



**System Monitor**

# **V2S Editor User's Guide**

**FileNet System Monitor 3.7.0**

**FileNet Corporation**

# Table of Contents

<b>1. Copyright Notice .....</b>	<b>1</b>
Trademarks .....	1
Notice .....	1
<b>2. Notices .....</b>	<b>3</b>
<b>3. About this document .....</b>	<b>4</b>
Who Should Read This Guide .....	4
List of documents .....	4
General information .....	4
Where you find this guide .....	4
Typeface Conventions .....	4
Contacting FileNet Support.....	5
<b>4. V2S Editor Overview .....</b>	<b>6</b>
What is V2S? .....	6
The V2S Editor .....	6
<b>5. Installation .....</b>	<b>7</b>
Installation Requirements .....	7
Supported V2S Editor platforms .....	7
Installation .....	7
<b>6. Using V2S Editor .....</b>	<b>8</b>
V2S Editor Basics .....	8
Editing a V2S file .....	9
Loading a V2S file into the editor .....	9
Editing the V2 specification.....	10
Direct jump to class and subexpression definition .....	10
Editing expressions properties .....	11
Editing syntactically blocks .....	15
Deleting expressions .....	16
Adding new expressions and comments .....	16
Adding, modifying and deleting classes, subexpressions and global binds .....	17
Properties of V2 specification.....	18
Reloading the v2 specification after modifications .....	18
Checking the V2S file for correctness .....	18
Saving an edited V2S file .....	19
Creating a new V2S file .....	19
Using a logfile for creating and testing a V2 specification .....	19
Loading a logfile .....	19
Parsing the logfile with the v2 specification.....	19
Further log file parsing .....	20
Search for event .....	20
Load partial V2S.....	20
Creating an event description from the log file .....	21
<b>7. Additional command line programs .....</b>	<b>24</b>
v2scheck - Check a v2s file for correctness .....	24
Usage.....	24
v2s2baroc - A baroc file generator .....	24
Usage.....	24

<b>A. The v2 format .....</b>	<b>26</b>
Storage form .....	26
Identifiers .....	26
General design of the v2 format .....	26
Comments.....	26
Header .....	27
Global Variables .....	27
Automatically assigned variables .....	27
Variables to set timestamp .....	28
Classes and sub-expressions.....	28
Classes .....	28
Sub-expressions .....	29
Expressions .....	29
Matching types.....	30
Character Match (individual characters).....	30
Character Match (individual characters by ASCII code).....	30
Multi match (multiple match) .....	30
Constant string match .....	32
Subexpression match.....	32
Mandatory, optional and repetitive expressions.....	32
Mandatory expression.....	33
Optional expression .....	33
Optional repetitive expression.....	33
Group binding .....	34
Example of format file sna.v2s .....	35
<b>B. An example personal properties file.....</b>	<b>37</b>

# List of Screenshots

The V2S Editor main window .....	8
The filechooser dialog .....	9
The error message when loading an invalid v2s file .....	9
Jumping to a class definition by selecting it's name from the listbox.....	10
Editing expression properties: The sub expression *DISCARD* before the change.....	11
Editing expression properties: Opening the properties window. ....	??
Editing expression properties: The properties window before the change.....	12
Editing expression properties: Changing TIMESTAMPLINE to VERSIONLINE.....	12
Editing expression properties: The properties window after the sub expression VERSIONLINE has been selected. ....	13
Editing expression properties: The modified class deinition. ....	14
The v2s area popup window .....	15
The block options dialog.....	15
Deleting an expression block.....	16
The class properties dialog .....	17
The Subexpression check failed dialog .....	18
Loading a logfile .....	19
Parsing a logfile with the v2 specification .....	20
Creating an event description from the log file - step 1 .....	21
Creating an event description from the log file - step 2 .....	21
Creating an event description from the log file - step 3 .....	22
Creating an event description from the log file - step 4 .....	??
Using the v2scheck program.....	24
Using the v2s2baroc program .....	25

# Chapter 1. Copyright Notice

FileNet System Monitor

(September, 2006)

Copyright © 2000-2006 by CENIT AG Systemhaus, Germany, including this documentation and all software. All rights reserved. May only be used pursuant to a CENIT AG Systemhaus Software License Agreement.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of CENIT AG Systemhaus. CENIT AG Systemhaus grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the CENIT AG Systemhaus copyright notice. No other rights under copyright are granted without prior written permission of CENIT AG Systemhaus. The document is not intended for production and is furnished as is without warranty of any kind. *All warranties on this document are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.*

Note to U.S. Government Users Documentation related to restricted rights Use, duplication or disclosure is subject to restrictions set forth in GSA

## 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

This document contains information proprietary to FileNet Corporation (FileNet). Due to continuing product development, product specifications and capabilities are subject to change without notice. You may not disclose or use any proprietary information or reproduce or transmit any part of this document in any form or by any means, electronic or mechanical, for any purpose, without written permission from FileNet.

FileNet has made every effort to keep the information in this document current and accurate as of the date of publication or revision. However, FileNet does not guarantee or imply that this document is error free or accurate with regard to any particular specification. In no event will FileNet be liable for direct, indirect, special incidental, or consequential damages resulting from any defect in the documentation, even if advised of the possibility of such damages. No FileNet agent, dealer, or employee is authorized to make any modification, extension, or addition to the above statements.

FileNet may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Furnishing this document does not provide any license to these patents, trademarks, copyrights, or other intellectual property.

Please take a few moments to read the *End User License Agreement* on the FileNet System Monitor 3.7.0 documentation CD. By installing the FileNet System Monitor 3.7.0 software, the customer agrees to be bound by the terms of this agreement. FileNet System Monitor, copyright-protected by CENIT AG Systemhaus, is licensed and rebranded by FileNet Corporation. FileNet, ValueNet, Visual WorkFlo, and OSAR are registered trademarks of FileNet Corporation. Document Warehouse and UserNet are trademarks of FileNet Corporation. All other product and brand names are trademarks or registered trademarks of their respective companies. See the *Centera License Agreement* for copyright information pertaining to EMC Centera.

Copyright © 1984, 2006 FileNet Corporation. All rights reserved.

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) (<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 appear like *this*

**userinput** Values that the user must provide appear like **this**

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

*gui label* Names of windows, dialogs, and other controls appear like *this*

*Program listings* appear like this:

```
001 # a program listing
002 echo "This is an example program listing (shell script) with nothing bu ✓
... t an extremely long echo command"
003 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. V2S Editor Overview

## What is V2S?

V2 is a language for logfile description. It is used by the CALA's v2 format filter to recognize events from complex logfiles and to transform them into the CALA event format FIR.

A V2 syntax description file (V2S) is a text file containing a logfile description in the v2 format.

For further information about the v2 format filter and FIRs refer to the FSM CALA User's Guide, the V2 syntax is described later.

## The V2S Editor

The V2S Editor is a graphical tool, that helps you to create and modify V2 syntax description files. You can also test V2S files against logfiles to see which events are recognized and which FIRs would be created if CALA processes this file.

The V2S Editor also provides a BAROC export feature for CALA/Tivoli integration.

# Chapter 5. Installation

## Installation Requirements

### Supported V2S Editor platforms

For detailed information about supported V2S Editor platforms and required Java JRE / JDK versions check the latest release notes

## Installation

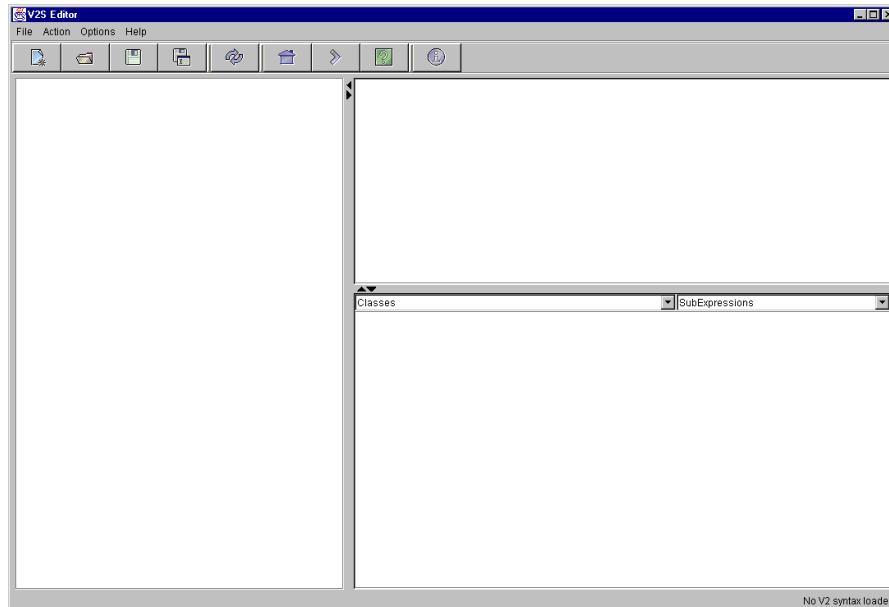
The V2S Editor can be installed by using the CALA's Tivoli integration for detailed information refer to the CALA User's Guide.

To install the V2S Editor manually, the program **install\_cala.sh** from the CALA installation CD can be used. This program is described in the appendix of the CALA Users's Guide.

# Chapter 6. Using V2S Editor

## V2S Editor Basics

After starting the V2S Editor you get the following window:



*The V2S Editor main window*

**Note:** The screenshot in this guide were taken from a V2S Editor with the Windows look & feel configured.

If you are using any other look & feel manager, the look of your user interface may differ in some details.

The V2S Editor main window consists of three parts:

- the logfile area (left)
- the result area (upper right)
- and the v2s area (lower right).

The logfile, for which a v2 specification should be created, can be load into the logfile area by choosing the Open logfile item from the File menu. The logfile area is writable, so that events can be added manually.

The result area is a read-only area which is used to display the resulting FIR when a v2 specification is tested against a logfile.

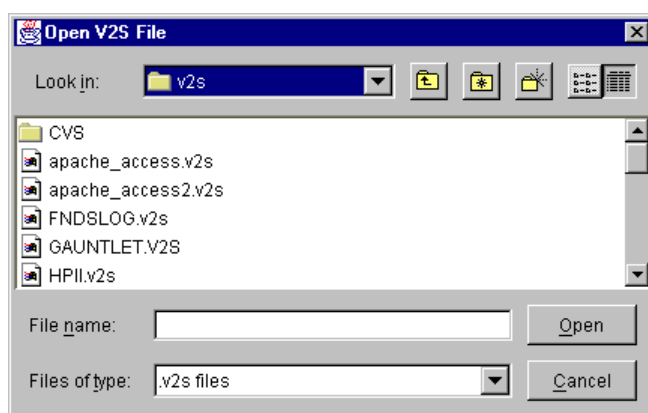
The v2s area is an editable text area which holds the v2 specification to be edited. It can be filled with an existing v2s file by choosing the item Open V2S from the File menu. The v2s area also contains two list boxes for direct access to class and subexpression definitions.

The size of the three areas can be changed by moving the separators with the mouse pointer.

## Editing a V2S file

### Loading a V2S file into the editor

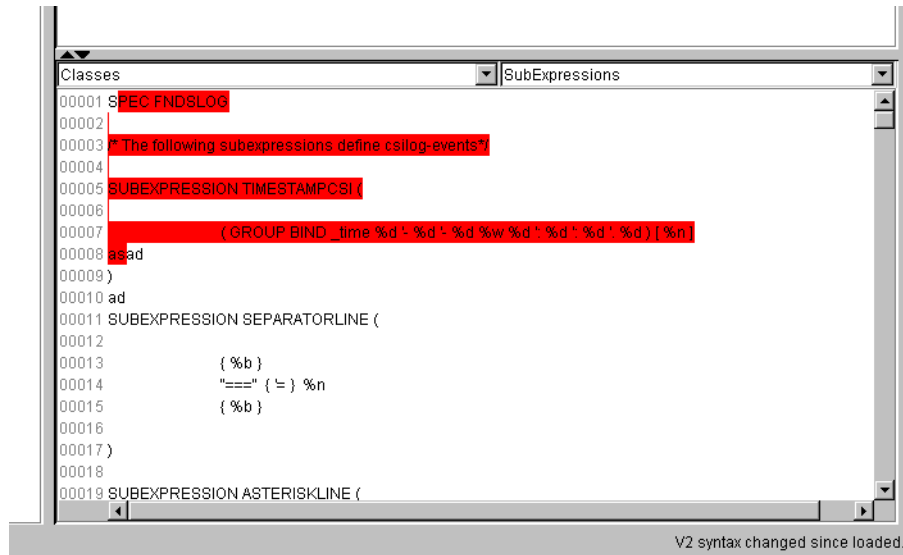
To load a v2s file, choose Open V2S from the File menu or press the corresponding toolbar button and a file dialog appears.



*The filechooser dialog*

When a v2s file was chosen and the Open button is pressed, the file dialog disappears and the chosen file is loaded into the v2s area, where it can be edited.

If the loaded v2s file is syntactically incorrect, an error dialog is displayed and the region where the error is suspected is highlighted.



*The error message when loading an invalid v2s file*

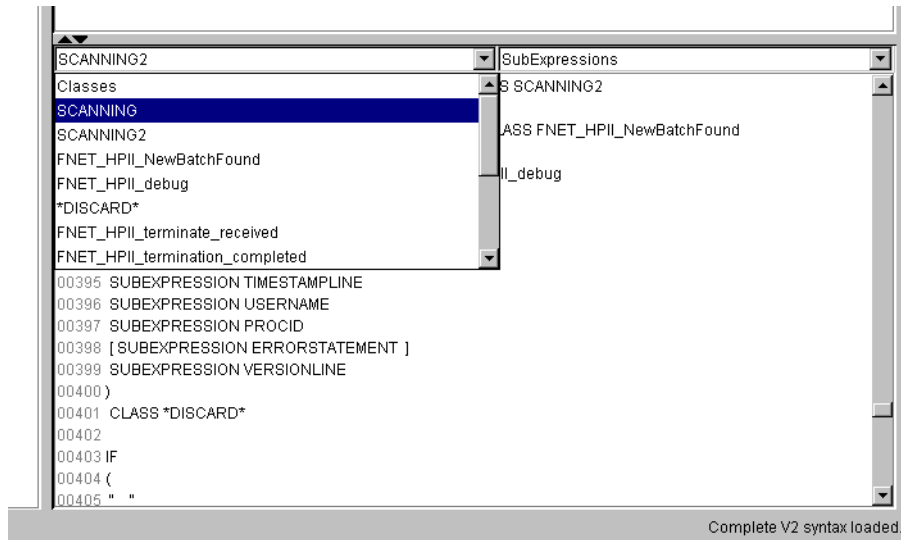
After the error has been corrected, the new specification has to be loaded into the v2s parser by choosing Reload V2 specification from the Action menu or by pressing the reload button from the toolbar.

## Editing the V2 specification

An experienced user may like to use the V2S Editor like a normal text editor to create and modify v2s files manually, for less experienced users all modifications can also be done using the graphical user interface explained below.

### Direct jump to class and subexpression definition

You can directly jump to the definition of a class or subexpression by selecting the corresponding entry in the listboxes above the v2s text area. The text view is scrolled to the selected class or subexpression definition .



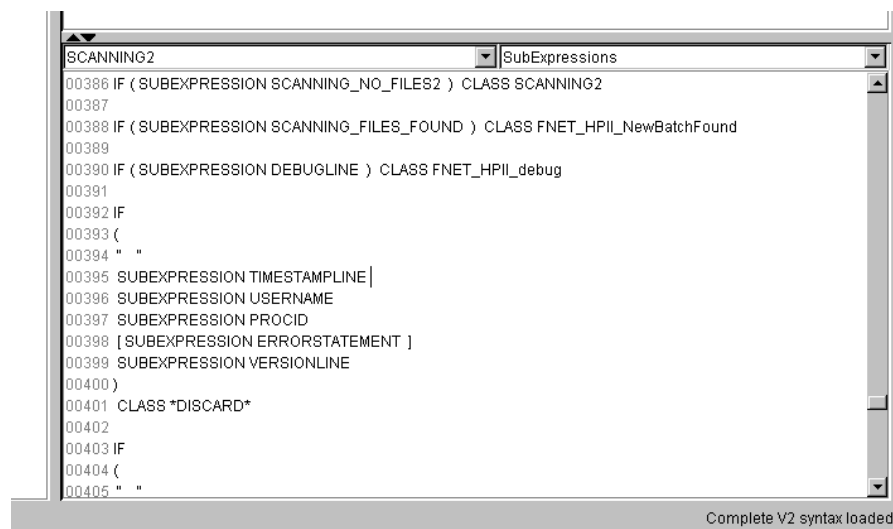
*Jumping to a class definition by selecting it's name from the listbox*

## Editing expressions properties

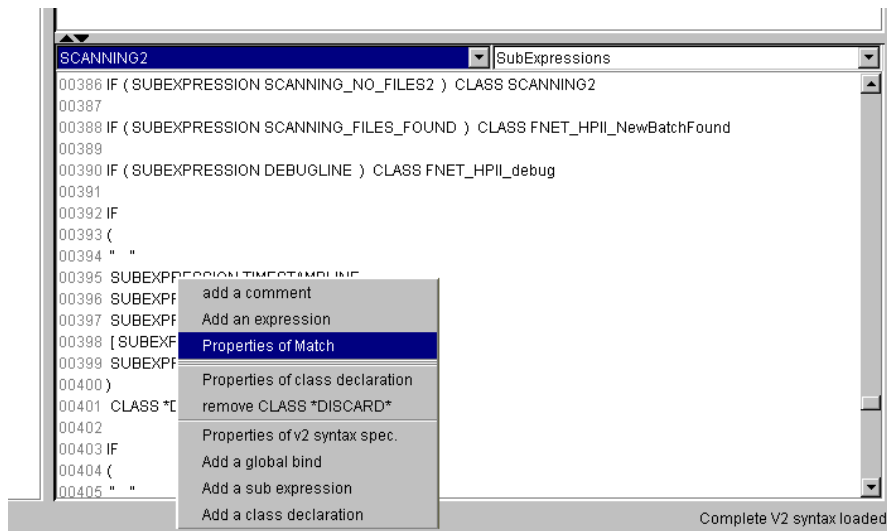
The properties of any expression can be edited by selecting the Properties item in the popup-menu. The popup-menu appears when the mouse is placed over an expression and the right mouse button is pressed.

The contents of the properties window depends on the selected expression. If any property has been changed and the ok button is pressed, the properties windows disappears and the expression is changed.

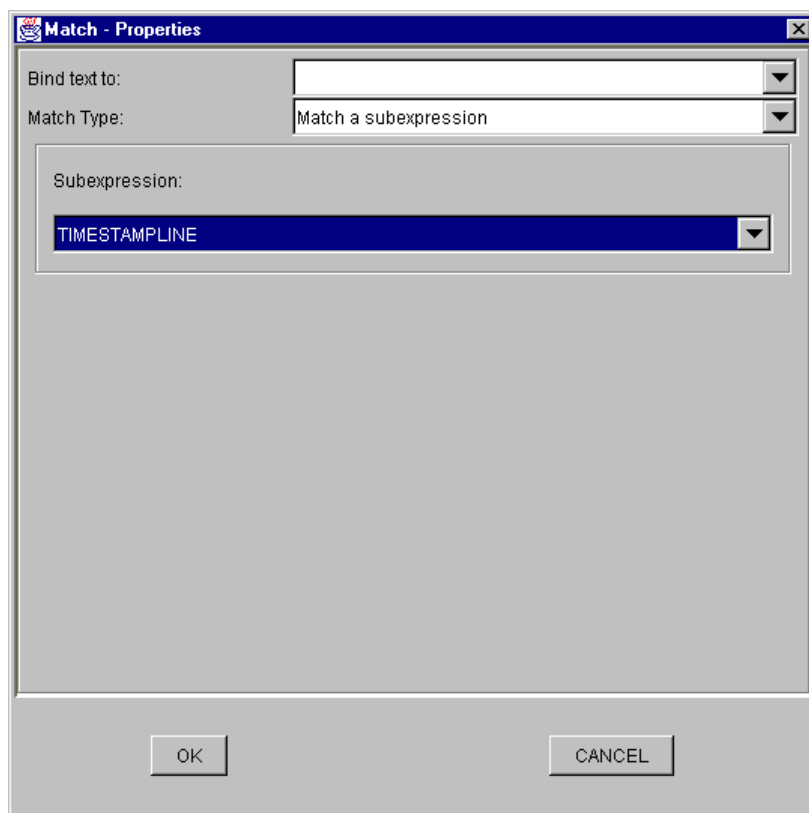
The screenshots below shows an example property change: the match for the subexpression `TIMESTAMPLINE` is changed into a match for the subexpression `VERSIONLINE`.



*Editing expression properties: The sub expression \*DISCARD\* before the change*

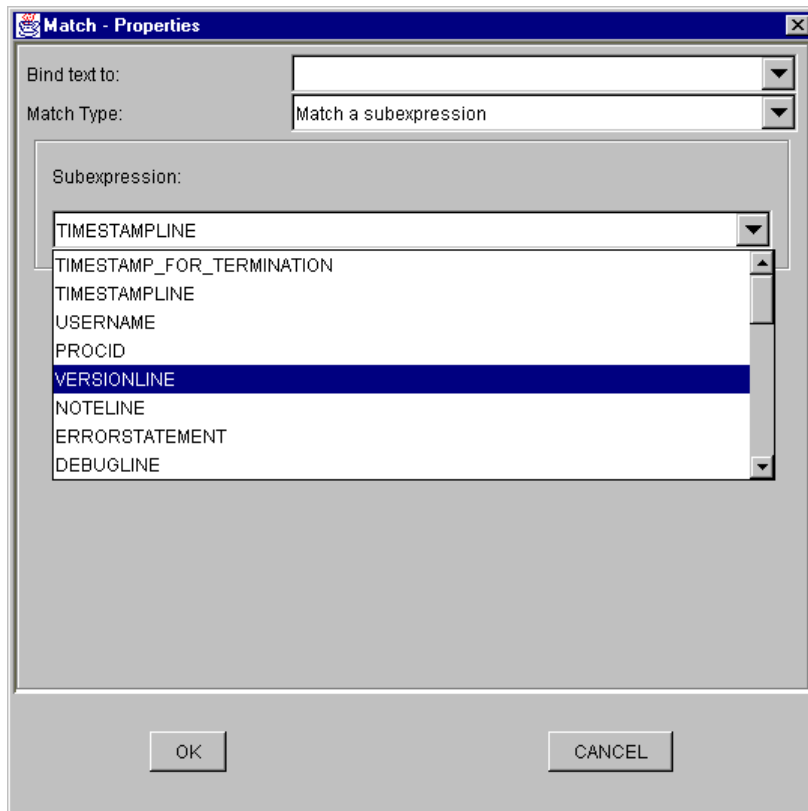


*Editing expression properties: Opening the properties window.*

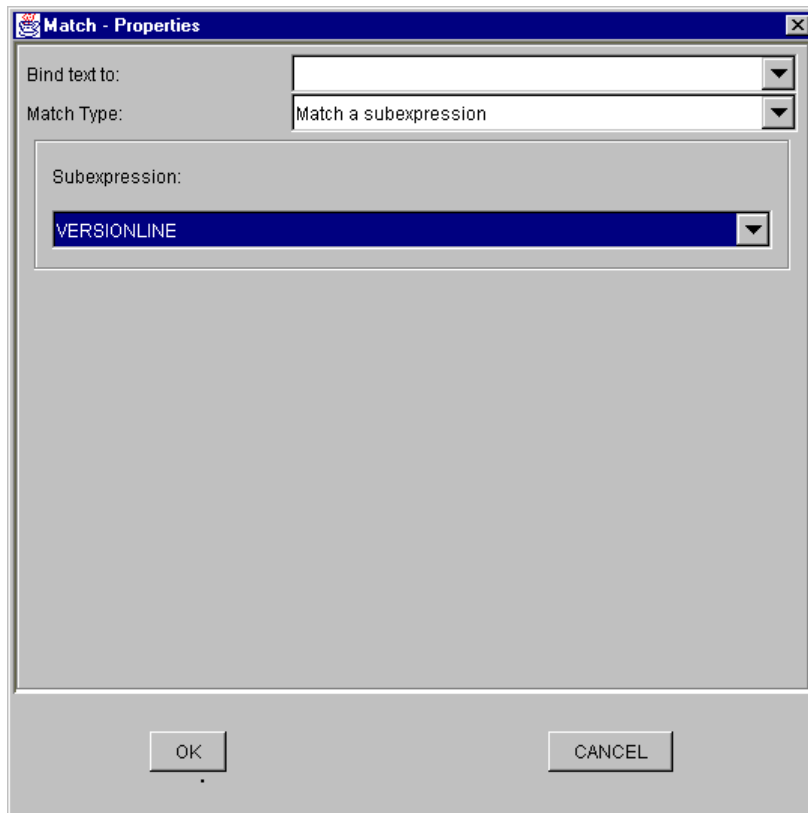


*Editing expression properties: The properties window before the change.*



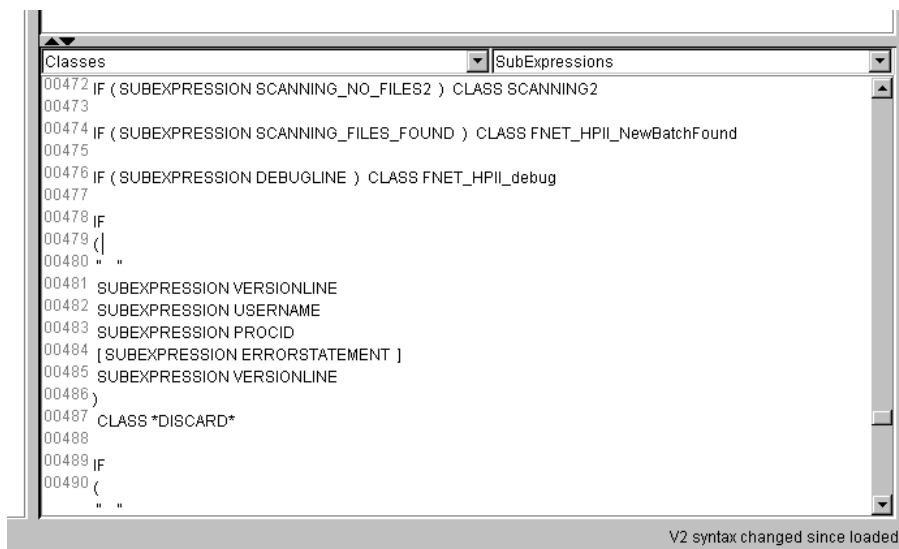


*Editing expression properties: Changing `TIMESTAMPLINE` to `VERSIONLINE`*



*Editing expression properties: The properties window after the sub expression `VERSIONLINE` has been selected.*

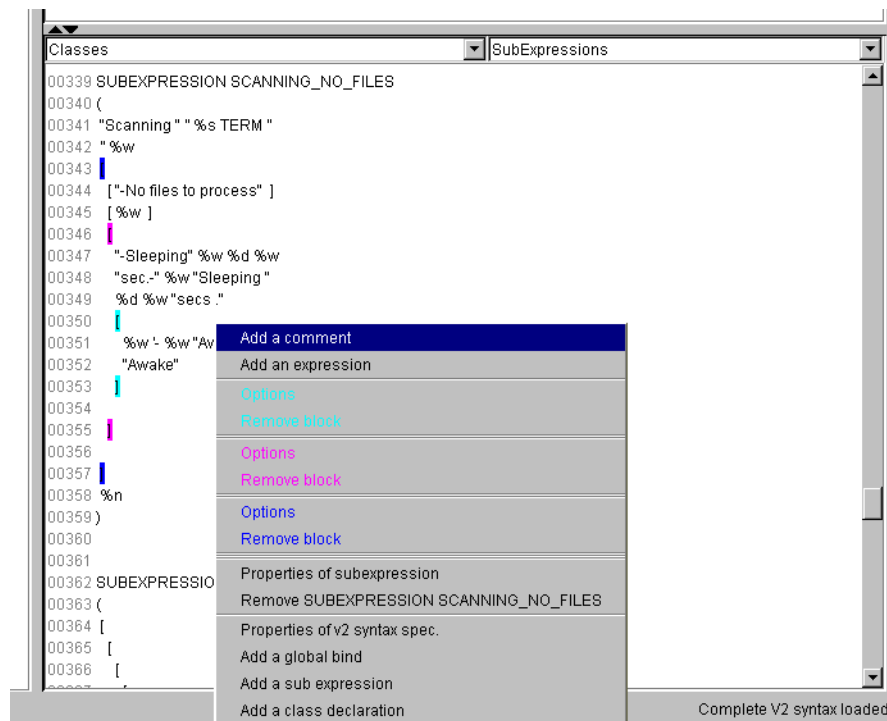
After pressing OK, the modifications are applied to the v2 specification.



*Editing expression properties: The modified class definition.*

## Editing syntactically blocks

When the popup-menu is shown for an expression which is within one or more syntactically blocks, there are two menu entries for each of these blocks. The brackets enclosing each of these blocks are marked with different colors and so are the corresponding menu entries. This simplifies finding the correct menu entry for the block to modify.

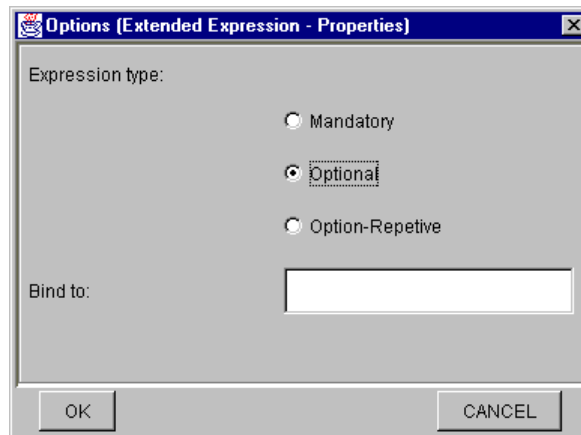


*The v2s area popup window*

There are two menu items for each block:

- Options
- Remove block

Options is chosen, a dialog similar to the properties dialog of expressions appears, where the type of options and a optional bind slot (a FIR slot which receives the complete text matched within this block) can be set.



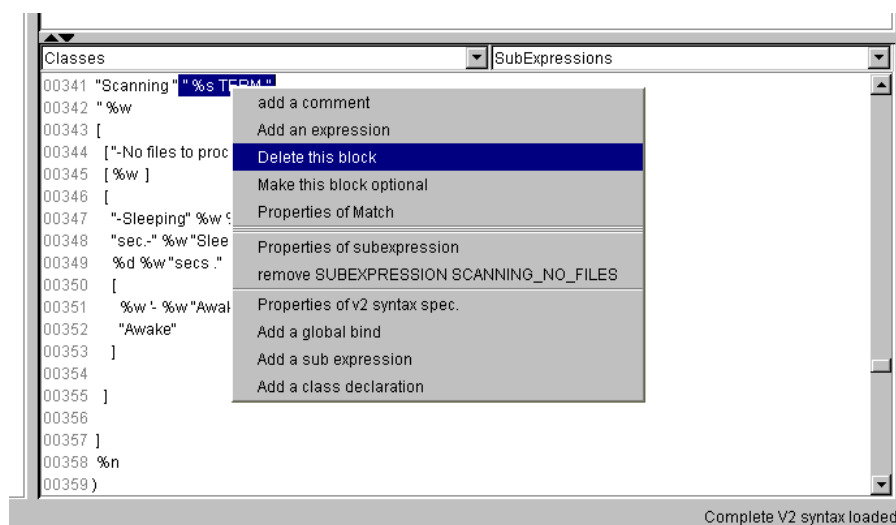
*The block options dialog*

If the Remove block item is selected, a confirmation dialog appears and the corresponding block is deleted (if confirmed).

A new block can be inserted by marking one or more expressions and choosing the popup-menu entry Make this block optional (which only appears if a block is marked). If done so, the Options dialog for the new block is displayed and if OK is pressed, the selected expressions are put into a block with the chosen properties.

## Deleting expressions

One or more expressions can be deleted by marking them and choosing Delete this block from the popup-menu.



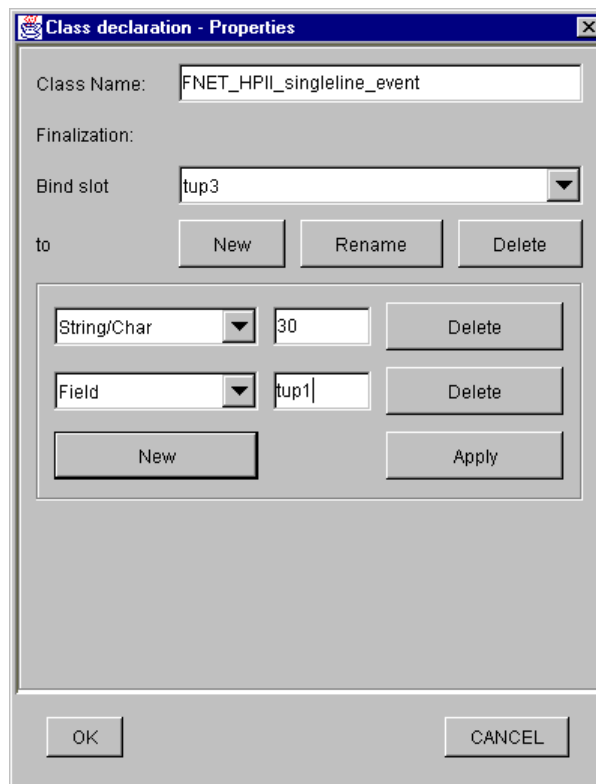
*Deleting an expression block*

## Adding new expressions and comments

New expressions or comments can be added by selecting the popup-menu items Add an expression or Add a comment . If one of these items has been selected, a properties menu appears and the new expression or comment is inserted at the cursor position.

## Adding, modifying and deleting classes, subexpressions and global binds

The property dialog of a class, subexpression or global bind is available in the popup-menu of each expression belonging to the class, subexpression or global bind definition (see screenshots above).



*The class properties dialog*

The classes properties dialog has one specialty: With listbox Bind slot to you can choose the slots definition you want to edit in the frame below. If a new slot definition has to be added, the New button must be selected.

The bind-to frame contains a list of strings and fields which are bound to the slot. If any changes are made in this frame, they have to be committed by pressing the Apply button before changing the selection of the listbox. To change the class declaration permanently, the OK button of the dialog windows must be pressed.

The menu entry for removing a class, subexpression or global bind is always available, when the properties dialog for this definition is available.

The popup-menu items for adding new classes, subexpressions and global binds are available everywhere in the v2s file. After choosing one of these items, a window with the existing classes,

subexpressions or global binds appears where the position for the new definition can be set (before or after the chosen definition).

If this has been selected, a properties dialog for the new definition appears. If it has been committed with OK, the new definition is inserted into the v2 specification.

A new class or subexpression definition is initialized with a default expression, which can be changed by the user afterwards.

## Properties of V2 specification

The menu item Properties of v2 specification is available everywhere in the v2 specification area. The only property which can be changed is the specification name.

## Reloading the v2 specification after modifications

The menu item Reload V2S or the toolbars Reload button tells the V2S Editor to update its internal information about the loaded v2 specification.

If the v2 specification has been changed, a reload is needed before any tests on this specification are performed. If a reload is needed, the menu item and toolbar button are enabled, otherwise they are disabled.

After manually changing the v2 specification a reload is also needed to verify the syntax and update the markers for the context menus.

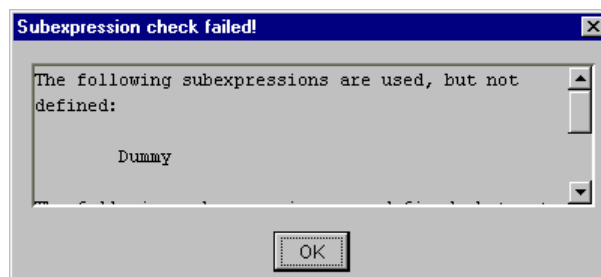
## Checking the V2S file for correctness

The syntax of the v2 specification is checked each time a new file is loaded or the Reload button is pressed (see [Reloading the v2 specification after modifications](#)).

If the syntax check has been passed, the `v2fmtfilt` will accept the v2s file as a correct v2 format syntax.

The V2SEdit gives you an additional test feature to check for unused or undefined subexpressions. To perform this check, choose the item Check subexpression from the Action menu.

If any error is found, a information window is displayed (see screenshot below), if the v2s file is ok, the message `Subexpression check ok` is displayed in the status bar on the lower left of the V2S Editor window.



*The Subexpression check failed dialog*

## Saving an edited V2S file

To save an edited V2S file, choose one of the menu items Save V2S or Save V2S as from the File menu or one of the corresponding toolbar buttons.

## Creating a new V2S file

A new V2S file can be created by selecting New V2S from the File menu. A properties dialog for the new V2 specification appears and, if committed, a new V2 specification is loaded into the v2s area.

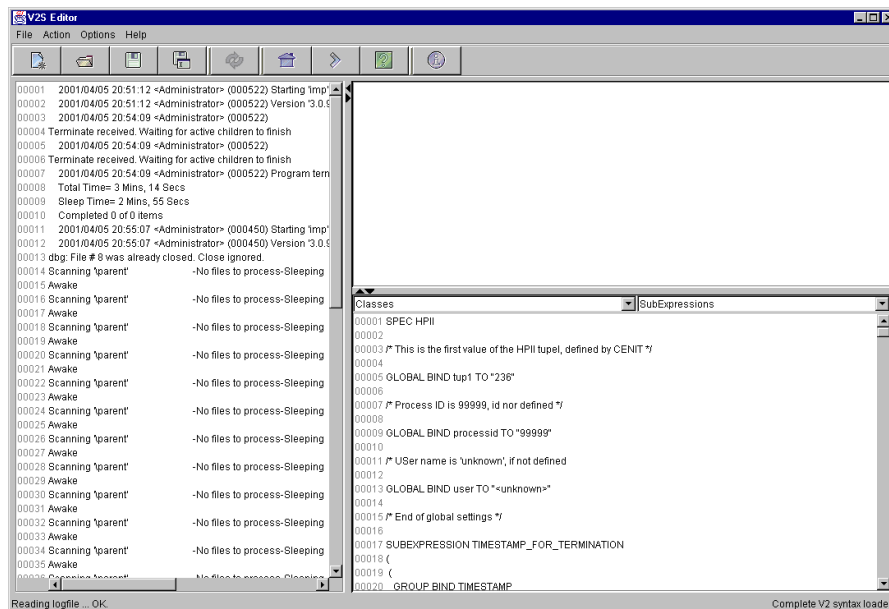
Each new v2 specification has one standard class for Unknown events. In most cases you should leave this class in the specification (as the last class declaration) to match events which are not matched by any other event class.

## Using a logfile for creating and testing a V2 specification

### Loading a logfile

To load a logfile the menu item Open Logfile in the File menu must be selected.

The logfile is loaded into the logfile area (left) and can be edited e.g. for removing or duplicating some events.



Loading a logfile

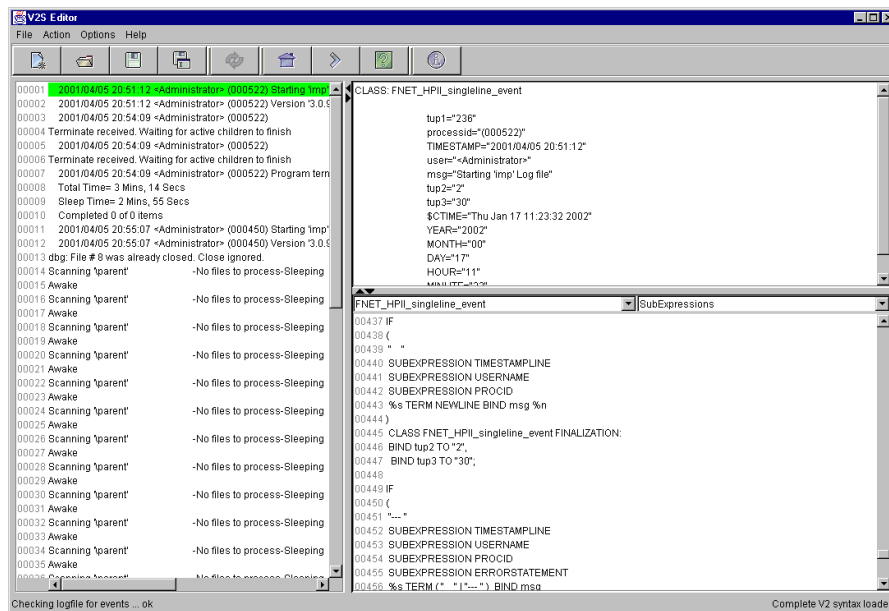
## Parsing the logfile with the v2 specification

If a v2 specification and a logfile is loaded, the v2 specification can be tested against the logfile.

If Find first event in the Action menu is selected, the v2 parser tries to match the logfile from the beginning and displays the matched event in the result area. The event is marked in the logfile and the v2s area is scrolled to the definition of the matched class.

When Find next event is chosen, the parser starts event processing at the cursor position. The find actions places the cursor after the last found event, but one can move it to any other position.

If no event could be found, the message `no more events` is displayed in the status line below the logfile area.



*Parsing a logfile with the v2 specification*

To get more information about the parsing process, select Show parse log from the Action menu and the result area shows a detailed trace from the last parsing action.

This information can be helpful to find errors in a v2 specification or simply to understand how v2s works.

## Further log file parsing

The V2S Editor has additional features for log file parsing, which can be used for debugging

### Search for event

The Search for event menu item in the Action menu is similar to the Find next event , but does also find events which don t start directly at the cursor position. The non-matching text is skipped.



## Load partial V2S

This option allows only a part of class definitions to be loaded into the parser. This can e.g. be used in combination with Search for event to find all events of a special class.

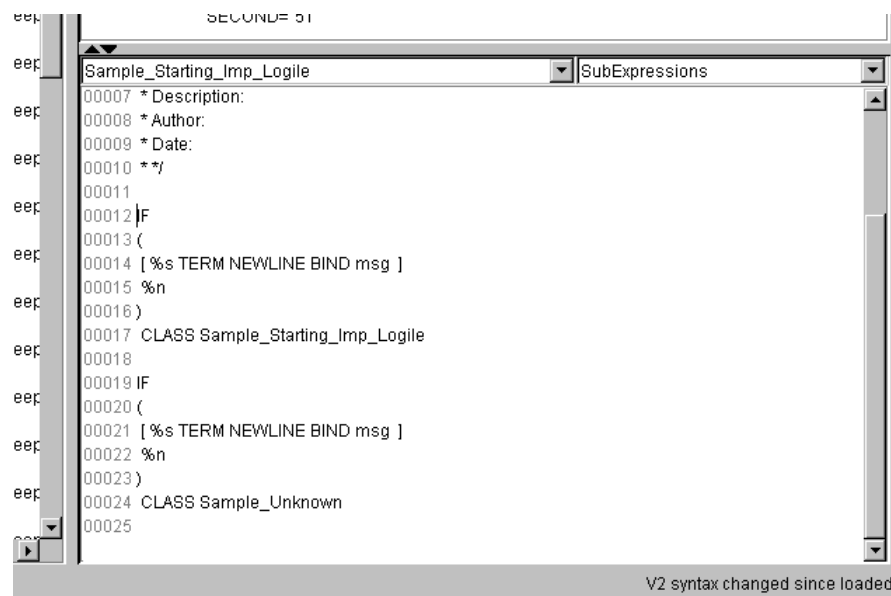
After selecting this menu item, a dialog box appears, where the classes to load can be selected. To load the full specification back into the parser, the v2 specification has to be reloaded (menu item Reload V2S).

## Creating an event description from the log file

The V2S Editor helps you to create new event class descriptions from a log file.

First a new event class has to be created, this can be done by choosing the item Add a class declaration in the v2s area popup menu. The new event class is created with the default expression.

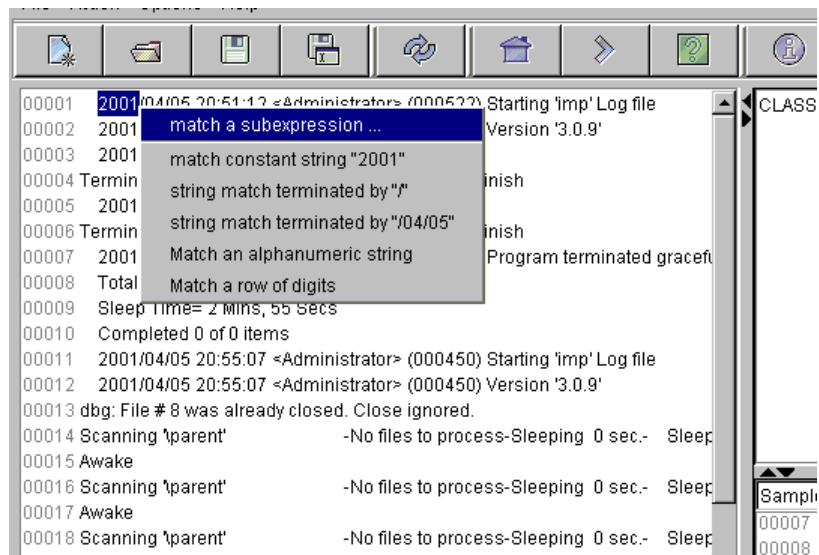
(See class `Sample_Starting_Imp_Logfile` in the screenshot)



*Creating an event description from the log file - step 1*

Moving the cursor has behind the opening brace of this class definition tells the editor, that new expression have to be placed there.

Now take a look at the log file area and mark the first field of the event you want to define. If done so, the popup menu shows you a list of expression the marked text would match with.

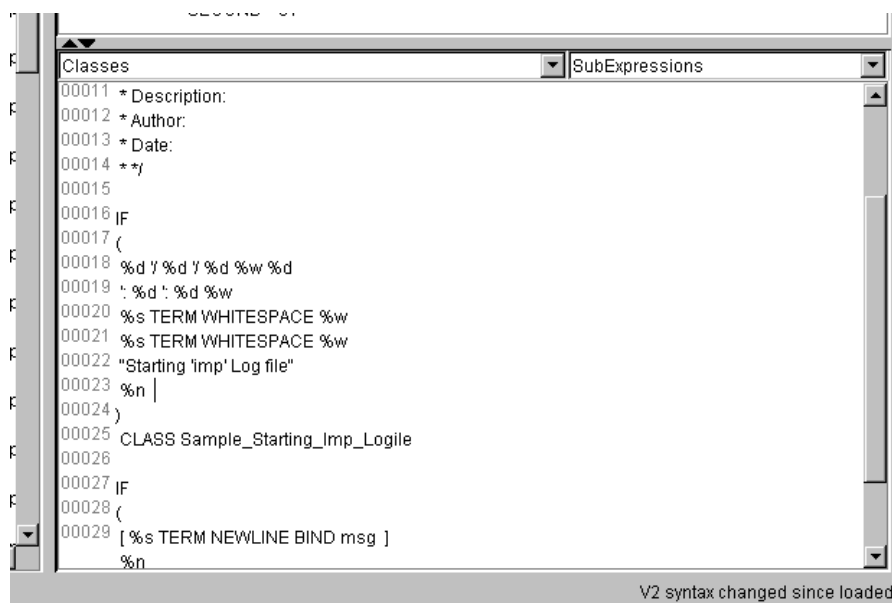


Creating an event description from the log file - step 2

After choosing the text type, the v2 representation of this text is inserted at the cursor position in the v2s area.

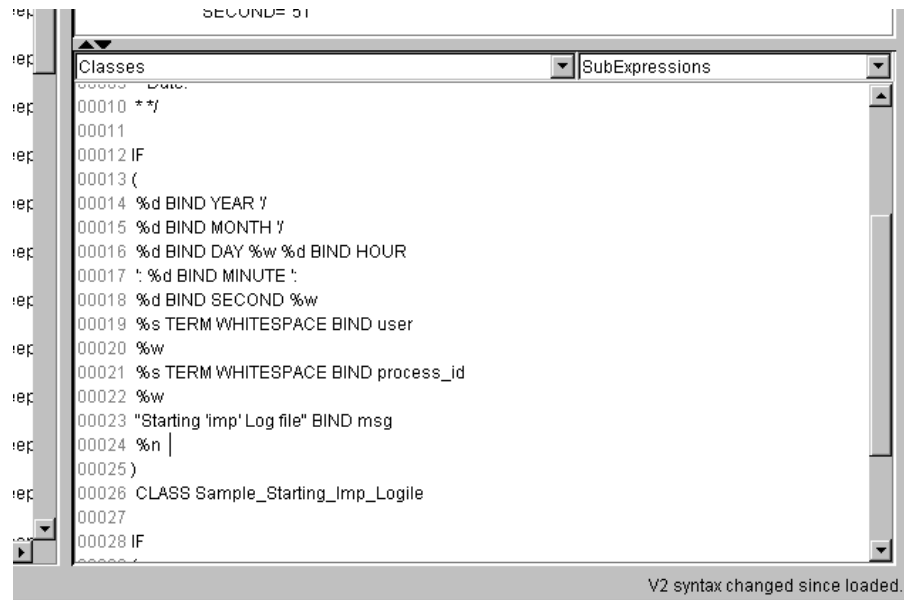
There is one special item in this popup menu the item Match a subexpression checks the marked text against all defined subexpressions and gives you a list of the ones which match. After selecting the a subexpression, a match for it is inserted into the v2s file.

Using this feature, the complete class definition can be build. (Don't forget to remove the standard expression from the definition, if it is no longer needed.)



Creating an event description from the log file - step 3

Finally there may be some options and slot assignments to make and the definition is complete.



*Creating an event description from the log file - step 4*

# Chapter 7. Additional command line programs

There are two additional command line programs shipped with the V2SEdit:

- **v2scheck**
- **v2s2baroc**

These programs are available as Unix shell scripts (`v2scheck.sh`, `v2s2baroc.sh`) and Windows batch files (`v2scheck.bat`, `v2s2baroc.bat`). This document describes the handling of the Unix shell scripts, but the usage of the batch files is exactly the same.

On all systems a java runtime environment  $\geq$  JRE 1.3 must be reachable over the `PATH` environment variable to execute the programs.

## v2scheck - Check a v2s file for correctness

The **v2scheck** program performs a syntax check and a subexpression check for a given v2s format file. If any check fails, detailed error messages are printed on the screen.

### Usage

The **v2scheck** program takes only one argument: the name of the file to check.



```
Solaris on CCC4
[cccc4:/export/home/matysiak/v2sedit]:v2scheck.sh data/v2s/apache_access_subexpression_error.v2s
*****
**
**      V2SCheck is part of the CENIT Advanced Logfile Adapter      **
**
**      (c) 2001-2002 CENIT AG Systemhaus                          **
**
*****

Reading V2S file "data/v2s/apache_access_subexpression_error.v2s" ... OK
Parsing V2S ... OK
Checking Subexpressions ... ERROR

The following subexpressions are used, but not defined:

    Dummy

The following subexpressions are defined, but not used:

    Dummy2

[cccc4:/export/home/matysiak/v2sedit]:
```

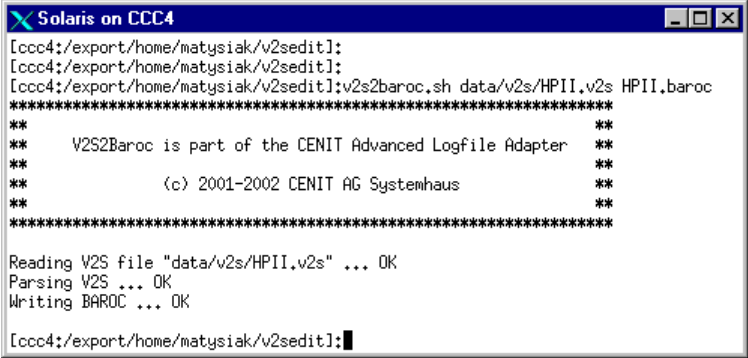
*Using the v2scheck program*

## v2s2baroc - A baroc file generator

This program creates a Tivoli baroc file from a v2s format file. The baroc files contains a event description to be loaded into the T/EC.

## Usage

The **v2s2baroc** program takes two arguments: the name of the v2s file and the name of the baroc file to be created.



```
Solaris on CCC4
[ccc4:/export/home/matysiak/v2sedit]:
[ccc4:/export/home/matysiak/v2sedit]:
[ccc4:/export/home/matysiak/v2sedit]:v2s2baroc.sh data/v2s/HP11.v2s HP11.baroc
*****
**      V2S2Baroc is part of the CENIT Advanced Logfile Adapter      **
**                               **
**      (c) 2001-2002 CENIT AG Systemhaus                          **
**                               **
*****
Reading V2S file "data/v2s/HP11.v2s" ... OK
Parsing V2S ... OK
Writing BAROC ... OK
[ccc4:/export/home/matysiak/v2sedit]:
```

*Using the v2s2baroc program*

# 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++.

```
001 /* <comment> */
002 // <comment terminated by new line>
```

**Example A-1. Example of comments in v2s**

## Header

The header has the format:

```
001 SPEC <name>
```

**Figure A-1. Format of v2s spec expression**

<name> is any identifier for the format specification.

```
001 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.

```
001 GLOBAL BIND <slot name> TO "<string>"
```

**Figure A-2. Format of v2s global bind expression**

```
001 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. These 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 <code>TIME_POSTFIX</code> to any value switches to 12-hour-mode. Sets time to P.M. if <code>TIME_POSTFIX</code> 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.

```
001 IF expression CLASS name [ FINALIZATION: BIND <slot> TO <any sequence of ✓
... V2S expressions > ]
```

**Figure A-3. Format of v2s class expression**

```
001 IF (
002 SUBEXPRESSION TIMESTAMP
003 "myprocess shut down"
```



```
004 ) 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.

```
001 SUBEXPRESSION name ( expression )
```

#### **Figure A-4. Format of v2s subexpression definition**

```
001 SUBEXPRESSION IPADDREXPR ( GROUP BIND IPADDR %d { '. %d } )
```

#### **Example A-5. Example of a v2s subexpression definition**

The "Macro" is called using

```
001 SUBEXPRESSION name
```

#### **Figure A-5. Format of v2s subexpression call**

```
001 "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.

```
001 \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).

```
001 %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

```
001 "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:

```
001 (" 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

```
001 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

This 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.

```
001 '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.

```
001 ( 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:

```
001 SPEC SNA
002
003 SUBEXPRESSION TIMESTAMP (
004     %d BIND HOUR
005     ':
006     %d BIND MINUTE
007     ':
008     %d BIND SECOND
009     " "
010     %a BIND TIMEZONE
011     " "
012     %d BIND DAY
013     " "
014     %a BIND MONTH
015     " "
016     %d BIND YEAR
017 )
018
019 SUBEXPRESSION TIMESTAMPLINE (
020     SUBEXPRESSION TIMESTAMP
021     " "
022     %d BIND CODE1
023     '-
024     %d BIND CODE2
025     '(
026     %d BIND CODE3
027     '-
028     %d BIND CODE4
029     ')
030     " "
031     %a BIND CODE5
032     " "
033     '(
034     %a BIND CODE6
035     ')
036     %n
037 )
038
039 IF (
040     "===== Log file initialised " SUBEXPRESSION TIMESTAMP " ===== ✓
041     ... =====" %n
042 ) CLASS LOGINIT
043
044 IF (
045     SUBEXPRESSION TIMESTAMPLINE
046     "Abnormal UNBIND request received" %n
047     "Sense code" %w '= %w %a BIND SENSECODE %n
048     "Local LU name" %w '= %w ( GROUP BIND LOCALLU %a '. %a ) %n
049     "Partner LU name" %w '= %w ( GROUP BIND PARTNERLU %a '. %a ) %n
050     "Mode name" %w '= %w %a BIND MODENAME %n
051     "UNBIND RU :" %n ( GROUP BIND UNBINDRU %a { " " %a } ) %n
052 ) CLASS ABNORMALUNBIND FINALIZATION:
053     BIND msg TO "Abnormal UNBIND request received " + SENSECODE,
054     BIND SENSECODE TO NOTHING;
```

**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. An example personal properties file

```
001 // Filename: V2SEdit_personal.properties
002 //
003 // Date: 12.12.2001
004 //
005 // Personal settings for V2S editor
006 //
007 // The settings in this file are user definable.
008 // Things which are not configured here (like button and dialog texts)
009 // are defined in V2SEdit.properties file or any of it's parents.
010 // The V2SEdit.properties file also contains a default value for each
011 // of the following parameters, which is used if the parameter is
012 // not configured here.
013 //
014 // The settings in this file are given like this:
015 //
016 // [property name]=[value]
017 //
018 // For properties expecting color values, the following colors are
019 // supported:
020 //
021 // black
022 // blue
023 // cyan
024 // darkGray
025 // gray
026 // green
027 // lightGray
028 // magenta
029 // orange
030 // pink
031 // red
032 // white
033 // yellow
034 //
035
036 // All lines starting with // are comments.
037
038 // Look and Feel: The user interface manager
039 //
040 // 0 = use system default (motif on Unix, windows on Windows)
041 // 1 = motif
042 // 2 = windows (only in MS operating systems)
043 // 3 = metal
044 //
045 // default: 0
046 v2sedit.menu.options.lookandfeel.defaultmanager=3
047
048 // Line number mode at startup
049 //
050 // The line no. mode can be switched on for each
051 // of the text areas in the main window.
052 // Setting the property to 0 turns the line no. mode off,
053 // each other value turns it on.
```

```

054 //
055 // Line no. mode is switched on for logfile and v2s by default.
056 //
057 v2sedit.menu.options.linenumbers.logfile.startup=1
058 v2sedit.menu.options.linenumbers.v2s.startup=1
059 v2sedit.menu.options.linenumbers.result.startup=0
060
061 // Text font and size for the logfile area in the main window.
062 //
063 // Default: SansSerif, 12
064 //
065 v2sedit.logarea.font=SansSerif
066 v2sedit.logarea.fontsize=12
067
068 // Highlight color for found events.
069 v2sedit.logarea.eventcolor=green
070
071 // Text font and size for the v2s area in the main window.
072 //
073 // Default: SansSerif, 12
074 //
075 v2sedit.v2sarea.font=SansSerif
076 v2sedit.v2sarea.fontsize=12
077
078 // Highlight color for errors in v2 syntax
079 v2sedit.v2sarea.errorcolor=red
080
081 // Show subexpression as tooltip? (0=no, 1=yes)
082 v2sedit.v2sarea.showtooltips=1
083
084 // Text font and size for the result area.
085 // (This is the area where the FIRs are displayed.)
086 //
087 // Default: SansSerif, 12
088 //
089 v2sedit.resultarea.font=SansSerif
090 v2sedit.resultarea.fontsize=12
091
092 // Logfile path and extensions
093 v2sedit.openlogfile.filechooser.path=data/logs
094 v2sedit.openlogfile.filechooser.extension=.log
095
096 // V2S file path and extensions
097 v2sedit.openv2sfile.filechooser.extension=.v2s
098 v2sedit.openv2sfile.filechooser.path=data/v2s
099
100 // Baroc file path and extensions
101 v2sedit.exportbaroc.filechooser.extension=.baroc
102 v2sedit.exportbaroc.filechooser.path=data/baroc
103
104 // baroc export: header and footer
105 //
106 // This text is added at the beginning (header) or end (footer) of
107 // the BAROC file. {0} is the name of the v2 syntax.
108 //
109 v2sspec.barocexport.header=//\n// Filename:\n//\n//\t {0}.baroc\n//\n// ✓
... Language:\n//\n//\t Tivoli BAROC\n//\n// Description:\n//\n//\t Create ✓
... d by the V2 format editor.\n//\n//\n\n
110 v2sspec.barocexport.footer=

```

```

111
112 // Display an "are you sure" dialog before exit? (0=no, 1=yes)
113 v2sedit.exit.displaydialog=0
114
115 // split lines in v2s windows at this length
116 v2sspec.splitLinesAtLength=30
117
118 // Creating a new V2S -> default names
119 v2sedit.specdecl.defaultname=Unknown
120
121 // Comment at the beginning of the syntax
122 // {0} is the syntax name
123 v2sedit.specdecl.defaultheadcomment=V2 definition {0}\n *\n * (c) 2002 ✓
... CENIT AG Systemhaus, Stuttgart (Germany)\n *\n * Description:\n * Autho ✓
... r:\n * Date:\n *
124
125 // Default class name
126 //{0} is the syntax name
127 v2sedit.specdecl.addclass.defaultclass={0}_Unknown
128
129 // Default subexpression name
130 //{0} is the syntax name
131 v2sedit.specdecl.addsubexpression.defaultexpression={0}_UNKNOWN
132
133 // remove redundant mandatory expressions?
134 //
135 // If set to 1, the expression
136 //
137 // IF
138 // (
139 //   ( %n )
140 //   %s TERM NEWLINE
141 // )
142 // CLASS mini_Unknown
143 //
144 // is replaced by
145 //
146 // IF
147 // (
148 //   %n
149 //   %s TERM NEWLINE
150 // )
151 // CLASS mini_Unknown
152 //
153 // (the mandatory expression is no longer needed)
154 v2sspec.extendedexpression.removeredundant=1
155
156
157 // colors for colored popup menu in v2s area
158 // 1 = inner
159 // 10 = outer
160 v2sspec.extendedexpression.color.1=blue
161 v2sspec.extendedexpression.color.2=magenta
162 v2sspec.extendedexpression.color.3=cyan
163 v2sspec.extendedexpression.color.4=green
164 v2sspec.extendedexpression.color.5=yellow
165 v2sspec.extendedexpression.color.6=pink
166 v2sspec.extendedexpression.color.7=red
167 v2sspec.extendedexpression.color.8=orange

```

```
168 v2sspec.extendedexpression.color.9=darkGray
169 v2sspec.extendedexpression.color.10=gray
```

**Example B-1. An example personal properties file**