16004’ Event sent successfully!
```

This example sends an event with the slot `msg` to `testmessage` to the server listening on port `16004` on the local machine.

d_v2fmtfilt and d_tecfmtfilt

These two binaries implement debug filters to test format files. They behave exactly like the components without the leading `d_` , but don t send any events to following components. The events are written into a file `v2fmtiflt.fir` respective `tecfmtfilt.fir` instead.

The debug filters can be used by altering the filters run! statement to start the debug filter instead of the normal binary.

```
v2fmtfilt=run!d_v2fmtfilt -P 11005 -AB 127.0.0.1,port!11005,targets!msgc
lsfsrv,formatlist!fn_log; fn_log.v2s,conf!port;run;targets;formatlist
```

Example F-2. An example configuration line for d_v2fmtfilt

Appendix G. CALA Configurator

CALA Configurator Basics

The CALA configurator is used to install and configure complex CALA configurations based on small (partial) CALA configuration exports (exported by CALA GUI). These small CALA configuration exports define all necessary parameter settings of one secondary date type. The CALA configurator is used by distribution via Tivoli ACP or during manual CALA installation procedures.

Supported components

The CALA Configurator supports all CALA components except for calaproxy.

Standard architectures

Templates for the following standard architectures are included in the Plus Module:

	reader	**msg**	**emitter**	**remote**	**template**
CFG_1	X			X	`logctlsrv.rdr_rem.templ`
CFG_2	X	X		X	`logctlsrv.rdr_msg_rem.templ`
CFG_3	X		X		`logctlsrv.rdr_tec.templ`
CFG_4	X	X	X		`logctlsrv.rdr_msg_tec.templ`
CFG_5		X	X		`logctlsrv.msg_tec.templ`
CFG_6		X		X	`logctlsrv.msg_rem.templ`
CFG_7			X		`logctlsrv.tec.templ`

"Reader" means ascfileread / ntevtlogread + filter and/or calamon and/or snmpread and/or the db readers.

"Emitter" means tecfmtemit + tecifcsrv and/or snmpemit and/or smtpemit and/or mysqlemit and/or cmdemit and/or reportemit.

The readers can send either to a remote component (CFG_1), to msgclsfsrv or to an emitter (CFG_3). If they send to msgclsfsrv, the target can either be a remote component (CFG_2) or an emitter (CFG_4).

In addition, it is possible to create configurations without readers. These configurations correspond to the remote component in CFG_1. The possible combinations include msgclsfsrv and an emitter on one machine (CFG_5), msgclsfsrv sending to a remote component (CFG_6) or an emitter (CFG_7). The single emitter is the remote component for CFG_2 and CFG_6.

You can create your own templates if you need different combinations of components.

Restrictions

The usage of CALA Configurator implies some restrictions:

General

- only text files can be transferred by the upcall / downcall mechanism; binary files will be broken
- text files are converted to UNIX format regardless of the target platform. CALA and its related applications handle UNIX format correctly even if started on Windows platforms. If you want to edit files on the client, you should use an editor that supports UNIX format.
- pre-filter, format files, and map files must be named according to the naming convention described below
- all parts of names covered by the naming convention must be given in lowercase (e.g. if you define a secondary data type named ORACLE, the corresponding map file must be called ORACLE_attrib.map, otherwise it will not be recognized)
- names of secondary data types may consist of letters, numbers, and underscores only. The first character must be a letter or number. Secondary data types must not start with the following prefixes:
  - aux_
  - calamon_
  - completer_
  - javasrv_
  - remapper_
  - remote_
  - report_
  - tec_

ascfileread/ntevtlogread and tecfmtfilt/v2fmtfilt

- reader and corresponding filter must always run on the same machine

calamon

- only scripts with the following extensions are detected and distributed automatically:
  - .pl
  - .sh
  - .bat
  - .cmd

- only scripts and commandtables with relative path names are detected and distributed automatically:

TEC interface

- tecfmtemit and tecifcsrv must always run on the same machine

Templates

The CALA Configurator creates the client configuration based on a template file which is included in the distribution. The Plus Module already contains some default templates. These templates reflect the supported architectures listed above. In addition, a complete template is included that contains all valid components.

The templates are located in the directory `$DBDIR/TME/PLUS/CALA/calacfg/_templ`.

Creating your own templates

You can create as many additional templates as you need. This enables you to specify your own default settings for the components that are installed (different port numbers, debug settings etc.).

All templates names must correspond to the pattern `logctlsrv.<variable part>.templ`. The variable part can be used for your own naming convention similar to the names of the template files. Make sure to include the `.` (dot) before and after the variable part of the name.

Templates must observe the following rules:

- if the template contains ascfileread or ntevtlogread, at least one filter must be specified
- if one of the filters is specified, ascfileread or ntevtlogread must be included, too
- if the template contains tecfmtemit, tecifcsrv must be included and vice versa
- the template must contain a target definition, either tecifcsrv or a remote component

Directory structure in the export directory of CALAGUI

The directory structure shown in the table below is created in the export directory of CALAGUI.

The directories `_misc` and `_targets` are included in the CALAGUI installation image because they are needed for all exports.

The subdirectories for the secondary data types are created depending on the data types you select during the export process (see *Configuration GUI*, section *Exporting parts of the configuration for use with the CALA configurator*). During the export for secondary data types, the corresponding `.cala` fie is created and all files that match the naming convention (format files, map files...) are copied to the `export` directory for this data type. This means that after the export, the `<sec_dt>` subdirectories contain all files related to the secondary data type.

Directory	Contents
`_misc`	definitions for auxkeys, completers, remappers and calamon. (`aux_*.cala`, `completer_*.cala`, `remapper_*.cala`, `calamon_*.cala`).
`_targets`	definitions for TEC interface server and remote components (`tec_*.cala`, `remote_*.cala`)

Directory	Contents
`<sec_dt>`	subdirectory per secondary data type, created by export from CALAGUI

Directory structure on each Tivoli server

All directories that are related to the CALA Configurator are created in a subdirectory of the Plus Module installation path. This "root directory" for the CALA Configurator is named `$DBDIR/TME/PLUS/CALA/calacfg`.

The "root" directory calacfg and all subdirectories starting with _ (underscore) are created by the installation of the CALA Plus Module.

The following table shows the directory structure below calacfg:

Directory	Contents
`_keys`	keyfiles for encryption
`_logs`	logfiles from distribution (`caladist.<EP-name>.<yyyymmddhhmm>.log`)
`_misc`	definitions for auxkeys, completers, remappers, javasrv and calamon. (`aux_*.cala`, `completer_*.cala`, `remapper_*.cala`, `javasrv_*.cala`, `calamon_*.cala`).
`_targets`	definitions for TEC interface server and remote components (`tec_*.cala`, `remote_*.cala`)
`_templ`	template files for configurations (`logctlsrv.*.templ`) The standard templates are copied here during module installation.
`<sec_dt>`	subdirectory per secondary data type, created by export from CALAGUI

Synchronizing the Configurator repository

The synchronization consists of two steps. The first step must be performed manually, the second step is automated.

The repository on the TMR server is the "main" repository. The list boxes for the tasks `Generate profile for CALACFG` and `Remove secondary data type from configuration` are based on the information found in this repository. This means that you can only select definitions that are located in a subdirectory of `$DBDIR/TME/PLUS/CALA/calacfg` on your TMR server.

Step 1: Synchronizing CALAGUI and TMR server

To synchronize your CALAGUI installation(s) and the TMR server, you must copy the complete directory structure that is located in the export directory of CALAGUI to your TMR server. The directories and files must be created in the directory `$DBDIR/TME/PLUS/CALA/calacfg`.

The easiest way to achieve this is to create a tar file containing the data located in the export directory, transfer the tar file to the TMR server and unpack it into the calacfg directory.

You can merge data from different CALAGUI installations. In this case, make sure that you use unique names for the files located in the `_targets` and `_misc` directories. You should also keep in mind that a secondary data type can be contained in several export directories from several machines and that any changes you made may be overwritten if you copy the files from another machine.

Step 2: Synchronizing TMR server and Gateways

This step is performed automatically by the task `Generate profile for CALACFG`. A tar file is created from the repository and sent to all Gateways. The Gateways unpack the tar file to `$DBDIR/TME/PLUS/CALA/calacfg`. The different synchronization modes are described within the description of the task `Generate profile for CALACFG`.

Where to put files referenced from within a configuration

There are several locations where you can put the files referenced from within a configuration:

- In the directory where configuration is saved from CALAGUI

If you put the files into the same directory as the configuration before performing an "export", all files that match the naming convention will be copied together with the corresponding .cala file.

- In the repository on the TMR server

You can create the files directly in the appropriate directory in the Configurator repository on the TMR server, e.g. if you want to keep your format files in a centralized place.

- In the repository on the Gateway(s)

Any file needed by the Configurator will be searched on the Gateway that hosts the Endpoint on which the Configurator runs. Changes made to the files on a Gateway will only be distributed to Endpoints hosted by this Gateway.

- On the clients

The installation process checks the subdirectories `custom` and `fmt` (for format files) / `misc` (for all other files). So if you have a format file that is needed by one client only (e.g. because only this client writes a specific logfile format), you can create the corresponding file directly in one these subdirectories. See following chapter for details.

Directory structure on client

The directory structure on the client is created during distribution of CALA to the client.

All directories that are related to the CALA Configurator are created in a subdirectory of the CALA installation path. This "root directory" for the CALA Configurator is named `$LCF_BINDIR/../CALA/adp_bin`.

The following table lists the directories that are created below adp_bin:

Directory	Contents

Directory	Contents
custom	repository for customer files All files referenced from within configuration file are searched in this directory first.
fmt	format files (*.fmt, *.v2s)
lastcfg	backup of last working configuration including logctlsrv.conf and the subdirectories repos, fmt and misc
misc	all remaining files referenced from within configuration (*.map, *.flt, ...)
repos	input files for CALA Configurator (*.cala and current template)
temp	working directory

Starting the Configurator

The Configurator is implemented in the standard CALA installation routines. These installation scripts are described in chapter *Module Configuration*, section *Manual CALA installation.*

Input files (.cala files)

The following chapter describes all valid configuration entries for the input files.

The input files can be created using the CALAGUI. You can generate the .cala files for the currently loaded configuration file by selecting Configuration⟶Export .cala files from the menu.

General parameters

These parameters can be specified in input files for secondary data types and calamon (<sec_dt>.cala, calamon_*.cala). They affect the handling of the additional files that are needed for the secondary data type that is described in the respective input file.

Fieldname	Contents
SERVER_PATH	path where to look for referenced files SERVER_PATH can be full path or path relative to $DBDIR/TME/PLUS/CALA/calacfg. Default is $DBDIR/TME/PLUS/CALA/calacfg/<sec_dt>
OVERWRITE_LOCAL	If this switch is set to Y, the referenced files are always requested from the Gateway, even if they are already located on the client. The files on the client are overwritten.

<sec_dt>.cala

The input files for secondary data types can contain definitions for several components.

The definitions per component are merged into one configuration string. So if there is an input file named oracle.cala with definitions for an Oracle logfile and an input file named solaris_syslog.cala that contains the parameters for the Solaris syslog file, both logfiles will be handled by the same instance of `ascfileread` because the logfile references will be merged into one `pathlist` and `ptrnlist`. See description of configuration below for details.

You can include any number of input files for secondary data types in your configuration.

ascfileread

Fieldname	Contents
`LOG_FILE_n`	name of logfile
`LOG_PATH_n`	path to logfile
`LOG_PRI_TYPE_n`	primary data type (`tec` / `v2`)

ntevtlogread

Fieldname	Contents
`EVT_LOG_n`	name of event log
`EVT_FILT_IN_n`	prefilt_in required `Y/N`; optional, default is `N`
`EVT_FILT_OUT_n`	prefilt_out required `Y/N`; optional, default is `N`
`EVT_PRI_TYPE_n`	primary data type (`tec` / `v2`)
`EVT_SKIP_OLD_n`	`0`=off, `1`=on; optional, default is `0`
`EVT_SPACE_REPL_n`	`0`=off, `1`=on; optional, default is `1`

snmpread

Fieldname	Contents
`SNMP_PRI_TYPE`	primary data type (`tec` / `v2`)
`SNMP_CLASS`	event class for SNMP events; optional, default is `CALA_SNMP`
`SNMP_FILT_IN`	prefilt_in required `Y/N`; optional, default is `N`
`SNMP_FILT_OUT`	prefilt_out required `Y/N`; optional, default is `N`

mssqlread, oracleread

Fieldname	Contents
`DB_TYPE_n`	database type (`oracle` / `mssql` / `jdbc`)
`DB_NAME_n`	database name
`DB_NAME_REMOTE_n`	`mssql`: remote server name; `oracle`: global database name
`DB_USER_n`	database user

Fieldname	Contents
DB_PASSWORD_n	encrypted database password
DB_TABLE_n	table
DB_ENTRY_ID_n	column for entry identification, format: <idfield>;<order>
DB_MAP_n	mapping between database fields and slot names, format: <dbfield>;<firfield>; may be specified more than once
DB_COPY_UNMAPPED_n	copy unmapped database fields 0 (no) / 1 (yes); optional, default is 1
DB_CLASS_n	class name; optional, default is MSSQLREAD_Base / ORACLE_Base
DB_CLASSMAP_n	map file and database field for class mapping, format: <map_file>;<dbfield>
DB_FILT_IN_n	prefilt_in required Y/N, optional, default is N
DB_FILT_OUT_n	prefilt_out required Y/N, optional, default is N
DB_PRI_TYPE_n	primary data type (tec / v2)
DB_TIMESTAMP_n	field definition(s) for timestamp; format 1: <db-field>, format 2: <date-part-id >;<db-field>;<text-position>; entries for format 2 may be specified more than once
DB_POLLINTERVAL_n	seconds between read operations on database;optional, default is 10
DB_DRIVERJAR_n	full-qualified name of the JDBC driver jarfile if DB_TYPE_n is jdbc; this information will be added to the run! statement of **jdbcread**

reportemit (datatype specific definitions)

These fields can be specified once per secondary datatype to create a datatype specific report. To create a default report for all datatypes, a report_*.cala files can be used (see description below).

Fieldname	Contents
REP_PRI_TYPE	primary data type (tec / v2)
REP_CRITICAL_SLOT	critical slot, format: <field>[;<value>]
REP_TEMPL	name of template file or DEFAULT; optional
REP_FILE	name of report file

msgclsfsrv

Fieldname	Contents
MAP_NAME_n	suffix for map filename

Fieldname	Contents
MAP_PRI_TYPE_n	primary data type (tec / v2)
MAP_TARGET_n	reference to a TARGET entry
RULE_NAME_n	suffix for rule filename
RULE_PRI_TYPE_n	primary data type (tec / v2)
RULE_TARGET_n	reference to a TARGET entry
RULE_CORRKEY_n	key(s) for correlation

cmdemit, mysqlemit, reportemit, smtpemit, snmpemit, tecfmtemit, remote_component

These are the definitions for all valid targets for a datatype. The actual targets result from the combination of the TARGET definition and the template that is used for configuration.

The TARGET_n entries are referenced by the MAP_TARGET_n and RULE_TARGET_n entries to define which maps and rules must be applied to the data stream that is sent to a specific target. If no MAPs and RULEs are defined, all specified TARGETs that have an entry in the template will be configured.

Fieldname	Contents
TARGET_n	name of target (internal name, char 1-7 are relevant) or name of remote target (remote_<suffix>)
TARGET_EVENT_FRAME_n	eventframe for dup detect for this target
TARGET_DUPEKEY_n	dupekey for dup detect for this target
TARGET_REQUIRES_n	additional .cala files required for this target (completer / remapper / auxkey definitions)

The targets cmdemit, mysqlemit, smtpemit and snmpemit require no further settings or additional .cala files.

For reportemit, a report_*.cala and/or datatype specific report definitions must be specified. For tecfmtemit, a tec_*.cala file is required to configure the tecifcsrv component. To configure a remote target, a remote_*.cala file is required that has the name specified in the TARGET_n entry (e.g. TARGET_1=remote_snmpemit means that remote_snmpemit.cala is required).

Only those targets will be configured that have an entry in the used template. To configure remote targets, the template must contain an entry that has an statement. In this case, all remote components that are referenced in a TARGET_n entry will be configured if the corresponding remote_*.cala file is available.

aux_*.cala

You can include any number of input files for auxkeys in your configuration.

Fieldname	Contents

Fieldname	Contents
AUX_NAME_n	auxkey name
<auxkeyname>_n	auxkey definition, format: <field>;<operator>
	Example: AUX_NAME_1=errcode1 errcode1_1=$CLASS;L errcode1_2=errcode;N AUX_NAME_2=location location_1=$CLASS;L location_2=location;L

completer_*.cala

You can include any number of input files for completers in your configuration.

Fieldname	Contents
COMPLETER_NAME_n	completer name
FOR_n_<completername>	component for which the completer must be applied
FILL_n_<completername>	value assignment; format: <slot>;<value>
UNLESS_n_<completername>	slot non-existent condition; format: <slot>
IF_n_<completername>	slot existent condition; format: <slot>
	Example: COMPLETER_NAME_1=report_cpl FOR_1_report_cpl=reportemit FILL_1_report_cpl=report_flag;1 FILL_2_report_cpl=secondary_flag;2 IF_1_report_cpl=sub_source COMPLETER_NAME_2=generalcpl1 FOR_1_generalcpl1=reportemit FILL_1_generalcpl1=source;CALALOGS UNLESS_1_generalcpl1=source UNLESS_2_generalcpl1=sub_source

remapper_*.cala

You can include any number of input files for remappers in your configuration.

Fieldname	Contents
REMAPPER_NAME_n	remapper name
FOR_n_<remappername>	component for which the remapper must be applied
FIELD_n_<remappername>	field to rename; format: <old>;<new>
CLASS_n_<remappername>	class to rename; format: <old>;<new>

Fieldname	Contents
	Example: REMAPPER_NAME_1=smtpemit_remap FOR_1_smtpemit_remap=smtpemit FIELD_1_smtpemit_remap=msg;MSGBODY FIELD_2_smtpemit_remap=class;KLASSE_EY REMAPPER_NAME_2=Remapclass FOR_1_Remapclass=smtpemit CLASS_1_Remapclass=Logfile;CALA_Logfile

calamon_*.cala

You can include any number of input files for calamon in your configuration.

Fieldname	Contents
`CMDTAB`	name of command table

javasrv_<logical_name>.cala

You can include any number of input files for javasrv components in your configuration. The <logical_name> must match the logical name for the corresponding component in the template file.

Fieldname	Contents
`XMLCONF`	name of configuration file

report_*.cala

You can include exactly one input file for reportemit in your configuration to define a standard report format for all datatypes. report_*.cala can be used in combination with REP_ entries in the <sec_dt>.cala files.

Fieldname	Contents
`REP_DEF_DEST_FILE`	name of report file
`REP_DEF_SLOT`	names of slots to include in report; format: `<slot>`; can be specified more than once
`REP_DEF_CRITICAL_SLOT`	critical slot, format: `<field>[;<value>]`

tec_*.cala

You can include exactly one input file for tecifcsrv in your configuration.

Fieldname	Contents
`TEC_SRV`	name of TEC servers
`TEC_PORT`	port of TEC server

remote_*.cala

You can include any number of input files for remote components in your configuration. For each `remote_<suffix>.cala` file, a remote component named remote_<suffix>__n for each REMOTE_IP / REMOTE_PORT pair will be included in the configuration if the used template has a remote_ entry and if at least one TARGET_n entry for this remote name is found. The data will be sent to all targets defined in this `remote_<suffix>.cala` file.

Fieldname	Contents
`REMOTE_IP_n`	IP address or hostname of remote component
`REMOTE_PORT_n`	port of remote component

Referenced files

Naming convention

CALA Configurator uses naming conventions for most files referenced from within the generated configuration. This means that the input files (except for `calamon_*.cala` and `javasrv_*.cala`) do not contain any file references which simplifies manual creation of input files.

Type of file	Naming pattern
format file	`<sec_dt>.fmt` or `<sec_dt>.v2s`
message map files	`<sec_dt>_<suffix>.map`
rules map files	`<sec_dt>_<suffix>.rmp`
prefilter for ntevtlogread	`<evtlogname>_in.flt` `<evtlogname>_out.flt`
prefilter for mssqlread / oracleread	`<dbname>_<tablename>_in.flt` `<dbname>_<tablename>_out.flt`
prefilter for snmpread	`_snmp_in.flt` `_snmp_out.flt`

Standard location

The standard location on the server is the same for all files listed in the table above `$DBDIR/TME/PLUS/CALA/calacfg/<sec_dt>`.

On the client, all files are located in the directory `$LCF_BINDIR/../CALA/adp_bin/misc` except for the format files which are located in `$LCF_BINDIR/../CALA/adp_bin/fmt`.

Details

Detailed description of configuration

The following chapter describes the connection between the parameters in the input files and the configuration entries that are generated by the CALA Configurator.

ascfileread

pathlist!<Number>;<Path>{;<Number>;<Path>}

is generated from all `LOG_PATH_n` entries from all `.cala` files. The <Number> in the pathlist entry does not correspond to the number `n` that is found in the `.cala` file as this number is not unique if more than one `.cala` file is processed.

ptrnlist!<Number>;<Pattern match>{;<Number>;<Pattern match>}

is generated from all `LOG_FILE_n` entries from all `.cala` files. The <Number> in the ptrnlist entry does not correspond to the number `n` that is found in the `.cala` file as this number is not unique if more than one `.cala` file is processed.

assoc!<path1istX>;<ptrn1istX>;<primary type>;<secondary type>{;<path1istX>;<ptrn1istX>;<primary type>;<secondary type>}

pathlistX and ptrnlistX result from the numbers generated for each `LOG_FILE_n` and `LOG_PATH_n` entry in the pathlist and ptrnlist statement. The primary type is given in the corresponding `LOG_PRI_TYPE_n` entry. The secondary type can be derived from the name of the input file.

targets!<target component>{;<target component>}

The targets list is generated depending on the `LOG_PRI_TYPE_n` entries found in the input files. If at least one `tec` entry is found, `tecfmtfilt` is added to the `targets` list. If at least one `v2` entry is found, `v2fmtfilt` is added to the `targets` list.

ntevtlogread

evtlog!<numeric id>;<logfile id>{;<numeric id>;<logfile id>}

is generated from all `EVT_LOG_n` entries from all `.cala` files. The <numeric id> in the evtlog entry does not correspond to the number `n` that is found in the `.cala` file as this number is not unique if more than one `.cala` file is processed.

spacereplacement!<numeric id>;<value>{;<numeric id>;<value>}

numeric id results from the numbers generated for each `EVT_LOG_n` entry in the `evtlog` statement. value is the value given in the corresponding `EVT_SPACE_REPL_n` entry or `1` if `EVT_SPACE_REPL_n` is not specified.

skip_old!<numeric id>;<value>{;<numeric id>;<value>}

numeric id results from the numbers generated for each `EVT_LOG_n` entry in the `evtlog` statement. value is the value given in the corresponding `EVT_SKIP_OLD_n` entry or `0` if `EVT_SKIP_OLD_n` is not specified. Additionally, `ntevtlogread` is called with the command line parameter *`-E`*.

prefilt_in!<numeric id>;<filter_file>{;<numeric id>;<filter_file>}

prefilt_out!<numeric id>;<filter_file>{;<numeric id>;<filter_file>}

If an entry `EVT_FILT_IN_n=Y` is found, a prefilt_in statement is generated for the corresponding event log. The <numeric id> corresponds to the number associated to the event log in the `evtlog` statement.

The same applies if an entry `EVT_FILT_OUT_n=Y` is found. In this case, a prefilt_out statement will be generated.

assoc!<numeric id>;<primary type>;<secondary type>{;<numeric id>;<primary type>;<secondary type>}

numeric id results from the numbers generated for each `EVT_LOG_n` entry in the `evtlog` statement. The primary type is given in the corresponding `EVT_PRI_TYPE_n` entry. The secondary type can be derived from the name of the input file.

targets!<target component>{;<target component>}

The targets list is generated depending on the `EVT_PRI_TYPE_n` entries found in the input files. If at least one `tec` entry is found, `tecfmtfilt` is added to the targets list. If at least one `v2` entry is found, `v2fmtfilt` is added to the `targets` list.

tecfmtfilt / v2fmtfilt

The filters are configured depending on the primary data types used by `ascfileread` and `ntevtlogread`. If only one primary data type is used by the readers, only the corresponding filter will be configured.

formatlist!<secondary type>;<name of fmt file>{;<secondary type>;<name of fmt file>}

The formatlist is generated based on the secondary data types and the corresponding format files.

targets!<target component>{;<target component>}

The generated targets definition depends on the settings in the input files and the template.

If at least one .cala file contains a `MAP` or `RULE` entry and the template contains an entry for `msgclsfsrv`, a `msgclsfsrv` is configured.

If no `MAP`s and `RULE`s are defined or if the template does not allow a `msgclsfsrv`, the `TARGET_n` entries in all `.cala` files will be checked. All referenced targets that have an entry in the used template will be added to the target list.

calamon

cmdtab!<name of command table file>

The name of the command table is always set to `cmdtab_merged.ctb`. This allows processing of more than one `calamon_*.cala`. The command tables specified in the input files are merged into one file with the given name.

targets!<target component>{;<target component>}

The generated targets definition depends on the settings in the input files and the template.

If at least one `.cala` file contains a `MAP` or `RULE` entry and the template contains an entry for `msgclsfsrv`, a `msgclsfsrv` is configured.

If no `MAPs` and `RULEs` are defined or if the template does not allow a `msgclsfsrv`, the `TARGET_n` entries in all `.cala` files will be checked. All referenced targets that have an entry in the used template will be added to the target list.

javasrv

xmlconf!<name of configuration file>

The name of the configuration file is specified in the `javasrv_*.cala` file that corresponds to the logical name of the component. If no `javasrv_*.cala` file is found for the logical name, the `javasrv` component will be removed from the configuration.

targets!<target component>{;<target component>}

The generated targets definition depends on the settings in the input files and the template.

If at least one `.cala` file contains a `MAP` or `RULE` entry and the template contains an entry for `msgclsfsrv`, a `msgclsfsrv` is configured.

If no `MAPs` and `RULEs` are defined or if the template does not allow a `msgclsfsrv`, the `TARGET_n` entries in all `.cala` files will be checked. All referenced targets that have an entry in the used template will be added to the target list.

snmpread

type!<primary type>;<secondary type>

The primary type is specified in the `SNMP_PRI_TYPE` entry. The secondary type is derived from the name of the input file.

class!<class name>

The class name is specified in the `SNMP_CLASS` entry.

This entry is optional. If it is not specified, no `class!` statement will be generated and the default class `CALA_SNMP` will be used for the events generated by `snmpread`.

prefilt_in!<filter_file>

prefilt_out!<filter_file>

If an entry `SNMP_FILT_IN=Y` is found, a prefilt_in statement is generated. The same applies if an entry `SNMP_FILT_OUT=Y` is found. In this case, a prefilt_out statement will be generated.

targets!<target component>{;<target component>}

The generated targets definition depends on the settings in the input files and the template.

If at least one `.cala` file contains a `MAP` or `RULE` entry and the template contains an entry for `msgclsfsrv`, a `msgclsfsrv` is configured.

If no MAPs and RULEs are defined or if the template does not allow a msgclsfsrv, the TARGET_n entries in all .cala files will be checked. All referenced targets that have an entry in the used template will be added to the target list.

mssqlread / oracleread

-D <db-host>

For mssqlread, the first entry found for DB_NAME_REMOTE_n will be added as db-host.

For oracleread, this option is not supported yet.

targets!<target component>{;<target component>}

The generated targets definition depends on the settings in the input files and the template.

If at least one .cala file contains a MAP or RULE entry and the template contains an entry for msgclsfsrv, a msgclsfsrv is configured.

If no MAPs and RULEs are defined or if the template does not allow a msgclsfsrv, the TARGET_n entries in all .cala files will be checked. All referenced targets that have an entry in the used template will be added to the target list.

db_log_types!<type>{;<type>}

For each block of DB_ entries in all .cala files, a type entry in the format db_<database>_<table> is generated.

db_log_types

For each entry in the db_log_types! statement, a db_log_type is generated.

<db_log_type-name>

The db_log_type-name is generated as described in the db_log_types! statement above.

db_user!<user>[;<password>]

The db_user statement contains the user name specified in the DB_USER_n entry and the encrypted password from the corresponding DB_PASSWORD_n entry. If DB_USER_n is not specified, no db_user entry is created for this db_log_type.

database!<database-name>

The database name contains the name specified in the DB_NAME_n entry.

table!<table-name>

The table name contains the name specified in the DB_TABLE_n entry.

db_entry_id!<id-field>;<DESC/ASCE>

The contents of the db_entry_id field is copied directly from the DB_ENTRY_ID_n field.

map!<db-field>;<fir-field>{;<db-field>;<fir-field>}

The map entry is generated from all DB_MAP_n entries for this db_log_type.

copy_unmapped!<0/1>

The value for copy_unmapped is taken from the DB_COPY_UNMAPPED_n entry.

defaultclass!<class-name>

The class name for the defaultclass entry is specified in the DB_CLASS_n entry.

classmap!<map-file>;<db-field>

The contents of the classmap field is copied directly from the `DB_CLASSMAP_n` field.

type!<primary type>;<secondary type>

The primary type is specified in the `DB_PRI_TYPE_n` entry. The secondary type is derived from the name of the input file.

prefilt_in!<filter_file>

prefilt_out!<filter_file>

If an entry `DB_FILT_IN_n=Y` is found, a prefilt_in statement is generated. The same applies if an entry `DB_FILT_OUT_n=Y` is found. In this case, a prefilt_out statement will be generated.

timestamp!<db-field>

timestamp!<date-part-id >;<db-field>;<text-position>{;<date-part-id >;<db-field>;<text-position>}

The timestamp field contains all `DB_TIMESTAMP_n` entries specified for this `db_log_type`.

pollinterval!<interval>

The pollinterval is copied from the `DB_POLLINTERVAL_n` entry.

msgclsfsrv

targets!<target component>{,<target component>}

The generated targets definition depends on the settings in the input files and the template.

The `MAP_TARGET_n` and `RULE_TARGET_n` entries in all `.cala` files will be checked. All referenced targets that have an entry in the used template will be added to the target list.

types!<mct-name>{;<mct-name>}

For each `MAP_` entry in all `.cala` files that reference the same component in their `MAP_TARGET_n` entry, an mct_name in the format `<sec_dt>_<pri_dt>_<target>_mct` is generated.

rules!<rmt name>{;<rmt_name>}

For each `RULE_` entry in all `.cala` files, an rmt_name in the format `<sec_dt>_<pri_dt>_rmt` is generated.

completers!<completer name>{;<completer name>}

The list of completer names is generated from the completer names found in the `completer_*.cala` files. Only those completers will be included whose `FOR_` entries references a target that is included in the actual configuration.

If there are duplicate names, the configuration is cancelled.

remappers!<remapper name>{;<remapper name>}

The list of remapper names is generated from the remapper names found in the `remapper_*.cala` files. Only those remappers will be included whose `FOR_` entries references a target that is included in the actual configuration.

If there are duplicate names, the configuration is cancelled.

auxkeys!<key name>{;<key name>}

The list of key names is generated from all auxkey names found in the `aux_*.cala` files.

If there are duplicate names, the configuration is cancelled.

MCT

For each entry in the types! statement, an MCT is generated.

<mct-name>

The mct-name is generated as described in the types! statement above.

types!<primary type;secondary type>{;<primary type;secondary type>}

The primary type is the value specified in the corresponding `MAP_PRI_TYPE_n` entry. The secondary type is derived from the name of the `.cala` file where the `MAP_PRI_TYPE_n` entry was found.

handledby!<emitter name>{;<emitter name>}

The emitter name is the name of the target referenced in the `MAP_TARGET_n` entry. All maps for a datatype that must be applied for the same emitter are combined into one MCT.

msgmaps!<msgmap filename;logical name>{;<msgmap filename>;<logical name>}

For each `MAP_NAME_n` entry, a corresponding msgmap entry will be generated where msgmap filename consists of the secondary datatype and the suffix given in `MAP_NAME_n`. The logical name is the same as the filename but the extension `.map` is replaced by the suffix `_map` . The sequence of the message map entries is the same as the sequence of the `MAP_` entries in the input file.

For each of the logical names a corresponding Message Map entry is generated.

eventframe!<seconds>

If the target referenced in the handledby! instruction has a corresponding `TARGET_EVENT_FRAME_n` entry, the corresponding value is used here. Otherwise the event frame statement is left out and the default (3600 seconds) is used.

dupekey!<field name;text position>{;<field name;text position>}

If the target referenced in the handledby! instruction has corresponding `TARGET_DUPEKEY_n` entries, the values are listed here. The sequence is the same as in the input file.

MessageMap entry

<log. msgmap name>=

The log. msgmap name is generated as described in the msgmaps! statement above.

The information required to generate the message map entries must be given as comment in the corresponding message map file.

key!<key name>;<text position>{;<key name>;<text position>}

For each key a comment line with the following structure must be given in message map file:

#key keyname <operator>

fields!<field>{;<field>}

For each field a comment line with the following structure must be given in message map file:

#field <fieldname>

RMT

For each entry in the rules! statement, an RMT is generated.

<rmt-name>

The rmt-name is generated as described in the rules! statement above.

for!<emitter name>{;<emitter name>}

The emitter name is the name of the target referenced in the `RULE_TARGET_n` entry for this rule.

type!<primary type;secondary type>

The primary type is the value specified in the corresponding `RULE_PRI_TYPE_n` entry. The secondary type is derived from the name of the .cala file where the `RULE_PRI_TYPE_n` entry was found.

rulesmaps!<rulesmap filename;logical name>{;<rulesmap filename>;<logical name>}

For each `RULE_NAME_n` entry, a corresponding rulesmap entry will be generated where rulesmap filename consists of the secondary datatype and the suffix given in `RULE_NAME_n`. The logical name is the same as the filename but the extension `.rmp` is replaced by the suffix `_rmp`. The sequence of the rules map entries is the same as the sequence of the `RULES_` entries in the input file.

For each of the logical names a corresponding Rules Map entry is generated.

corrkey!<field;text position>{;<field;text position>}

If the `RULE_n` statement has corresponding `RULE_CORRKEY_n` entries, the values are listed here. The sequence is the same as in the input file.

RulesMap entry

<log. rulesmap name>=

The log.rulesmap name is generated as described in the `rulesmaps!` statement above.

The information required to generate the rules map entries must be given as comment in the corresponding rules map file.

key!<field>;<text position>{;<field>;<text position>}

For each key a comment line with the following structure must be given in a rules map file:

#key <keyname> <operator>

conditions!<field>{;<field>}

For each condition a comment line with the following structure must be given in a rules map file:

#condition <conditionname>

fields!<field>{;<field>}

For each field a comment line with the following structure must be given in a rules map file:

#field <fieldname>

Auxkey entry

First, all `AUX_NAME_n` in all `aux_*.cala` files are checked if they are unique. If any duplicates are found, the configuration is cancelled.

<key name>=

For each `AUX_NAME_n` entry a key name is generated.

<field>;<text position>{;<field>;<text position>}

All entries that are found for the `AUX_NAME_n` entry are put together to form the auxkey definition. The sequence is the same as in the input file.

Completer entry

First, all `COMPLETER_NAME_n` in all `completer_*.cala` files are checked if they are unique. If any duplicates are found, the configuration is cancelled.

<completer name>=

For each `COMPLETER_NAME_n` entry a completer name is generated.

for!<emitter name>{;<emitter name>}

All `FOR_` entries that are found for the `COMPLETER_NAME_n` entry and that reference targets in the current configuration are put together to form the for definition.

fill!<slot name;value>{;<slot name;value>}

All `FILL_` entries that are found for the `COMPLETER_NAME_n` entry are put together to form the fill definition. The sequence is the same as in the input file.

unless!<slot name>{;<slot name>}

All `UNLESS_` entries that are found for the `COMPLETER_NAME_n` entry are put together to form the unless definition. The sequence is the same as in the input file.

if!<slot name>{;<slot name>}

All `IF_` entries that are found for the `COMPLETER_NAME_n` entry are put together to form the if definition. The sequence is the same as in the input file.

Remapper entry

First, all `REMAPPER_NAME_n` in all `remapper_*.cala` files are checked if they are unique. If any duplicates are found, the configuration is cancelled.

<remapper name>=

For each `REMAPPER_NAME_n` entry a remapper name is generated.

for!<emitter name>{;<emitter name>}

All `FOR_` entries that are found for the `REMAPPER_NAME_n` entry and that reference targets in the current configuration are put together to form the for definition.

fieldalias!<old field name;new field name>{;<old field name;new field name>}

All `FIELD_` entries that are found for the `REMAPPER_NAME_n` entry are put together to form the `fieldalias` definition. The sequence is the same as in the input file.

classalias!<old class name;new class name>{;<old class name;new class name>}

All `CLASS_` entries that are found for the `REMAPPER_NAME_n` entry are put together to form the `classalias` definition. The sequence is the same as in the input file.

tecfmtemit

targets!<target component>{;<target component>}

The only valid target is `tecifcsrv`.

tecifsrv

Only `tecifcsrvend` and `tecifcsrvuns` are supported. If the value specified for `TEC_SRV` contains `@EventServer`, `tecifcsrvend` is configured, otherwise `tecifcsrvuns` will be used.

-h <hostname>

The `hostname` is the value specified for `TEC_SRV` in the `tec_*.cala` input file.

-p <port no.>

If the `tec_*.cala` input file contains a `TEC_PORT` entry, the given value will be added to the command line parameters for `tecifcsrv`.

reportemit

dest_file!<filename>

The dest_file entry uses the filename given in `REP_DEF_DEST_FILE` from the `report_*.cala` file as general report file for all datatypes.

report_slots!<field name>{;<field name>}

The report_slots instruction is generated from all `REP_DEF_SLOT` entries from the `report_*.cala` file.

critical_slot!<field name>[;<field value>]

The critical_slot entry is taken from the `REP_DEF_CRITICAL_SLOT` field specified in the `report_*.cala` file.

critical_slots!<primary type>;<secondary type>;<field name>;<field value>{;<primary type>;<secondary type>;<field name>;<field value>}

For each `REP_CRITICAL_SLOT` entry found in a .cala file for a secondary datatype, an entry in the critical_slots list is generated. The primary type is taken from the corresponding `REP_PRI_TYPE` field. The secondary type is derived from the name of the .cala file where the `REP_PRI_TYPE` entry was found.

report_file!<primary type>;<secondary type>;<template filename>;<report filename>{;<primary type>;<secondary type>;<template filename>;<report filename>}

For each `REP_FILE` entry found in a .cala file for a secondary datatype, an entry in the report_file list is generated. The primary type is taken from the corresponding `REP_PRI_TYPE` field. The secondary type is derived from the name of the .cala file where the `REP_PRI_TYPE` entry was found. If `REP_TEMPL` is specified in the .cala file, the specified file will be used as template. If `REP_TEMPL` is not specified, `DEFAULT` will be used instead.

remote components

ip!<ip address>{;<ip address>}

All IP addresses found in the `REMOTE_IP` entries in the `remote_*.cala` file are listed in the IP-statement. The sequence is the same as in the input file.

port!<port no.>{;<port no.>}

All ports found in the `REMOTE_PORT` entries in the `remote_*.cala` file are listed in the IP-statement. The sequence is the same as in the input file.

Example for .cala files, templates and the resulting configuration

`fndw4log.cala` - Definition for secondary datatype fndw4log

LOG_FILE_1=srvlink.log LOG_PATH_1=/tmp LOG_PRI_TYPE_1=v2 MAP_NAME_1=evt MAP_PRI_TYPE_1=v2 MAP_TARGET_1=TARGET_1 MAP_TARGET_1=TARGET_2 MAP_NAME_5=filter_tec MAP_PRI_TYPE_5=v2 MAP_TARGET_5=TARGET_1 # possible targets for this datatype: tec TARGET_1=tecfmtemit TARGET_REQUIRES_1=aux_fnislog.cala TARGET_REQUIRES_1=remapper_fnislog.cala # this definition supports sending to a remote msgclsfsrv only (no mapping for client provided.) TARGET_2=remote_panagon

This `.cala` file defines one logfile for the `fndw4log` datatype. Two map files are required to process the data: `fndw4log_evt.map` and `fndw4log_filter.map`. The datastream can be sent to `tecfmtemit` or to a remote component. The definition for the remote component must be given in the file `remote_panagon.cala`. To complete the datastream for `tecfmtemit`, the files `aux_fnislog.cala` and `remapper_fnislog.cala` are required.

`remapper_fnislog.cala` - Definition for remapper

REMAPPER_NAME_1=tecfmtemit_remap FIELD_1_tecfmtemit_remap=$ESCCNT;occurrences_before_send FOR_1_tecfmtemit_remap=tecfmtemit REMAPPER_NAME_2=snmpemit_remap FIELD_1_snmpemit_remap=origin;ORIGIP FIELD_2_snmpemit_remap=error_id;VALUE1 FOR_1_snmpemit_remap=snmpemit

This remapper file contains definitions for two different remappers. The first is required for `tecfmtemit`, the second for `snmpemit`.

`tec_panagon.cala` - Definition for tecifcsrv

This file is required if `tecfmtemit` (TARGET_1) should be configured as target.

TEC_SRV=@EventServer

`remote_panagon.cala` - Definition for remote component

This file is required if the remote component (TARGET_2) should be configured as target.

REMOTE_IP=ccc4.stgt.cenit.de REMOTE_PORT=11012

Template

This is a sample template.

#operating-system: __INTERP__ #name of configuration: __CFGNAME__ #user: __USER__ #password: __PASS__ serverlist=ascfileread,tecfmtfilt,v2fmtfilt,msgclsfsrv,tecfmtemit,tecifcsrv,snmpemit,remote_comp

ascfileread=run!ascfileread -E -H __HOSTNAME__ -AB 127.0.0.1 -
P 11001,port!11001,
targets!__ASC_TARGETS__,pathlist!__PATHLIST__,ptrnlist!__PTRNLIST__,
assoc!__ASC_ASSOC__,conf!port;run;targets;pathlist;ptrnlist;assoc
tecfmtfilt=run!tecfmtfilt -AB 127.0.0.1 -P 11003,port!11003,
targets!__FLT_TARGETS__,formatlist!__FMTLIST__,conf!port;run;targets;formatlist
v2fmtfilt=run!v2fmtfilt -AB 127.0.0.1 -P 11004,port!11004,
targets!__FLT_TARGETS__,formatlist!__FMTLIST__,conf!port;run;targets;formatlist
msgclsfsrv=run!msgclsfsrv -AB 127.0.0.1 -P 11009,port!11009, tar-
gets!__MSG_TARGETS__,__MCT__,__RMT__,__AUXKEYS__,__COMPLETERS__,__REMAPPERS__,
conf!port;run;targets__MSGCLSF_CONF__
tecfmtemit=run!tecfmtemit -AB 127.0.0.1 -P 11010,port!11010,
targets!tecifcsrv,conf!port;run;targets tecifcsrv=run!tecifcsrv__TEC_TYPE__ -
AB 127.0.0.1 -ZCREATE_STATUS_EVENTS=1 -P 11011
-h __TEC_SRV____TEC_PORT__,port!11011,conf!port;run
snmpemit=run!snmpemit -AB 127.0.0.1 -ZCREATE_STATUS_EVENTS=1 -
P 11012,port!11012,conf!port;run
remote_comp=ip!__REM_IP__,port!__REM_PORT__,conf!port;ip

Resulting configuration file

#operating-system: hp-ux #name of configuration: Generated by CALACFG #user:
#password:
serverlist=ascfileread,v2fmtfilt,msgclsfsrv,tecfmtemit,tecifcsrv,remote_panagon
ascfileread=run!ascfileread -E -H tivhp11i -AB 127.0.0.1 -P 11001,port!11001,
targets!v2fmtfilt,pathlist!1;/tmp,ptrnlist!1;srvlink.log,assoc!1;1;v2;fndw4log,
conf!port;run;targets;pathlist;ptrnlist;assoc
v2fmtfilt=run!v2fmtfilt -AB 127.0.0.1 -P 11004,port!11004,
targets!msgclsfsrv,formatlist!fndw4log;fmt/fndw4log.v2s,
conf!port;run;targets;formatlist
msgclsfsrv=run!msgclsfsrv -AB 127.0.0.1 -P 11009,port!11009,
targets!tecfmtemit,remote_panagon,
types!fndw4log_v2_tecfmtemit_mct;fndw4log_v2_remote_panagon,
auxkeys!aux_fnislog_0_2;aux_fnislog_0_0;aux_fnislog_0_8;aux_fnislog_0_15,
remappers!tecfmtemit_remap,conf!port;run;targets;types;auxkeys;remappers
# mct definitions
fndw4log_v2_tecfmtemit_mct=type!v2;fndw4log,handledby!tecfmtemit,
msgmaps!misc/fndw4log_evt.map;fndw4log_evt_map;
misc/fndw4log_filter_tec.map;fndw4log_filter_tec_map
fndw4log_v2_remote_panagon_mct=type!v2;fndw4log,handledby!remote_panagon,
msgmaps!misc/fndw4log_evt.map;fndw4log_evt_map; # map definitions
fndw4log_evt_map=key!error_id;L,fields!severity;msg;error_cause;corrective_action
fndw4log_filter_tec_map=key!severity;L,fields!$CLASS;severity # auxkey definitions
aux_fnislog_0_0=error_id;L aux_fnislog_0_15=error_id;L;original_error_text;F0T15
aux_fnislog_0_2=error_id;L;original_error_text;F0T2
aux_fnislog_0_8=error_id;L;original_error_text;F0T8 # remapper definitions
tecfmtemit_remap=for!tecfmtemit,fieldalias!$ESCCNT;occurrences_before_send
tecfmtemit=run!tecfmtemit -AB 127.0.0.1 -P 11010,port!11010,
targets!tecifcsrv,conf!port;run;targets
tecifcsrv=run!tecifcsrvend -AB 127.0.0.1 -ZCREATE_STATUS_EVENTS=1 -P 11011
-h @EventServer,port!11011,conf!port;run
remote_panagon=ip!ccc4.stgt.cenit.de,port!11012

Configured components

`tecfmtfilt` has been removed from the configuration because no `LOG_PRI_TYPE_n=tec` is specified.

`snmpemit` has been removed from the configuration because no corresponding `TARGET` entry was given in `fndw4log.cala`.

Both `TARGET_1` (`tecfmtemit`/`tecifcsrv`) and `TARGET_2` (`remote_panagon`) have been included in the configuration because the template contains entries for both targets. In addition, the definition file `remote_panagon.cala` has been included. If this file had been unavailable, no remote target would have been configured.

Note that the remote component is named remote_panagon according to the used definition file. The name remote_comp is used in the template only.

Configuration details of `msgclsfsrv`

There are two MCT entries in the types! statement, one for each target that the `fndw4log` datastream is sent to.

The first MCT, fndw4log_v2_tecfmtemit_mct, is used to prepare the datastream for `tecfmtemit`. It uses `fndw4log_evt.map` as well as `fndw4log_filter_tec.map` because both maps have a `MAP_TARGET_n=TARGET_1` entry which references `tecfmtemit`.

The second MCT, fndw4log_v2_remote_panagon_mct, is used to prepare the datastream for the remote component `remote_panagon`. It uses `fndw4log_evt.map` because only this map has a `MAP_TARGET_n=TARGET_2` entry which references `remote_panagon`.

The auxkey definitions are all included in the file `aux_fnislog.cala` (not shown above) which is referenced by `TARGET_REQUIRES_1`.

The configuration includes only one remapper definition, `tecfmtemit_remap`. The second definition, `snmpemit_remap`, is skipped because its `FOR_` entry references `snmpemit`. This component is not included in the current configuration so the remapper is not required.

Appendix H. A complete logctlsrv.conf

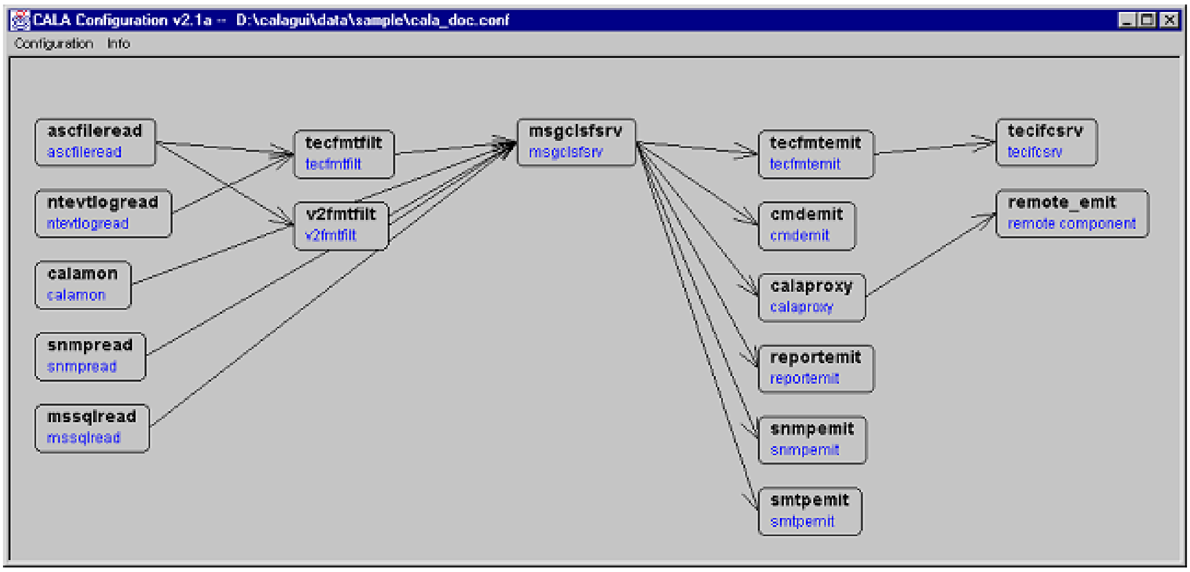
This is the complete `logctlsrv.conf` created from the CALAGUI using all available components. (See configuration samples in *Configuration GUI* and *Component-specific configuration*.

```

#operating-system: nt
#name of configuration: cala_doc

logctlsrv_port=11000
logctlcmd_port=11001
cala_srv_port=1102
maintenance=Fri 2300;Sat 0300;01 2200;02 0500

serverlist=ascfileread,ntevtlogread,calamon,snmpread,mssqlread,tecfmtfilt,v2fmtfilt,msgclsfsrv,tecfmtemit,cmdemit,calaproxy,reportemit,snmpemit,smtpemit,tecifcsrv,remote_emit

# new column
ascfileread=run!ascfileread -P 11002,port!11002,targets!tecfmtfilt;v2fmtfilt,pathlist!1;/var/adm;2;/fnsw/local/logs/elogs,ptrnlist!1;messages;2;elogYYYYMMDD,assoc!1;1;tec;solaris_syslog;2;2;v2;fn_log,conf!port;run;targets;pathlist;ptrnlist;assoc
ntevtlogread=run!ntevtlogread -P 11003,port!11003,targets!tecfmtfilt,evtlog!1;security;2;application;3;system,spacereplacement!1;1;2;1;3;1,assoc!1;tec;nt_security;2;tec;nt_application;3;tec;nt_system,skip_old!1;0;2;1;3;1,prefilt_in!2;prefilt_nt_app.flt,prefilt_out!1;prefilt_nt_sec.flt,conf!port;run;targets;evtlog;spacereplacement;assoc;skip_old;prefilt_in;prefilt_out
calamon=run!calamon -P 11006,port!11006,targets!msgclsfsrv,source!CALA,subsource!monitoring,cmdline_slot!probe_arg,cmdtab!cmdtab.tbl,conf!port;run;targets;source;subsource;cmdline_slot;cmdtab
snmpread=run!snmpread -P 11008 -p 162,port!11008,targets!msgclsfsrv,type!tec;snmp,class!SNMP_Event,conf!port;run;targets;type;class
mssqlread=run!mssqlread -P 11011 -AB 127.0.0.1,port!11011,targets!msgclsfsrv,db_log_types!log_audit_fndsdb,conf!port;run;targets;db_log_types
log_audit_fndsdb=table!AUDIT_LOG,database!fndsdb,db_entry_id!AL_DATETIME;ASCE,map!AL_PROCESSID;pid;AL_STATUS;AL_STATUS;AL_WORKSTATN_ADDR;workstation;AL_EVENT_PARAM1;msg;AL_EVENT_PARAM2;PARAM2;AL_EVENT_PARAM3;PARAM3;AL_EVENT_PARAM4;PARAM4;AL_USER;USER,copy_unmapped!0,timestamp!AL_DATETIME,type!tec;FNDS_MSSQL,defaultclass!FNDS_AUDITLOG_Error,classmap!ds_class.map;AL_EVENT_ID,pollinterval!30

# new column
tecfmtfilt=run!tecfmtfilt -P 11004,port!11004,targets!msgclsfsrv,formatlist!solaris_syslog;solaris_syslog.fmt;nt_system;nt_system.fmt;nt_application;nt_application.fmt;nt_security;nt_security.fmt,conf!port;run;targets;formatlist

v2fmtfilt=run!v2fmtfilt -P 11005,port!11005,targets!msgclsfsrv,formatlist!fn_log;fn_log.v2s,conf!port;run;targets;formatlist

# new column
msgclsfsrv=run!msgclsfsrv -P 11010,port!11010,targets!tecfmtemit;cmdemit;calaproxy;reportemit;snmpemit;smtpemit,completers!report_cpl,remapper
```

```
s!smtpemit_remap,types!map_sev,rules!test_rule,auxkeys!errcode1;location,conf!port;run;targets;completers;remappers;types;rules;auxkeys
report_cpl=for!reportemit,fill!report_flag;1,if!sub_source
smtpemit_remap=for!smtpemit,fieldalias!msg;MSGBODY
map_sev=type!v2;fn_log,handledby!calaproxy;tecfmtemit,msgmaps!fn_severity.map;fn_severity,eventframe!7200,dupekey!$CLASS;L;ORIGIN;L
fn_severity=key!severity;L,fields!severity
test_rule=for!reportemit,type!tec;cala_test,corrkey!$CLASS;L;sub_source;L,rulesmaps!tr_map.rmp;tr_map
tr_map=key!$CLASS;L,conditions!sub_source;~count,fields!~count;count;$ACTION
errcode1=$CLASS;L;errcode;N
location=$CLASS;L;location;L

# new column
tecfmtemit=run!tecfmtemit -P 1120,port!1120,targets!tecifcsrv,conf!port;run;targets
cmdemit=run!cmdemit -P 11021,port!11021,conf!port;run
calaproxy=run!calaproxy -P 11022,port!11022,targets!remote_emit,conf!port;run;targets
reportemit=run!reportemit -P 11023,port!11023,dest_file!/home/cala/reports/default.rep,report_slots!date;msg,critical_slot!report_flag;1,report_file!tec;solaris_syslog;DEFAULT;sun.rep;tec;nt_security;nt_security.tpl;nt_security.rep,conf!port;run;dest_file;report_slots;critical_slot;report_file
snmpemit=run!snmpemit -P 11024,port!11024,conf!port;run
smtpemit=run!smtpemit -P 11025,port!11025,conf!port;run

# new column
tecifcsrv=run!tecifcsrvend -P 11030 -h @EventServer,port!11030,conf!port;run
remote_emit=ip!tecserver.stgt;tecserver2.stgt,port!12021;12022,conf!port;ip

```

Example H-1. a complete `logctlsrv.conf`

This is the CALAGUI main window of this configuration:



CALA Configuration v2.1a -- D:\calagui\data\sample\cala_doc.conf
Configuration Info
ascfileread
ascfileread
ntevtlogread
ntevtlogread
calamon
calamon
snmpread
snmpread
mssqlread
mssqlread
tecfmtfilt
tecfmtfilt
v2fmtfilt
v2fmtfilt
msgclsfsrv
msgclsfsrv
tecfmtemit
tecfmtemit
cmdemit
cmdemit
calaproxy
calaproxy
reportemit
reportemit
snmpemit
snmpemit
smtpemit
smtpemit
tecifcsrv
tecifcsrv
remote_emit
remote component

Appendix I. Detailed description of the status report

The status report shows detailed information for each configured component. To get a status report call logctlcmd with the argument status .

The status report is split into one part for each component and some general parts for configuration, environment and internal queues of logctlsrv.

configuration status

The configuration status shows the configuration items read from the configuration file, it also shows the ip address and port of the log control server.

```
configuration
  'controller_ip' = '10.0.114.201'
  'controller_port' = '11000'
  'logctlsrv_port' = '11000'
  'logctlcmd_port' = '11001'
  'cala_srv_port' = '1102'
  'maintenance' = 'Fri 2300;Sat 0300;01 2200;02 0500'
  'serverlist' = 'ascfileread,ntevtlogread,calamon,snmpread,tecfmtfilt,v2fmtfilt
,msgclsfsrv,tecfmtemit,cmdemit,calaproxy,reportemit,snmpemit,smtpemit,tecifcsrv,
remote_emit'
  'ascfileread' = 'run!ascfileread -P 11002 -AB 127.0.0.1,port!11002,targets!tec
fmtfilt;v2fmtfilt,pathlist!1;/var/adm;2;/fnsw/local/logs/elogs,ptrnlist!1;messag
es;2;elogYYYYMMDD,assoc!1;1;tec;solaris_syslog;2;2;v2;fn_log,conf!port;run;targe
ts;pathlist;ptrnlist;assoc'
  'ntevtlogread' = 'run!ntevtlogread -P 11003 -AB 127.0.0.1,port!11003,targets!t
ecfmtfilt,evtlog!1;security;2;application;3;system,spacereplacement!1;1;2;1;3;1,
assoc!1;tec;nt_security;2;tec;nt_application;3;tec;nt_system,skip_old!1;0;2;1;3;
1,prefilt_in!2;prefilt_nt_app.flt,prefilt_out!1;prefilt_nt_sec.flt,conf!port;run
;targets;evtlog;spacereplacement;assoc;skip_old;prefilt_in;prefilt_out'
  'calamon' = 'run!calamon -P 11006 -AB 127.0.0.1,port!11006,targets!msgclsfsrv,
source!CALA,subsource!monitoring,cmdline_slot!probe_arg,cmdtab!cmdtab.tbl,conf!p
ort;run;targets;source;subsource;cmdline_slot;cmdtab'
  'snmpread' = 'run!snmpread -P 11008 -AB 127.0.0.1 -p 162,port!11008,targets!ms
gclsfsrv,type!tec;snmp,class!SNMP_Event,conf!port;run;targets;type;class'
  'tecfmtfilt' = 'run!tecfmtfilt -P 11004 -AB 127.0.0.1,port!11004,targets!msgcl
sfsrv,formatlist!solaris_syslog;solaris_syslog.fmt;nt_system;nt_system.fmt;nt_ap
plication;nt_application.fmt;nt_security;nt_security.fmt,conf!port;run;targets;f
ormatlist'
  'v2fmtfilt' = 'run!v2fmtfilt -P 11005 -AB 127.0.0.1,port!11005,targets!msgclsf
srv,formatlist!fn_log;fn_log.v2s,conf!port;run;targets;formatlist'
...
```

Example I-1. An example status output

environment

The environment information shows the value of some environment variables used by CALA.

CALA_DIR
: the directory containing the CALA files

CALA_CACHE_DIR
: the directory where CALA stores its cachefiles

CENIT_ROOT
: the cenit tools base installation directory

CENIT_INSTID
: the installation id (used to distingiush several CALA installations on one machine)

CALA_ENV_FILE
: the name of the environment file sourced at startup

INTERP
: the Tivoli interpreter type (used by T/EC interface servers)

BINDIR
: the Tivoli binary directory (used by T/EC interface servers)

DBDIR
: the Tivoli database directory (used by T/EC interface servers)

TIV_ENV_FILE
: the Tivoli environment file (used by T/EC interface servers)

LCF_BINDIR
: the Tivoli endpoint binary directory (used by T/EC interface servers)

LCF_DATDIR
: the Tivoli endpoint data directory (used by T/EC interface servers)

The variables `CALA_DIR` and `CALA_CACHE_DIR` may have the value <unset> which means that they have not been set before starting CALA.

If `CALA_DIR` is unset, it is set to the local directory when starting CALA. `CALA_CACHE_DIR` is set to `CALA_DIR` if not specified.

All other variables are only needed if any T/EC interface server has been configured. They are only shown if they are set.

```
environment
  'CALA_DIR' = 'D:/cala'
  'CALA_CACHE_DIR' = ' D:/cala/.cache'
  'INTERP' = 'w32-ix86'
  'BINDIR' = 'C:/Tivoli/bin/w32-ix86'
  'DBDIR' = 'C:/Tivoli/db/pnsp1104.db'
```

Example I-2. An example status output of environment settings

log control server queues

There are three types of queues used by the log control server:

- the input queue holds data packages received from other processes
- the schedule queue holds internal data packages
- the outbound queues hold data packages to be send to other processes

The status output shows for each queue the number of data packages currently waiting to be processed.

```
input queue: 0 entries waiting

schedule queue: 0 entries waiting

pending outbound queues
  10.0.114.201:11001
    outbound packets: 0 entries waiting
  10.0.114.201:11003
    outbound packets: 0 entries waiting
  10.0.114.201:11006
    outbound packets: 0 entries waiting
  10.0.114.201:11002
    outbound packets: 0 entries waiting
  10.0.114.201:11004
    outbound packets: 0 entries waiting
  10.0.114.201:11005
    outbound packets: 0 entries waiting
  10.0.114.201:1120
    outbound packets: 0 entries waiting
  10.0.114.201:11010
    outbound packets: 0 entries waiting
  10.0.114.201:11022
    outbound packets: 0 entries waiting
```

Example I-3. An example status output of queue entries

component status general properties

The status of each component is shown in a own paragraph of the status output. There are some general properties shown for each component.

property name	meaning	value type, possible values	example

property name	meaning	value type, possible values	example
ip	IP address of the host running the process	IP address	127.0.0.1
adapter(s)_ bount_to	a list of network adapters, the process uses for reception	a list of network adapters	127.0.0.1
process	process id of the process	hexadecimal process id, or remote if process is remote	608A3100
version	the version of the binary (version without revision)	number	2.2
revision	the revision of the binary	version-string	2.01-002
startup_time	date and time when the process has been started	date in the format YYYY:MM:DD:hh:mm:ss	2002:05:22:12:55:23
up_time	processes up-time	time in the format YYYY:MM:DD:hh:mm:ss	0000:00:00:02:15:17
flags	flags the log control server holds for that process (see description below)	hexadecimal number	0000
outbound queue	data packages to be sent (to targets or to log control server)	no.entries waiting	0 entries waiting
running	the process exists and is running	boolean: 0,1 process is not running process is running	1
setup	the process is running and configuration is up to date	boolean: 0,1 configuration is needs updated configuration is up to date	1
is_local	process is local	boolean: 0,1 process is local process is remote	1
checked	process is configured and initialized	boolean: 0,1 configuration, initialization is pending configuration, initialization done	1

The flags field can have the following values (maybe conjunct)

value	flag name
0000	OK
0100	STARTUP_YEAR_INACCURATE
0200	UP_TIME_INACCURATE
0400	TIME_MOVED_BACKWARDS
0800	RESTART_RECOMMENDED
1000	TRANSMISSION_ERRORS

value	flag name
2000	SERVER_IS_DOWN

Each value except `0000` signs a critical failure and will result in a restart of the concerned process.

The component specific paragraph also shows the component specific configuration.

```
  'ntevtlogread' 'checked' = '1'
  'ntevtlogread' 'run' = 'ntevtlogread -P 11003 -AB 127.0.0.1'
  'ntevtlogread' 'port' = '11003'
  'ntevtlogread' 'targets' = 'tecfmtfilt'
  'ntevtlogread' 'evtlog' = '1;security;2;application;3;system'
  'ntevtlogread' 'spacereplacement' = '1;1;2;1;3;1'
  'ntevtlogread' 'assoc' = '1;tec;nt_security;2;tec;nt_application;3;tec;nt_syst
em'
  'ntevtlogread' 'skip_old' = '1;0;2;1;3;1'
  'ntevtlogread' 'prefilt_in' = '2;prefilt_nt_app.flt'
  'ntevtlogread' 'prefilt_out' = '1;prefilt_nt_sec.flt'
  'ntevtlogread' 'conf' = 'port;run;targets;evtlog;spacereplacement;assoc;skip_o
ld;prefilt_in;prefilt_out'
  'ntevtlogread' 'is_local' = '1'
  'ntevtlogread' 'ip' = '10.0.114.201'
  'ntevtlogread' 'process' = '50083100'
  'ntevtlogread' 'running' = '1'
  'ntevtlogread' 'setup' = '1'
  'ntevtlogread' 'adapter(s)_bound_to' = '127.0.0.1'
  'ntevtlogread' 'version' = '1.1'
  'ntevtlogread' 'startup_time' = '2002:05:22:12:55:18'
  'ntevtlogread' 'up_time' = '0000:00:00:02:15:17'
  'ntevtlogread' 'flags' = '0000'
  'ntevtlogread' 'revision' = '2.01-002'
  'ntevtlogread' outbound queue: 0 entries waiting
```

Example I-4. An example process status output

target status

All components having targets configured (all but the emitters), have additional properties to show the status of the connections between clients and servers.

property name	meaning	value type, possible values	example

property name	meaning	value type, possible values	example
stat_target_<target>	the status of the target configuration	unkown The target is not configure or not configuration for this target has been received yet. refresh The target has already been configured, but needs an update, because configuration has been changed. kown The target has been configured and the configuration is up to date	known
stat_target_<target>_ip_addr	the IP address of the target	ip address or unknown	127.0.0.1
stat_target_<target>_ip_port	the port of the target	port no. or unkown	16006
stat_target_<target>_status	the connection status	connected the connection is established unconnected the connection has been lost	connected

```
'ntevtlogread' 'stat_target_tecfmtfilt_ip_addr' = '127.0.0.1'
'ntevtlogread' 'stat_target_tecfmtfilt_ip_port' = '11004'
'ntevtlogread' 'stat_target_tecfmtfilt_status' = 'connected'
```

Example I-5. An example target status output

client status

property name	meaning	value type, possible values	example
client(<id>)_<ip-address>_ecnryption_level	shows the encryption level for each connected remote client	numeric, see *Security* for a list of encryption levels	1

<id> is a program internal id used to differentiate multiple connection from the same host.

```
001  'calaproxy' 'client(303c00)_10.0.3.201_encryption_level' = '1'
```

Example I-6. An example encryptionlevel output

ascfileread and ntevtlogread

property name	meaning	value type, possible values	example
stream_<id>_status	the reading status of this file	see list of possible states below	FESTAT_WAITING
stream_<id>_sequence	ascfileread: the number of blocks read from this file (a block may contain one event, only a part of an event or several events) ntevtlogread: the number of read events	number	12
stream_<id>_name	ascfileread: the filename of the logfile ntevtlogread: the name of the event log	filename/eventlogname	`/var/log/messages`

state	description
FESTAT_NEW	new file, not opened yet
FESTAT_OPEN	file opened, nothing read yet
FESTAT_READING	reading from file (there are still some unread characters)
FESTAT_WAITING	file contents have been read, waiting for new events
FESTAT_OUTDATED	file pattern is outdated
FESTAT_CLOSED	file has been closed
FESTAT_SUSPENDED	pipes only: unable to allocate memory - reading is suspended until new memory can be allocated

`<id>` is a CALA persistent internal identifier for data streams, it is used by the readers and filters to identify the stream after a restart of CALA.

```
'ascfileread' 'stream_778E6164_status' = 'FESTAT_WAITING'
'ascfileread' 'stream_778E6164_sequence' = '37'
'ascfileread' 'stream_778E6164_name' = '.\opc.log'
```

Example I-7. An example stream status output

tecfmtfilt and v2fmtfilt

property name	meaning	value type, possible values	example
stat_format_<sectype>	the status of the format syntax	known the format syntax file has been found and is loaded failed the syntax file could not be found or has syntactically errors	known

```
'tecfmtfilt' 'stat_format_websphere' = 'known'
```

Example I-8. An example formatfile status output

oracleread and mssqlread

property name	meaning	value type, possible values	example
open_db(<db>)_table(<table>)	successfully connected to database	successfully connected to database, ERROR unable to open data base	OK
last_entry__db(<db>)_ table(<table>)_field(<field>)	the last value of the entry id field	anything	

```
'mssqlread' 'open_db(tec)_table(tec.tec_t_evt_rep)' = 'OK'
'mssqlread' 'last_entry__db(tec)_table(tec.tec_t_evt_rep)_field(event_hndl)' = '1'
```

Example I-9. An example database connection status output (reader)

mysqlemit

property name	meaning	value type, possible values	example

property name	meaning	value type, possible values	example
database_<server>_<user>_<database>	successfully connected to database	successfully connected to database ERROR unable to open data base	

```
'mysqlemit' 'database_iccserv.stgt.cenit.de_calaweb_cala' = 'OK'
```

Example I-10. An example database connection status output (emitter)

reportemit

property name	meaning	value type, possible values	example
timestamp_last_ reported	timestamp of the last event written into any report file	timestamp	Wed May 22 15:03:56
timestamp_last_ unreported	timestamp of the last discarded event	timestamp	Wed May 22 14:58:32

```
'reportemit' 'timestamp_last_reported' = 'Wed May 22 15:03:56'
```

Example I-11. An example reportemit status output

How to detect configuration errors using the status output

After reconfiguring CALA, the following properties should be verified:

- general: stat_target_ <target> should never be unkown when CALA is running for at least ca. 1 minute (this shows, that the target configuration is incorrect)
- format filters: If stat_format_ <sectype> is failed , either the format file doesn t exist, or it s syntax is incorrect.
- database readers: ERROR in the property open_db(<db>)_table(<table>) shows, that the database configuration or user permissions is faulty.

Appendix J. Supported character sets

List of supported character sets

CALA internally works with UTF-8 encoded strings. The following character sets are supported for input and output (refer to component specific configuration):

European languages

: ASCII, ISO-8859-{1,2,3,4,5,7,9,10,13,14,15,16}, KOI8-R, KOI8-U, KOI8-RU, CP{437,737,775,852,853,855,857,858,860,861,863,865,869,1125, 1250,1251,1252,1253,1254,1257}, CP{850,866}, Mac{Roman,CentralEurope,Iceland,Croatian,Romania},Mac{Cyrillic,Ukraine,Greek,Turkish}, Macintosh

Semitic languages

: ISO-8859-{6,8}, CP{1255,1256}, CP862, Mac{Hebrew,Arabic}

Japanese

: EUC-JP, SHIFT_JIS, CP932, ISO-2022-JP, ISO-2022-JP-2, ISO-2022-JP-1

Chinese

: EUC-CN, HZ, GBK, GB18030, EUC-TW, BIG5, CP950, BIG5-HKSCS, ISO-2022-CN, ISO-2022-CN-EXT

Korean

: EUC-KR, CP949, ISO-2022-KR, JOHAB

Armenian

: ARMSCII-8

Georgian

: Georgian-Academy, Georgian-PS

Tajik

: KOI8-T

Thai

: TIS-620, CP874, MacThai

Laotian

: MuleLao-1, CP1133

Vietnamese

: VISCII, TCVN, CP1258

Platform specifics

: HP-ROMAN8, NEXTSTEP

Full Unicode

- UTF-8
- UCS-2, UCS-2BE, UCS-2LE
- UCS-4, UCS-4BE, UCS-4LE
- UTF-16, UTF-16BE, UTF-16LE
- UTF-32, UTF-32BE, UTF-32LE
- UTF-7
- C99, JAVA

Full Unicode with machine dependent endianness and alignment: UCS-2-INTERNAL, UCS-4-INTERNAL

Note: Microsoft Windows uses UCS2-LE when writing unicode data.

It has also some limited support for transliteration, i.e. when a character cannot be represented in the target character set, it can be approximated through one or several similarly looking characters. Transliteration is activated when //TRANSLIT is appended to the target encoding name.

The empty charset ("") specifies the systems' default charset.

Appendix K. Licenses

Overview

FileNet System Monitor 4.0.0 includes software from the following sources, and the use of this product is subject to the licenses associated with those embedded software products as described in the following pages.

Product/Library	**URL**	**License**
MySQL	http://www.mysql.com	MySQL Commercial License
Apache	http://httpd.apache.org/	The Apache Software License
PHP	http://www.php.net	The PHP License
win-bash	http://win-bash.sf.net	The GNU Public License
cygwin	http://www.cygwin.com	Cygwin API Licensing Terms, The GNU Public License
Perl for windows	ftp://theoryx5.uwinnipeg.ca/pub/other See also: http://www.perl.com	Dual licensed: The Artistic License or The GNU Public License
OpenJNLP	Mozilla Public License 1.1 (MPL 1.1)	
UnxUtils	http://unxutils.sourceforge.net/	The GNU Public License
Java JRE (Java Runtime Environment)	http://www.java.com	JavaTM 2, Standard Edition (J2SETM) Specification (*Specification*)
Java JRE (Java Runtime Environment)	http://java.com/en/download/license.jsp	Sun Microsystems, Inc. Binary Code License Agreement
Pari GP	http://pari.math.u-bordeaux.fr/	The GNU Public License (only version <= 1.39)
libxml2	http://www.xmlsoft.org/	The MIT License
net-snmp	http://www.net-snmp.org/	Net-SNMP License
OpenSSL	http://www.openssl.org/	OpenSSL License
java tar	http://www.trustice.com/java/tar	Public Domain
common.net	http://jakarta.apache.org/commons/net	The Apache Software License
cookswing	http://cookxml.sourceforge.net/cookswing	CookSwing License
Log4J	http://logging.apache.org/	The Apache Software License
Xerces2	http://xerces.apache.org/	The Apache Software License, SAX LICENSE, W3C® SOFTWARE NOTICE AND LICENSE
JBCL	http://www.borland.com/	BORLAND JBUILDER PROFESSIONAL VERSION 5

Product/Library	URL	License
JavaBeans(tm) Activation Framework	http://java.sun.com/products/javabeans	Sun Microsystems, Inc. Binary Code License Agreement
GNU iconv library	http://www.gnu.org/software/libiconv	*The GNU Lesser General Public License*
JFree Common	http://www.jfree.org/jcommon/index.php	*The GNU Lesser General Public License*
JFree Chart	http://www.jfree.org/jfreechart	*The GNU Lesser General Public License*

The Apache Software License

```
/* ====================================================================
* The Apache Software License, Version 1.1
* Copyright (c) 2000 The Apache Software Foundation. All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.
*
* 3. The end-user documentation included with the redistribution,
* if any, must include the following acknowledgment:
* "This product includes software developed by the
* Apache Software Foundation (http://www.apache.org/)."
* Alternately, this acknowledgment may appear in the software itself,
* if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" must
* not be used to endorse or promote products derived from this
* software without prior written permission. For written
* permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
* nor may "Apache" appear in their name, without prior written
* permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
====================================================================
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation. For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*
* Portions of this software are based upon public domain software
* originally written at the National Center for Supercomputing Applicat
ions,
* University of Illinois, Urbana-Champaign.
```

The PHP License

The PHP License, version 3.0

Redistribution and use in source and binary forms, with or without modification, is permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "PHP" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact group@php.net.

4. Products derived from this software may not be called "PHP", nor may "PHP" appear in their name, without prior written permission from group@php.net. You may indicate that your software works in conjunction with PHP by saying "Foo for PHP" instead of calling it "PHP Foo" or "phpfoo"

5. The PHP Group may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number.
   Once covered code has been published under a particular version of the license, you may always continue to use it under the terms of that version. You may also choose to use such covered code under the terms of any subsequent version of the license published by the PHP Group. No one other than the PHP Group has the right to modify the terms applicable to covered code created under this License.

6. Redistributions of any form whatsoever must retain the following acknowledgment:
   "This product includes PHP, freely available from <http://www.php.net/>".

THIS SOFTWARE IS PROVIDED BY THE PHP DEVELOPMENT TEAM “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PHP DEVELOPMENT TEAM OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```
This software consists of voluntary contributions made by many
individuals on behalf of the PHP Group.

The PHP Group can be contacted via Email at group@php.net.

For more information on the PHP Group and the PHP project,
please see <http://www.php.net>.

This product includes the Zend Engine, freely available at
<http://www.zend.com>.
```

MySQL Commercial License

See http://www.mysql.com/company/legal/licensing/commercial-license.html

The Commercial License is an agreement with MySQL AB for organizations that do not want to release their application source code. Commercially licensed customers get a commercially supported product with assurances from MySQL. Commercially licensed users are also free from the requirement of making their own application open source.

When your application is not licensed under either the GPL-compatible Free Software License as defined by the Free Software Foundation or approved by OSI, and you intend to or you may distribute MySQL software, you must first obtain a commercial license to the MySQL product.

Typical examples of MySQL distribution include:

- Selling software that includes MySQL to customers who install the software on their own machines.
- Selling software that requires customers to install MySQL themselves on their own machines.
- Building a hardware system that includes MySQL and selling that hardware system to customers for installation at their own locations.

Specifically:

- If you include the MySQL server with an application that is not licensed under the GPL or GPL-compatible license, you need a commercial license for the MySQL server.
- If you develop and distribute a commercial application and as part of utilizing your application, the end-user must download a copy of MySQL; for each derivative work, you (or, in some cases, your end-user) need a commercial license for the MySQL server and/or MySQL client libraries.
- If you include one or more of the MySQL drivers in your non-GPL application (so that your application can run with MySQL), you need a commercial license for the driver(s) in question. The MySQL drivers currently include an ODBC driver, a JDBC driver and the C language library.
- GPL users have no direct legal relationship with MySQL AB. The commercial license, on the other hand, is MySQL AB’s private license, and provides a direct legal relationship with MySQL AB.

With a commercial non-GPL MySQL **server** license, one license is required per database server (single installed MySQL binary). There are no restrictions on the number of connections, number of CPUs, memory or disks to that one MySQL database server. The MaxDB server is licensed per CPU or named user.

Non-Profits, Academic Institutions, and Private Individuals

If you represent a non-profit organization or an academic institution, we recommend you publish your application as an open source / free software project using the GPL license. In this manner, you are free to use MySQL software free of charge under the GPL license. We believe that if you have strong reasons to not publish your application in accordance with the GPL, you should purchase commercial licenses. Note that non-profits may apply to MySQL for free commercial licenses and such applications will be carefully considered.

If you are a private individual you are free to use MySQL software for your personal applications as long as you do not distribute them. If you distribute them, you must make a decision between the Commercial License and the GPL.

Please note that even if you ship a free demo version of your own application, the above rules apply.

Recommendations

Please note that MySQL AB can only give advice on which license is right for you. The final judgment, of course can be made only by a court of law. With that said, we recommend the commercial license to all commercial and government organizations. This frees you from the broad and strict requirements of the GPL license.

To all free software enthusiasts we recommend our products under the GPL license. We believe that MySQL AB is one of the world's largest companies that offers all its software under the GPL license.

To anyone in doubt, we recommend the commercial license. It is never wrong. Thanks to our cost-effective way of producing software, we are able to sell our commercial licenses at prices well under the industry average.

FOSS Exception

We have created a license exception which enables Free and Open Source software ("FOSS") to be able to include the GPL-licensed MySQL client libraries despite the fact that not all open source licenses are compatible with the GPL.Read more about the exception.

Older Versions

Note that some older versions of the MySQL database server (prior to 3.23.19) are using the Version 4, March 5, 1995, license. See the documentation for the specific version for more information.

When in Doubt

If you have any questions about Licensing, please contact the MySQL Sales Team to explore the options available for your specific scenario. http://www.mysql.com/buy-mysql

OSI = Open Source Initiative, www.opensource.org/licenses

GPL = GNU General Public License, http://www.gnu.org/copyleft/gpl.html

Version 4.1, 12 March 2004

Cygwin API Licensing Terms

This is a copy of CYGWIN_LICENSE from the cygwin sources

Source: http://cygwin.com/licensing.html

Mozilla Public License 1.1 (MPL 1.1)

```
1. Definitions.

1.0.1. "Commercial Use" means distribution or
otherwise making the Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or
contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of
the Original Code, prior Modifications used by a Contributor, and the
Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or
Modifications or the combination of the Original Code and Modifications,
in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a
mechanism generally accepted in the software development community for
the electronic transfer of data.

1.5. "Executable" means Covered Code in any form
other than Source Code.

1.6. "Initial Developer" means the individual or
entity identified as the Initial Developer in the Source Code notice
required by Exhibit A.

1.7. "Larger Work" means a work which combines
Covered Code or portions thereof with code not governed by the terms of
this License.

1.8. "License" means this document.
1.8.1. "Licensable" means having the right to grant, to the
maximum extent possible, whether at the time of the initial grant or
subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or
deletion from the substance or structure of either the Original Code or
any previous Modifications. When Covered Code is released as a series of
files, a Modification is:
A. Any addition to or deletion from the contents of a file
containing Original Code or previous Modifications.
B. Any new file that contains any part of the Original Code or
previous Modifications.

1.10. "Original Code" means Source Code of computer
software code which is described in the Source Code notice required by
Exhibit A as Original Code, and which, at the time of its release under
this License is not already Covered Code governed by this
License.
1.10.1. "Patent Claims" means any patent claim(s), now owned or
hereafter acquired, including without limitation, method, process,
and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the
Covered Code for making modifications to it, including all modules it
contains, plus any associated interface definition files, scripts used
```

to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor’s choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You"; (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may
be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims.If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.If Contributor’s Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor’s Modifications are Contributor’s original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients’ rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients’ rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient’s rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions.Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declatory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant’s Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant’s Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant’s Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY’S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys’ fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work

with Initial Developer and Contributors to distribute such
responsibility on an equitable basis. Nothing herein is intended or
shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as
"Multiple-Licensed". "Multiple-Licensed" means
that the Initial Developer permits you to utilize portions of the
Covered Code under Your choice of the NPL or the alternative licenses,
if any, specified by the Initial Developer in the file described in
Exhibit A.

EXHIBIT A -Mozilla Public License.

"The contents of this file are subject to the Mozilla Public
License Version 1.1 (the "License"); you may not use this file except in
compliance with the License. You may obtain a copy of the License
athttp://www.mozilla.org/MPL/

Software distributed under the License is distributed on an "AS
IS" basis, WITHOUT WARRANTY OFANY KIND, either express or implied. See
the License for the specific language governing rights andlimitations
under the License.

The Original Code is ______________________________________.

The Initial Developer of the Original Code is ________________________.
Portions created by ______________________ are
Copyright (C) _____________________________. All RightsReserved.

Contributor(s): ______________________________________.

Alternatively, the contents of this file may be used under the
terms of the _____ license (the "[___] License"), in which
case the provisions of [______] License are applicable instead of those
above. ; If you wish to allow use of your version of this file only
under the terms of the [____] License and not to allow others to use
your version of this file under the MPL, indicate your decision by
deleting the provisions above and replace them with the notice and other
provisions required by the [___] License. If you do not delete the
provisions above, a recipient may use your version of this file under
either the MPL or the [___] License."

[NOTE: The text of this Exhibit A may differ slightly from the
text of the notices in the Source Code files of the Original Code. You
should use the text of this Exhibit A rather than the text found in the
Original Code Source Code for Your Modifications.]

Source: http://www.opensource.org/licenses/artistic-license.php

The Artistic License

```
August 15, 1997


Preamble

The intent of this document is to state the conditions under
which a Package may be copied, such that the Copyright Holder
maintains some semblance of artistic control over the development of
the package, while giving the users of the package the right to use
and distribute the Package in a more-or-less customary fashion, plus
the right to make reasonable modifications.


Definitions

"Package" refers to the collection of files distributed by the
Copyright Holder, and derivatives of that collection of files created
through textual modification.

"Standard Version" refers to such a Package if it has not been
modified, or has been modified in accordance with the wishes of the
Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or
copyrights for the package.

"You" is you, if you’re thinking about copying or distributing
this Package.

"Reasonable copying fee" is whatever you can justify on the
basis of media cost, duplication charges, time of people involved,
and so on. (You will not be required to justify it to the Copyright
Holder, but only to the computing community at large as a market that
must bear the fee.)

"Freely Available" means that no fee is charged for the item
itself, though there may be fees involved in handling the item. It
also means that recipients of the item may redistribute it under the
same conditions they received it.


1. You may make and give away verbatim copies of the source
form of the Standard Version of this Package without
restriction, provided that you duplicate all of the original
copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other
modifications derived from the Public Domain or from the
Copyright Holder. A Package modified in such a way shall still
be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any
way, provided that you insert a prominent notice in each
changed file stating how and when you changed that file, and
provided that you do at least ONE of the following:

    a. place your modifications in the Public Domain or
```

otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b. use the modified Package only within your corporation or organization.

c. rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a. distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b. accompany the distribution with the machine-readable source of the Package with your modifications.

c. give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package’s interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package via the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4,

provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt is made to make this Package’s interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

Source: http://www.opensource.org/licenses/artistic-license.php

Sun Microsystems and Java Licenses

Java™ 2, Standard Edition (J2SE™) Specification (*Specification*)

NOTICE; LIMITED LICENSE GRANTS

Sun Microsystems, Inc. ("Sun") hereby grants you a fully-paid, non-exclusive, non-transferable, worldwide, limited license (without the right to sublicense), under the Sun's applicable intellectual property rights to view, download, use and reproduce the Specification only for the purpose of internal evaluation, which shall be understood to include developing applications intended to run on an implementation of the Specification provided that such applications do not themselves implement any portion(s) of the Specification.

Sun also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights or patent rights it may have in the Specification to create and/or distribute an Independent Implementation of the Specification that: (i) fully implements the Spec(s) including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; and (iii) passes the TCK (including satisfying the requirements of the applicable TCK Users Guide) for such Specification. The foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for any other purpose.

You need not include limitations (i)-(iii) from the previous paragraph or any other particular "pass through" requirements in any license You grant concerning the use of your Independent Implementation or products derived from it. However, except with respect to implementations of the Specification (and products derived from them) that satisfy limitations (i)-(iii) from the previous paragraph, You may neither: (a) grant or otherwise pass through to your licensees any licenses under Sun's applicable intellectual property rights; nor (b) authorize your licensees to make any claims concerning their implementation's compliance with the Spec in question.

For the purposes of this Agreement: "*Independent Implementation*" shall mean an implementation of the Specification that neither derives from any of Sun's source code or binary code materials nor, except with an appropriate and separate license from Sun, includes any of Sun's source code or binary code materials; and "*Licensor Name Space*" shall mean the public class or interface declarations whose names begin with "java", "javax", "com.sun" or their equivalents in any subsequent naming convention adopted by Sun through the Java Community Process, or any recognized successors or replacements thereof.

This Agreement will terminate immediately without notice from Sun if you fail to comply with any material provision of or act outside the scope of the licenses granted above.

TRADEMARKS

No right, title, or interest in or to any trademarks, service marks, or trade names of Sun or Sun's licensors is granted hereunder. Sun, Sun Microsystems, the Sun logo, Java, J2SE, and the Java Coffee Cup Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

DISCLAIMER OF WARRANTIES

THE SPECIFICATION IS PROVIDED "AS IS". SUN MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE OR THAT ANY PRACTICE OR IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER RIGHTS. This document does not represent any commitment to release or implement any portion of the Specification in any product.

THE SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. SUN MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.

LIMITATION OF LIABILITY

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF SUN AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will indemnify, hold harmless, and defend Sun and its licensors from any claims arising or resulting from: (i) your use of the Specification; (ii) the use or distribution of your Java application, applet and/or clean room implementation; and/or (iii) any claims that later versions or releases of any Specification furnished to you are incompatible with the Specification provided to you under this license.

RESTRICTED RIGHTS LEGEND

U.S. Government: If this Specification is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Specification and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

REPORT

You may wish to report any ambiguities, inconsistencies or inaccuracies you may find in connection with your use of the Specification ("Feedback"). To the extent that you provide Sun with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant Sun a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose related to the Specification and future versions, implementations, and test suites thereof.

(LFI#132520/Form ID#011801)

Source: http://java.sun.com/j2se/1.4.2/j2sdk-1_4_2-doc-license.html

Sun Microsystems, Inc. Binary Code License Agreement

```
                        Sun Microsystems, Inc.
                    Binary Code License Agreement

READ THE TERMS OF THIS AGREEMENT AND ANY PROVIDED SUPPLEMENTAL LICENSE TERMS
(COLLECTIVELY "AGREEMENT") CAREFULLY BEFORE OPENING THE SOFTWARE MEDIA
PACKAGE. BY OPENING THE SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS OF
THIS AGREEMENT. IF YOU ARE ACCESSING THE SOFTWARE ELECTRONICALLY, INDICATE
YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END
OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN
THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF THE
SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" BUTTON AT THE END
OF THIS AGREEMENT.

1. LICENSE TO USE. Sun grants you a non-exclusive and non-transferable
license for the internal use only of the accompanying software and
documentation and any error corrections provided by Sun (collectively
"Software"), by the number of users and the class of computer hardware for
which the corresponding fee has been paid.

2. RESTRICTIONS. Software is confidential and copyrighted. Title to
Software and all associated intellectual property rights is retained by Sun
and/or its licensors. Except as specifically authorized in any Supplemental
License Terms, you may not make copies of Software, other than a single copy
of Software for archival purposes. Unless enforcement is prohibited by
applicable law, you may not modify, decompile, or reverse engineer
Software. You acknowledge that Software is not designed, licensed or
intended for use in the design, construction, operation or maintenance of
any nuclear facility. Sun disclaims any express or implied warranty of
fitness for such uses. No right, title or interest in or to any trademark,
service mark, logo or trade name of Sun or its licensors is granted under
this Agreement.

3. LIMITED WARRANTY. Sun warrants to you that for a period of ninety (90)
days from the date of purchase, as evidenced by a copy of the receipt, the
media on which Software is furnished (if any) will be free of defects in
materials and workmanship under normal use. Except for the foregoing,
Software is provided "AS IS". Your exclusive remedy and Sun’s entire
liability under this limited warranty will be at Sun’s option to replace
Software media or refund the fee paid for Software.

4. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS
OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED
WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR
NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS
ARE HELD TO BE LEGALLY INVALID.

5. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO
EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR
DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE
DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT
OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS
BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun’s
liability to you, whether in contract, tort (including negligence), or
otherwise, exceed the amount paid by you for Software under this Agreement.
The foregoing limitations will apply even if the above stated warranty fails
of its essential purpose.
```

6. Termination. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Upon Termination, you must destroy all copies of Software.

7. Export Regulations. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

8. U.S. Government Restricted Rights. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government’s rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

9. Governing Law. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

10. Severability. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

11. Integration. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

JAVA OPTIONAL PACKAGE

JAVABEANS(TM) ACTIVATION FRAMEWORK, VERSION 1.0.2
SUPPLEMENTAL LICENSE TERMS

These supplemental license terms ("Supplemental Terms") add to or modify the terms of the Binary Code License Agreement (collectively, the "Agreement"). Capitalized terms not defined in these Supplemental Terms shall have the same meanings ascribed to them in the Agreement. These Supplemental Terms shall supersede any inconsistent or conflicting terms in the Agreement, or in any license contained within the Software.

1. Software Internal Use and Development License Grant. Subject to the terms and conditions of this Agreement, including, but not limited to Section 3 (Java(TM) Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce internally and use internally the binary form of the Software, complete and unmodified, for the sole purpose of designing, developing and testing your Java applets and applications ("Programs").

2. License to Distribute Software. In addition to the license granted in Section 1 (Software Internal Use and Development License Grant) of these Supplemental Terms, subject to the terms and conditions of this Agreement, including but not limited to, Section 3 (Java Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce and distribute the Software in binary code form only, provided that you (i) distribute the Software complete and unmodified and only bundled as part of your Programs, (ii) do not distribute additional software intended to replace any component(s) of the Software, (iii) do not remove or alter any proprietary legends or notices contained in the Software, (iv) only distribute the Software subject to a license agreement that protects Sun’s interests consistent with the terms contained in this Agreement, and (v) agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys’ fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

3. Java Technology Restrictions. You may not modify the Java Platform Interface ("JPI", identified as classes contained within the "java" package or any subpackages of the "java" package), by creating additional classes within the JPI or otherwise causing the addition to or modification of the classes in the JPI. In the event that you create an additional class and associated API(s) which (i) extends the functionality of the Java platform, and (ii) is exposed to third party software developers for the purpose of developing additional software which invokes such additional API, you must promptly publish broadly an accurate specification for such API for free use by all developers. You may not create, or authorize your licensees to create additional classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

4. No Support. Sun is under no obligation to support the Software or to provide you with updates or error corrections. You acknowledge that the Software may have defects or deficiencies which cannot or will not be corrected by Sun.

5. Trademarks and Logos. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at http://www.sun.com/policies/trademarks. Any use you make of the Sun Marks inures to Sun’s benefit.

6. Source Code. Software may contain source code that is provided solely for reference purposes pursuant to the terms of this Agreement. Source code may not be redistributed unless expressly provided for in this Agreement.

7. Termination for Infringement. Either party may terminate this Agreement immediately should any Software become, or in either party’s opinion be likely to become, the subject of a claim of infringement of any intellectual property right.

For inquiries please contact: Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303

(LFI#115020/Form ID#011801)

BORLAND JBUILDER PROFESSIONAL VERSION 5

BORLAND NO-NONSENSE LICENSE STATEMENT AND LIMITED WARRANTY

IMPORTANT - READ CAREFULLY

This license statement and limited warranty constitutes a legal agreement ("License Agreement") between you (either as an individual or a single entity) and Borland Software Corporation ("Borland") for the software product ("Software") identified above, including any software, media, and accompanying on-line or printed documentation.

BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT.

If you are the original purchaser of the Software and you do not agree with the terms and conditions of the License Agreement, promptly return the unused Software to the place from which you obtained it for a full refund.

If you are accepting this License Agreement on behalf of a corporation, partnership or other legal entity, the use of the terms "you" and "your" in this License Agreement will refer to such entity.

TERMS AND CONDITIONS

1. GRANT OF LICENSE.

a. Subject to the terms and conditions of this License Agreement, Borland grants to you a personal, nonexclusive, nontransferable and limited license to install and use the Software for the purposes set forth herein. Unless you have purchased additional licenses from Borland, you may only install and use a single copy of the Software on a computer and freely move the Software from one computer to another, provided that you are the only individual using the Software. If you are an entity, Borland grants you the right to designate one individual within your organization ("Named User") to have the right to use the Software in the manner provided herein. If you have purchased additional licenses from Borland or a Borland authorized reseller, you may install and use the number of copies of the Software up to the number of users, CPU s, servers and/or at the sites granted to you in writing by Borland ("Licensed Copies").

b. This Software is owned by Borland or its suppliers and is protected by copyright law, international copyright treaties, as well as other proprietary notices. Therefore, you must treat this Software like any other copyrighted material (e.g., a book) and you agree that the total number of copies of the Software used by you may not exceed the number of Licensed Copies paid for by you, except that you may either make one copy of the Software solely for backup or archival purposes or transfer the Software to a single hard disk provided you keep the original solely for backup or archival purposes.

c. Subject to the further terms and conditions of this License Agreement, the term of this license is perpetual (unless terminated as provided below). You may transfer the Software and documentation on a permanent basis provided you retain no copies and the recipient agrees to the terms of this License Agreement.

2. LICENSE RESTRICTIONS.

a. Except as provided in this License Agreement, you receive no rights and agree not to transfer, rent, lease, lend, copy, modify, translate, port, localize, create derivative works of, market, distribute, sublicense, time-share or electronically transmit or receive the Software, media or documentation. You acknowledge that the Software in source code form remains a confidential trade secret of Borland and/or its suppliers and therefore you agree not to modify the Software or attempt to reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation. As a confidential trade secret, you shall use your best efforts to protect the proprietary or confidential information supplied by Borland in its Software (including any source code), in the same manner in which you would protect your own proprietary or confidential information, but not less than reasonable precautions to protect such proprietary or confidential information and you shall not use such proprietary or confidential trade secret for your own benefit or the benefit of any other person or entity, except as may be specifically permitted hereunder.

b. If you have purchased an upgrade version of the Software, it constitutes a single product with the Borland software that you upgraded. You may use or transfer the upgrade version of the Software together with the original only in accordance with this License Agreement.

3. REGISTRATION.

You are required to register the Software with Borland. You will be prompted to register the Software at the time of your first use of the Software, at which time you will be notified (or directed to resources explaining) how information provided by you during registration may be used and you will be afforded the opportunity to opt out of certain uses of such information.

4. ADDITIONAL LICENSE TERMS

ADDITIONAL LICENSE TERMS FOR BORLAND JBUILDER PROFESSIONAL VERSION 5

Borland grants to you as an individual, a personal, nonexclusive license to install and use the Software for the sole purposes of designing, developing, testing, and deploying, in executable form only, application programs which you create.

ADDITIONAL LICENSE TERMS FOR EDUCATIONAL USERS

If this License Agreement was entered into for educational purposes, you may not use the Software for any commercial, business, governmental or institutional purpose of any kind, other than teaching the Courses (as defined below), in the case of instructors ("Noncommercial Purposes"). Borland grants to you as an individual, a personal, nonexclusive limited license to install and use the Software for the sole purpose of providing or receiving instruction within the limited scope of guided computer programming and/or software training courses ("Courses"). You may only reproduce, distribute and use programs that you create for personal, Noncommercial Purposes, within the scope of the course work required by the Courses.

GENERAL TERMS THAT APPLY TO COMPILED PROGRAMS AND REDISTRIBUTABLES

You may write and compile (including byte-code compile) your own application programs using the Software, including any libraries and source code included for such purpose with the Software. You may reproduce and distribute, in executable form only, programs which you create using the Software without additional license or fees, subject to all of the conditions in this License Agreement.

Borland products may include certain files ("Redistributables") intended for distribution by you to the users of programs you create. Redistributables include, for example, those files identified in the accompanying printed or on-line documentation as redistributable files, those files preselected for deployment by an install utility provided with the Software (if any), or those files pre-selected by a third party install utility which operates under control of an install script which Borland has certified (if any) for use by licensed users of this Software for deploying applications. In any event, the Redistributables for the Software are only those files specifically designated as such by Borland. From time to time, Borland may designate other files as Redistributables. You should refer to the documentation, including any "readme" or "deploy" files included with the Software, for additional information.

Subject to all of the conditions in this License Agreement, you may reproduce and distribute exact copies of the Redistributables, provided that such copies are made from the original copy of the Software or the copy transferred to the single hard disk. Copies of Redistributables may only be distributed with and for the sole purpose of executing application programs permitted under this License Agreement that you have created using the Software. Under no circumstances may any copies of Redistributables be distributed separately. Regardless of any modifications which you make and regardless of how you might compile, link, and/or package your programs, under no circumstances may the libraries (including runtime libraries), code, Redistributables, and/or other files of the Software (including any portions thereof) be used for developing programs by anyone other than you. Only you as the licensed user (or the Named User for your entity) have the right to use the libraries (including runtime libraries), code, Redistributables, or other files of the Software (or any portions thereof) for developing programs created with the Software. In particular, you may not share copies of the Redistributables with other co-developers. You may not reproduce or distribute any Borland documentation without Borland's permission.

The license granted in this License Agreement for you to create your own compiled programs and distribute your programs and the Redistributables (if any) is subject to all of the following conditions: (i) all copies of the programs you create must bear a valid copyright notice, either your own or the Borland copyright notice that appears on the Software; (ii) you may not remove or alter any Borland copyright, trademark or other proprietary rights notice contained in any portion of Borland libraries, source code, Redistributables or other files that bear such a notice; (iii) all rights and obligations of the parties here are personal to them and this License Agreement is not intended to benefit nor shall it be deemed to give rise to, any rights in any third party; consequently, Borland provides no warranty at all to any person, other than the Limited Warranty provided to the original purchaser of the Software, and you will remain solely responsible to anyone receiving your programs for support, service, upgrades, or technical or other assistance, and such recipients will have no right to contact Borland for such services or assistance; (iv) you will indemnify, defend and hold Borland, its related companies and its suppliers, harmless from and against any claims or liabilities arising out of the use, reproduction or distribution of your programs; (v) your programs must be written using a licensed, registered copy of the Software; (vi) your programs must add primary and substantial functionality, and may not be merely a set or subset of any of the libraries (including runtime libraries), code, Redistributables or other files of the Software; (vii) regardless of any modifications which you make and regardless of how you might compile, link, or package your programs, the libraries (including runtime libraries), code, Redistributables, and/or other files of the Software (including any portions thereof) may not be used in programs created by your end users (i.e., users of your programs) and may not be further redistributed by your end users; and (viii) you may not use Borland's or any of its suppliers' names, logos, or trademarks to market your programs, except to state that your program was written using the Software.

The Software might include source code, redistributable files, and/or other files provided by a third party vendor ("Third Party Software"). Since use of Third Party Software might be subject to license restrictions imposed by the third party vendor, you should refer to the on-line documentation (if any) provided with Third Party Software for any license restrictions imposed by the third party vendor. In any event, any license restrictions imposed by a third party vendor are in addition to, not in lieu of, the terms and conditions of this License Agreement.

All Borland Software provided under this License Agreement, including but not limited to libraries, source code, Redistributables and other files remain Borland's exclusive property. Borland will retain all right title and interest in and to the libraries, source code, Redistributables and other files, including the Intellectual Property contained in such property (including but not limited to, ownership of all copyrights, patents, trademarks, service marks worldwide). Regardless of any modifications that you make, you may not distribute any files (particularly Borland source code and other non-executable files) except those that Borland has expressly designated as Redistributables. Nothing in this License Agreement permits you to derive the source code of files that Borland has provided to you in executable form only, or to reproduce, modify, use, or distribute the source code of such files. You are not, of course, restricted from distributing source code or byte code that is entirely your own. Source code which you generate with a Borland source code generator, such as the Application Wizard, is considered by Borland to be your code.

Contact Borland for the applicable royalties due and other licensing terms for all other uses and/or distribution of the Redistributables.

ADDITIONAL LICENSE TERMS FOR JAVA BEANS COMPONENT LIBRARY

You may not distribute any program or file which includes, is created from, or otherwise incorporates portions of the Software identified as JAVA BEANS COMPONENT LIBRARY (if any), if such program or file is a general purpose development tool, library, component, and/or environment for Java, or is otherwise generally competitive with or a substitute for Borland's JBUILDER or the JAVA BEANS COMPONENT LIBRARY Software.

ADDITIONAL LICENSE TERMS FOR DATAEXPRESS

You may not distribute any program or file which includes, is created from, or otherwise incorporates portions of the Software identified as DATAEXPRESS (if any) if such program or file is a general purpose development tool, library, component, and/or environment for Java, or is otherwise generally competitive with or a substitute for Borland's JBUILDER or the DATAEXPRESS Software.

ADDITIONAL LICENSE TERMS FOR DBSWING

You may not distribute any program or file which includes, is created from, or otherwise incorporates portions of the Software identified as DBSWING (if any) if such program or file is a general purpose development tool, library, component, and/or environment for Java, or is otherwise generally competitive with or a substitute for Borland’s JBUILDER or the DBSWING Software.

ADDITIONAL LICENSE TERMS FOR JBUILDER OPEN TOOLS API

You may not distribute any program or file which includes, is created from, or otherwise incorporates portions of the Software identified as JBUILDER OPEN TOOLS API if such program or file is a general purpose development tool, library, component, and/or environment for Java, or is otherwise generally competitive with or a substitute for Borland’s JBUILDER or the JBUILDER OPEN TOOLS API Software.

ADDITIONAL LICENSE TERMS FOR API DECOMPILER

Notwithstanding the limitation against decompiling the Software, Borland grants to you as the licensed user of the Software the limited right to use that portion of the Software identified as "API Decompiler" for inspecting the public application programming interface (API) of the JAVA BEANS COMPONENT LIBRARY, DATAEXPRESS, DBSWING and OPENTOOLS API Software.

ADDITIONAL LICENSE TERMS FOR JDATASTORE

The portion of the Software identified as JDATASTORE (if any) is licensed for development purposes only. To deploy a particular application program created with the Software that requires JDATASTORE, you must first obtain a deployment license (available separately) from Borland for each deployment of the particular application program.

ADDITIONAL LICENSE TERMS FOR INTERNETBEANS

You may not distribute any program or file which includes, is created from, or otherwise incorporates portions of the Software identified as INTERNETBEANS (if any) if such program or file is a general purpose development tool, library, component, and/or environment for Java, or is otherwise generally competitive with or a substitute for Borland’s JBUILDER or the INTERNETBEANS Software.

ADDITIONAL LICENSE TERMS FOR XMLBEANS

You may not distribute any program or file which includes, is created from, or otherwise incorporates portions of the Software identified as XMLBEANS (if any) if such program or file is a general purpose development tool, library, component, and/or environment for Java, or is otherwise generally competitive with or a substitute for Borland’s JBUILDER or the XMLBEANS Software.

ADDITIONAL LICENSE TERMS FOR SITRAKA SOFTWARE

A portion of the Software includes certain Third Party Software provided by Sitraka ("Sitraka Software"). The Sitraka Software is licensed for proof of development purposes only. To develop a particular application program or to deploy that application program created with the Software that requires the Sitraka Software, you must first obtain a license (available separately) from Sitraka.

ADDITIONAL LICENSE TERMS FOR /N SOFTWARE SOFTWARE

A portion of the Software includes certain Third Party Software provided by /n software ("/n software Software"). The /n software Software is licensed for development purposes only. To deploy a particular application program created with the Software that requires the /n software Software, you must first obtain a deployment license (available separately) from /n software for each deployment of the particular application program.

5. LIMITED WARRANTY

a. Software provided under this License Agreement, including but not limited to libraries, source code, Redistributables and other files are provided "as is" without warranty of any kind except as expressly provided in this paragraph. Borland warrants that, except with respect to the Redistributables, the Software, as updated and when properly used, will perform substantially in accordance with the accompanying documentation, and the Software media will be free from defects in materials and workmanship, for a period of ninety (90) days from the date of receipt. Any implied warranties on the Software are limited to ninety (90) days. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

b. Borland’s and its suppliers’ entire liability and your exclusive remedy shall be, at Borland’s option, either (a) return of the price paid, or (b) repair or replacement of the Software that does not meet Borland’s Limited Warranty and which is returned to Borland with a copy of your receipt. DO NOT RETURN ANY PRODUCT UNTIL YOU HAVE CALLED THE BORLAND CUSTOMER SERVICE DEPARTMENT AND OBTAINED A RETURN AUTHORIZATION NUMBER. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, or misapplication. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. Outside the United States, neither these remedies nor any product support services offered by Borland are available without proof of purchase from an authorized non-U.S. source.

c. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, BORLAND AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR

FAILURE TO PROVIDE SUPPORT SERVICES. BORLAND DOES NOT WARRANT THAT THE BORLAND SOFTWARE WILL BE ERROR FREE OR WILL OPERATE WITHOUT INTERRUPTION. THE ENTIRE RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE BORLAND SOFTWARE AND ACCOMPANYING WRITTEN MATERIALS REMAINS WITH YOU. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

6. LIMITATION OF LIABILITY

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL BORLAND OR ITS LICENSORS OR SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS OR BUSINESS INTERRUPTION, GOODWILL, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) WHETHER BASED ON PRINCIPLES OF CONTRACT, TORT (INCLUDING NEGLIGENCE), DUTY, INDEMNITY, CONTRIBUTION OR OTHERWISE, ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF BORLAND HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, BORLAND’S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS LICENSE AGREEMENT SHALL BE LIMITED TO THE GREATER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT OR U.S. $25; PROVIDED, HOWEVER, IF YOU HAVE ENTERED INTO A BORLAND SUPPORT SERVICES AGREEMENT, BORLAND’S ENTIRE LIABILITY REGARDING SUPPORT SERVICES SHALL BE GOVERNED BY THE TERMS OF THAT AGREEMENT. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

7. AUDIT RIGHTS

You agree to keep all usual and proper records and books of account and all usual and proper entries relating to each installation, copy and authorized user of the Software. Borland may cause an audit and/or inspection to be made of your applicable records and facilities in order to verify your compliance with the terms of this License Agreement. Within thirty (30) days of notice by Borland to you of any error or omission disclosed by such audit, you will make prompt adjustment and reimbursement to Borland of such error or omission. Any such audit or inspection will be conducted by an audit and/or inspection team selected by Borland (other than on a contingent fee basis). Any audit and/or inspection will be conducted during regular business hours at your facilities, with five (5) days written notice. You agree to provide Borland s designated audit or inspection team access to the relevant records and facilities and to otherwise cooperate with such audit or inspection team. Any such audit and/or inspection will be paid for by Borland, provided, however, that in the event that any such examination discloses a shortfall in payment of more than five percent (5%) for any quarter, you agree to (i) pay or reimburse Borland for the reasonable expenses of the examination, as determined in good faith by the parties at the completion of the examination, and (ii) immediately remit payment to Borland for the full amount of any disclosed shortfalls (in addition to the reasonable expenses for such examination).

8. EXPORT CONTROLS

You may not download or otherwise export or re-export the Software or any underlying information or technology except in full compliance with all United States and other applicable laws and regulation. In particular, without limiting the foregoing, the Software cannot be downloaded, exported or re-exported into (or to a national or resident of) Cuba, Iraq, Libya, Sudan, North Korea, Iran, Syria, or any other country to which the United States has embargoed goods; or to anyone on the United States Treasury Department’s list of Specially Designated Nationals, the United States Commerce Department’s Entity List, or the United States Commerce Department’s Denied Parties list. You agree to the foregoing and you are representing and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use this Software.

9. HIGH RISK ACTIVITIES

The Software is not fault-tolerant and is not designed intended, or licensed for use in line control equipment or in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, and life support or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). Without limiting the generality of the foregoing, Borland and its suppliers specifically disclaim any express or implied warranty of fitness for High Risk Activities.

10. U.S. GOVERNMENT RESTRICTED RIGHTS

The Software and any accompanying documentation are "Commercial Items", as that term is defined at 48 CFR Section 2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation", as such terms are used in 48 CFR Sections 12.212 and 227.7202, as applicable. Consistent with 48 CFR Sections 12.212 or 227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished rights reserved under the copyright laws of the United States. Manufacturer is Borland Software Corporation, 100 Enterprise Way, Scotts Valley, CA 95066-3249.

11. TERMINATION

Your license to use the Software shall become effective on the date you agree to the terms and conditions of this License Agreement. Your license to use the Software shall terminate automatically if you fail to comply with the limitations described in this License Agreement. No notice shall be required from Borland to effectuate such termination. Upon expiration or termination of this License Agreement for any reason, you shall make no further use of the Software and shall destroy all copies of the Software and all of its component parts on all systems, in all forms, in all types of media and computer memory, and whether or not modified or combined with other materials.

12. GENERAL PROVISIONS

This License Agreement is governed by the laws of the State of California, U.S.A., excluding its or any other jurisdiction s choice of law rules and excluding the United Nations Convention for Contracts for the International Sale of Goods, and you further consent to the exclusive jurisdiction by the state and federal courts sitting in Santa Clara County in the State of California for any dispute regarding this License Agreement. This License Agreement gives you specific legal rights; you may have others which vary from state to state and from country to country. Borland reserves all rights not specifically granted in this License Agreement.

This License Agreement will not be modified except by a properly executed written agreement. Any terms and conditions of any purchase order or other instrument issued by you in connection with this License Agreement which are in addition to, inconsistent with or different from the terms and conditions of this License Agreement will be of no force or effect.

If any provision of this License Agreement is found void or unenforceable, the remainder will remain valid and enforceable according to its terms. If any remedy provided is determined to have failed for its essential purpose, all limitations of liability and exclusions of damages set forth in the Limited Warranty shall remain in effect.

Failure by either party at any time to enforce any obligation by the other party, to claim a breach of any term of this License Agreement or to exercise any power agreed to hereunder will not be construed as a waiver of any right, power or obligation under this License Agreement, will not affect any subsequent breach, and will not prejudice either party as regards any subsequent action.

Except as expressly permitted hereby, you may not assign any rights or obligations under this License Agreement without the prior consent of Borland.

The provisions of this License Agreement that by their nature and content are intended to survive the performance hereof shall so survive the completion and termination of this License Agreement.

```
IF YOU AGREE TO THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT,
please press the "I ACCEPT THE LICENSE AGREEMENT" button below. This will
be the legal equivalent of your signature on a written contract, and
equally binding. You must agree to these terms and conditions in order to
download and install the Software. If you do not agree with these terms and
conditions, you should press the "EXIT" button below to exit this download
process, as Borland is unwilling to license the Software to you in such
case.
```

SAX LICENSE

This license applies to all interfaces and classes in the org/xml/sax hierarchy.

This module, both source code and documentation, is in the Public Domain, and comes with *NO WARRANTY*. See http://www.saxproject.org for further information.

Copyright Status

SAX is free.

In fact, it’s not possible to own a license to SAX, since it’s been placed in the public domain.

No Warranty

Because SAX is released to the public domain, there is no warranty for the design or for the software implementation, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide SAX "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of SAX is with you. Should SAX prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute SAX, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use SAX (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the SAX to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

Copyright Disclaimers

This page includes statements to that effect by David Megginson, who would have been able to claim copyright for the original work.

SAX 1.0

Version 1.0 of the Simple API for XML (SAX), created collectively by the membership of the XML-DEV mailing list, is hereby released into the public domain.

No one owns SAX: you may use it freely in both commercial and non-commercial applications, bundle it with your software distribution, include it on a CD-ROM, list the source code in a book, mirror the documentation at your own web site, or use it in any other way you see fit.

David Megginson, sax@megginson.com

1998-05-11

SAX 2.0

I hereby abandon any property rights to SAX 2.0 (the Simple API for XML), and release all of the SAX 2.0 source code, compiled code, and documentation contained in this distribution into the Public Domain. SAX comes with NO WARRANTY or guarantee of fitness for any purpose.

David Megginson, david@megginson.com

2000-05-05

W3C SOFTWARE NOTICE AND LICENSE

http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code.
3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

This formulation of W3C’s notice and license became active on December 31 2002. This version removes the copyright ownership notice such that this license can be used with materials other than those owned by the W3C, reflects that ERCIM is now a host of the W3C, includes references to this specific dated version of the license, and removes the ambiguous grant of "use". Otherwise, this version is the same as the previous version and is written so as to preserve the Free Software Foundation’s assessment of GPL compatibility and OSI’s certification under the Open Source Definition. Please see our Copyright FAQ for common questions about using materials from our site, including specific terms and conditions for packages like libwww, Amaya, and Jigsaw. Other questions about this notice can be directed to site-policy@w3.org.

Joseph Reagle <mailto:site-policy@w3.org

Last revised by Reagle $Date: 2006/09/11 11:37:35 $

The GNU Public License

```
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                      59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

                        Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation’s software and to any other program whose authors commit to
using it. (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.) You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have. You must make sure that they, too, receive or can get the
source code. And you must show them these terms so they know their
rights.

We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software. If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors’ reputations.

Finally, any free program is threatened constantly by software
patents. We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary. To prevent this, we have made it clear that any
patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and
modification follow.
```

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is

void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among

countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program’s name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The GNU Lesser General Public License

```
  GNU LESSER GENERAL PUBLIC LICENSE
       Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
     51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]

    Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

  To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library.  Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
```

introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user’s freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or

```
distribute the Library is void, and will automatically terminate your
rights under this License.  However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

  9. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Library or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

  10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties with
this License.

  11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all.  For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus
excluded.  In such case, this License incorporates the limitation as if
```

written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively

```
convey the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!
```

The MIT License

```
Except where otherwise noted in the source code (e.g. the files hash.c,
list.c and the trio files, which are covered by a similar licence but
with different Copyright notices) all the files are:

 Copyright (C) 1998-2003 Daniel Veillard.  All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is fur-
nished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FIT-
NESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL THE
DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CON-
NECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Daniel Veillard shall not
be used in advertising or otherwise to promote the sale, use or other deal-
ings in this Software without prior written authorization from him.
```

RSA Security Releases RSA Encryption Algorithm into Public Domain

http://www.rsasecurity.com/press_release.asp?doc_id=261&id=1034

"c = me mod n" Made Available Two Weeks Early

BEDFORD, MA., Wednesday, September 06, 2000 — RSA® Security Inc. (NASDAQ: RSAS) today announced it has released the RSA public key encryption algorithm into the public domain, allowing anyone to create products that incorporate their own implementation of the algorithm. This means that RSA Security has waived its rights to enforce the patent for any development activities that include the RSA algorithm occurring after September 6, 2000.

Represented by the equation "$c = m^e \mod n$," the RSA algorithm is widely considered the standard for encryption and the core technology that secures the vast majority of the e-business conducted on the Internet. The U.S. patent for the RSA algorithm (# 4,405,829, "Cryptographic Communications System And Method") was issued to the Massachusetts Institute of Technology (MIT) on September 20, 1983, licensed exclusively to RSA Security and expires on September 20, 2000.

"So much misinformation has been spread recently regarding the expiration of the RSA algorithm patent that we wanted to create an opportunity to state the facts," said Art Coviello, chief executive officer of RSA Security. "RSA Security's commercialization of the RSA patent helped create an entire industry of highly secure, interoperable products that are the foundation of the worldwide online economy. Releasing the RSA algorithm into the public domain now is a symbolic next step in the evolution of this market, as we believe it will cement the position of RSA encryption as the standard in all categories of wired and wireless applications and devices. RSA Security intends to continue to offer the world's premier implementation of the RSA algorithm and all other relevant encryption technologies in our RSA BSAFE® software solutions and we remain confident in our leadership in the encryption market."

The MD5 code can be downloaded here: http://www.faqs.org/rfcs/rfc1321.html

Net-SNMP License

```
License

Various copyrights apply to this package, listed in various separate parts
below. Please make sure that you read all the parts. Up until 2001,
the project was based at UC Davis, and the first part covers all code
written during this time. From 2001 onwards, the project has been
based at SourceForge, and Networks Associates Technology, Inc hold the
copyright on behalf of the wider Net-SNMP community, covering all
derivative work done since then. An additional copyright section has
been added as Part 3 below also under a BSD license for the work
contributed by Cambridge Broadband Ltd. to the project since 2001.
An additional copyright section has been added as Part 4 below also
under a BSD license for the work contributed by Sun Microsystems, Inc.
to the project since 2003.

Code has been contributed to this project by many people over
the years it has been in development, and a full list of contributors
can be found in the README file under the THANKS section.


---- Part 1: CMU/UCD copyright notice: (BSD like) -----


Copyright 1989, 1991, 1992 by Carnegie Mellon University

Derivative Work - 1996, 1998-2000
Copyright 1996, 1998-2000 The Regents of the University of California

All Rights Reserved

Permission to use, copy, modify and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appears in all copies and
that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of CMU and The Regents of
the University of California not be used in advertising or publicity
pertaining to distribution of the software without specific written
permission.

CMU AND THE REGENTS OF THE UNIVERSITY OF CALIFORNIA DISCLAIM ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL CMU OR
THE REGENTS OF THE UNIVERSITY OF CALIFORNIA BE LIABLE FOR ANY SPECIAL,
INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING
FROM THE LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF
CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.


---- Part 2: Networks Associates Technology, Inc copyright notice (BSD) -----

Copyright (c) 2001-2003, Networks Associates Technology, Inc
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
```

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the Networks Associates Technology, Inc nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---- Part 3: Cambridge Broadband Ltd. copyright notice (BSD) -----

Portions of this code are copyright (c) 2001-2003, Cambridge Broadband Ltd. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* The name of Cambridge Broadband Ltd. may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---- Part 4: Sun Microsystems, Inc. copyright notice (BSD) -----

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Use is subject to license terms below.

This distribution may include materials developed by third parties.

Sun, Sun Microsystems, the Sun logo and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the Sun Microsystems, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---- Part 5: Sparta, Inc copyright notice (BSD) -----

Copyright (c) 2003-2004, Sparta, Inc
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Sparta, Inc nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

OpenSSL License

```
LICENSE ISSUES
==============

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts. Actually both licenses are BSD-style
Open Source licenses. In case of any license issues related to OpenSSL
please contact openssl-core@openssl.org.

OpenSSL License
---------------

/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit.  (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT “AS IS” AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
```

```
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com). This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

Original SSLeay License
-----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to. The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code. The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young’s, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *    Eric Young (eay@cryptsoft.com)"
 *    The word ’cryptographic’ can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG “AS IS” AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
```

```
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed. i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

Source: http://www.openssl.org/source/license.html

CookSwing License

The java tar public domain license

```
Public Domain

This work was autored by Timothy Gerard Endres, time@gjt.org.
This work has been placed into the public domain.
You are free to use this work in any way you wish.

DISCLAIMER

This software is provided AS-IS, with ABSOLUTELY NO WARRANTY. YOU ASSUME ALL
RESPONSIBILITY FOR ANY AND ALL CONSEQUENCES THAT MAY RESULT FROM THE USE OF
THIS SOFTWARE!
```

The MX4J License

MX4J is released under an Apache-style license. In practice this means that you can do almost anything you want with the code, including its use in commercial software. The actual text of the license is included below:

The MX4J License, Version 1.0

Copyright (c) 2001-2004 by the MX4J contributors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:
   "This product includes software developed by the MX4J project (http://mx4j.sourceforge.net)."
   Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The name "MX4J" must not be used to endorse or promote products derived from this software without prior written permission.
   For written permission, please contact
   biorn_steedom [at] users [dot] sourceforge [dot] net

5. Products derived from this software may not be called "MX4J", nor may "MX4J" appear in their name, without prior written permission of Simone Bordet.

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE MX4J CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==

This software consists of voluntary contributions made by many individuals on behalf of the MX4J project. For more information on MX4J, please see the MX4J website.

<http://mx4j.sourceforge.net>.