IBM Enterprise Content Management System Monitor
Version 5.2

V2S Editor User's Guide

SC27-4910-04

IBM Enterprise Content Management System Monitor
Version 5.2

V2S Editor User's Guide

SC27-4910-04

Before using this information and the product it supports, read the information in "Notices" at the end of this document.

This edition applies to version 5, release 2, modification 0 of IBM Enterprise Content Management System Monitor (product number 5724R91) and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

Preface

About this document

Who should read this guide?

The target audience for this guide are those who install or maintain ECM SM environments.

Every effort has been made to provide you with complete installation instructions. If information becomes available after the creation of the installation media from which you accessed this guide, we will provide an updated version of the guide on the IBM/FileNet Customer Service and Support web site (http://www.ibm.com/support). As a general rule, you should refer to the IBM web site to obtain the current version of this guide.

This guide provides instructions for installing and/or upgrading IBM Enterprise Content Management System Monitor, and identifies the IBM/FileNet and 3rd Party products that are certified for the current release. Be aware that each release of IBM Enterprise Content Management System Monitor may have multiple Interim Fixes, or Fix Packs available for installation, each with potentially different dependencies and installation requirements. Therefore, before you attempt to install or upgrade IBM Enterprise Content Management System Monitor, review the list of releases and their associated dependencies on the IBM Support web site (http://www.ibm.com/support).

Before you start

Users of the guide should have knowledge about Unix and/or Microsoft Windows® operating system, web servers, database systems and middleware platforms. The configuration of managed systems (clients) requires advanced knowledge of all IBM ECM systems that should be monitored.

If you lack the requisite skill sets it is strongly recommended to have IBM Lab Services or a certified ValueNet Partner in order to install this product.

Where you find this guide

You can find this documentation on the ECM SM installation media in the following folder:

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

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

Feedback on documentation

Send your comments by e-mail to comments@us.ibm.com. Be sure to include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

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 ECM SM 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.

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.

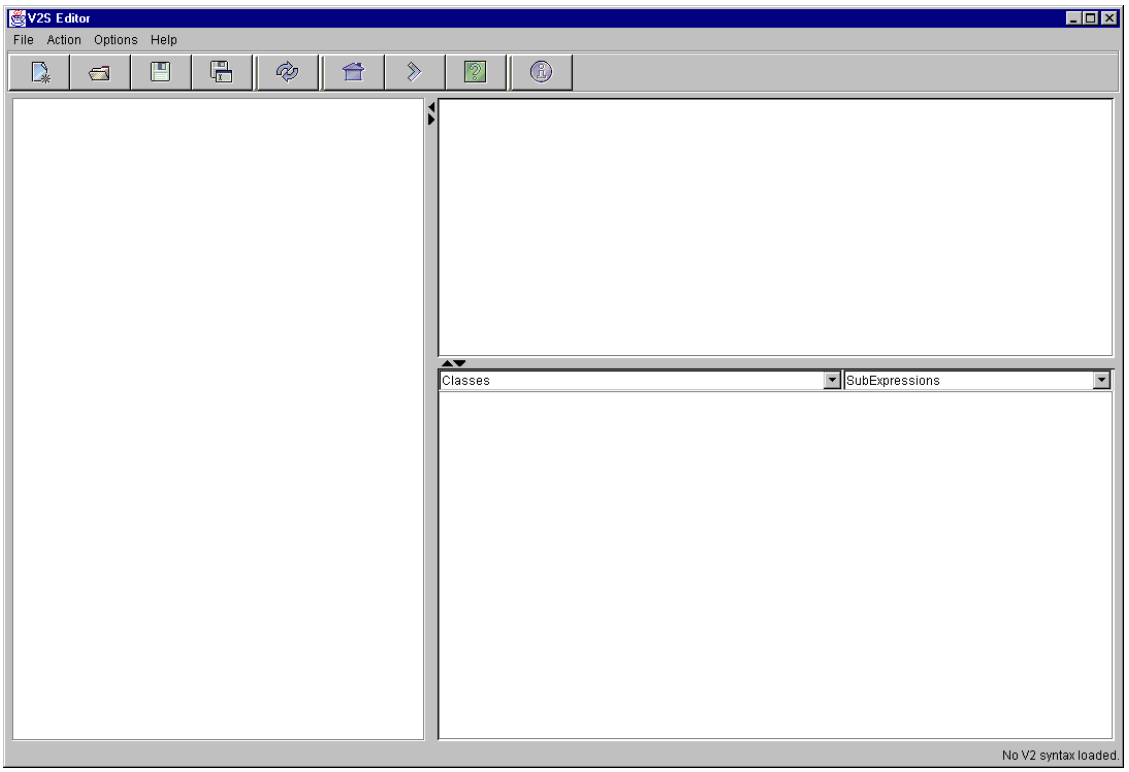
Installation based on the Web frontend

Since version 4.5.0 the V2S Editor is a Java Webstart application that can be downloaded and automatically installed on the desktop by selecting the 'V2S Editor' link on the 'Client Administration' console of the web Console.

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.

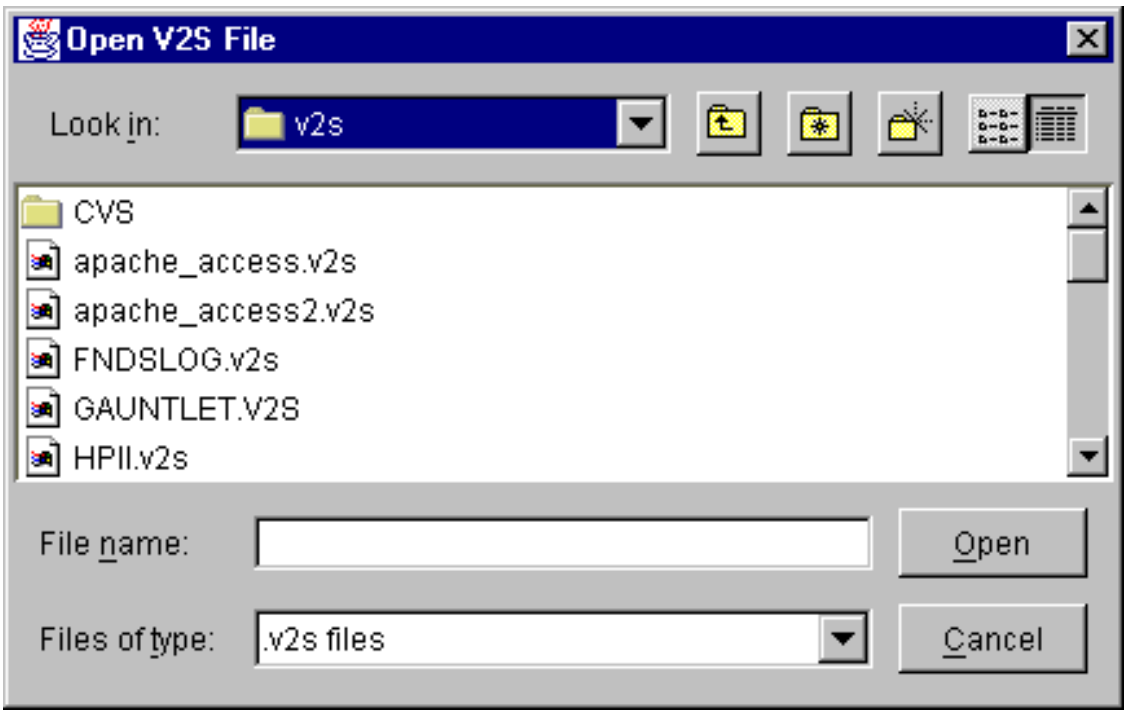
The v2s area is a editable text area which holds the v2 specification to be edited. It can be filled with an existing v2s file by choosing the item *Open V2S* from the **File** menu. The v2s area also contains two list boxes for direct access to class and subexpression definitions.

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

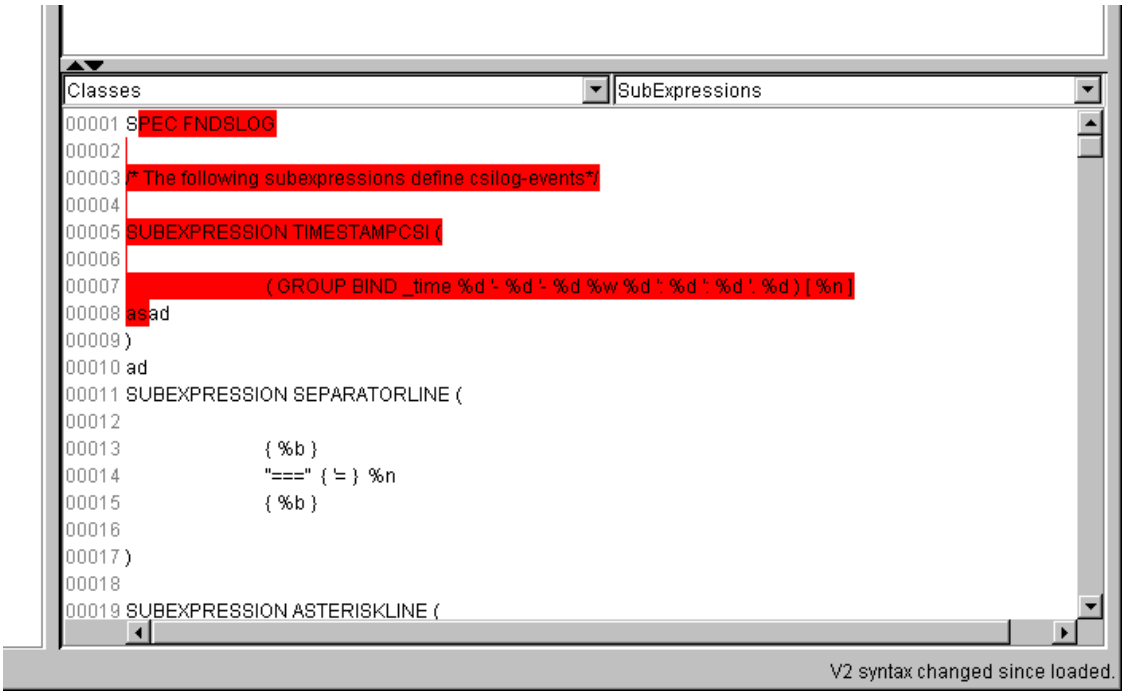
To load a v2s file, choose *Open V2S* from the **File** menu or press the corresponding toolbar button and a file dialog appears.



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, a error dialog is displayed and the region where the error is suspected is highlighted.



The error message when loading an invalid v2s file

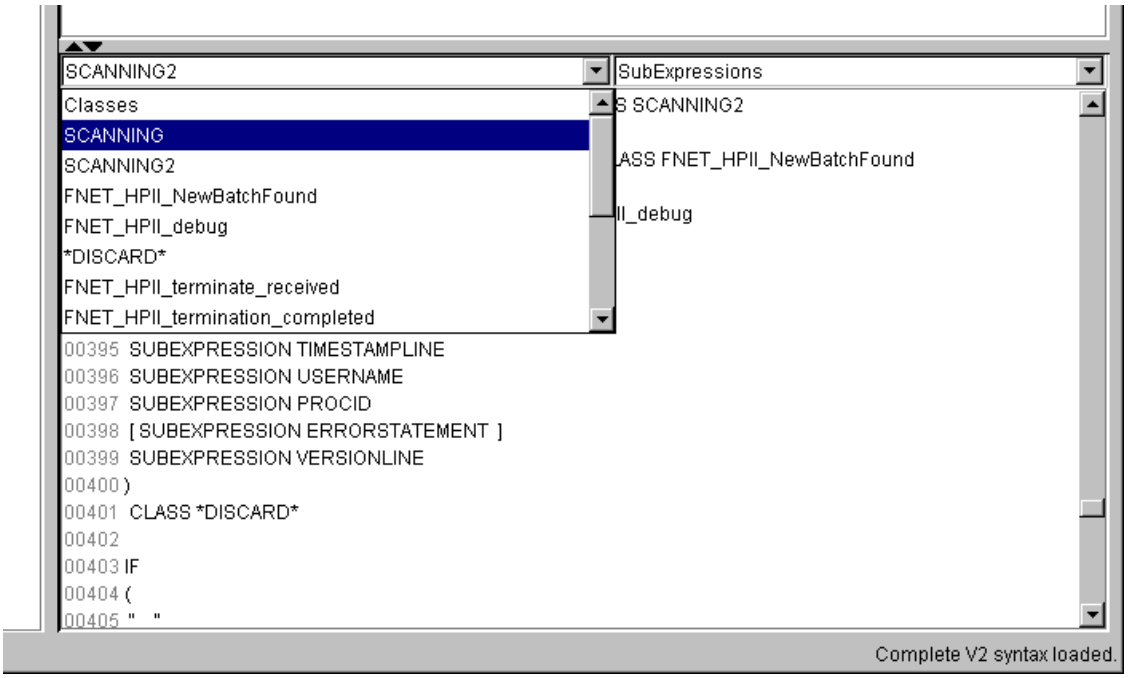
After the error has been corrected, the new specification has to be loaded into the v2s parser by choosing *Reload V2 specification* from the **Action** menu or by pressing the **reload** button from the toolbar.

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 .



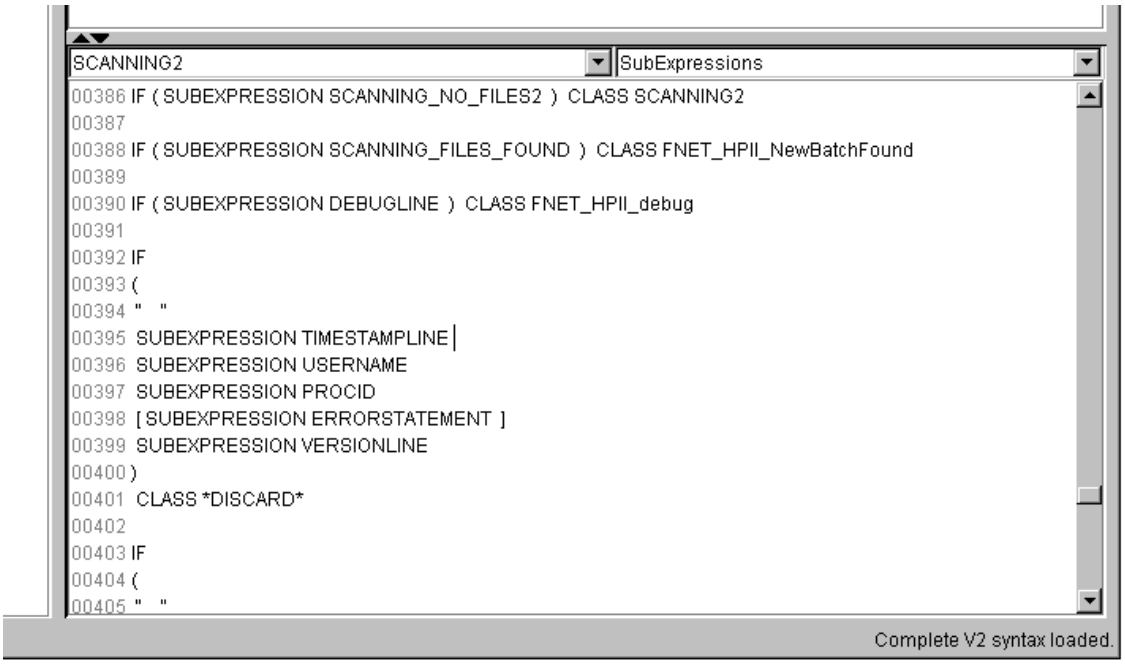
Jumping to a class definition by selecting it's name from the listbox

Editing expressions properties

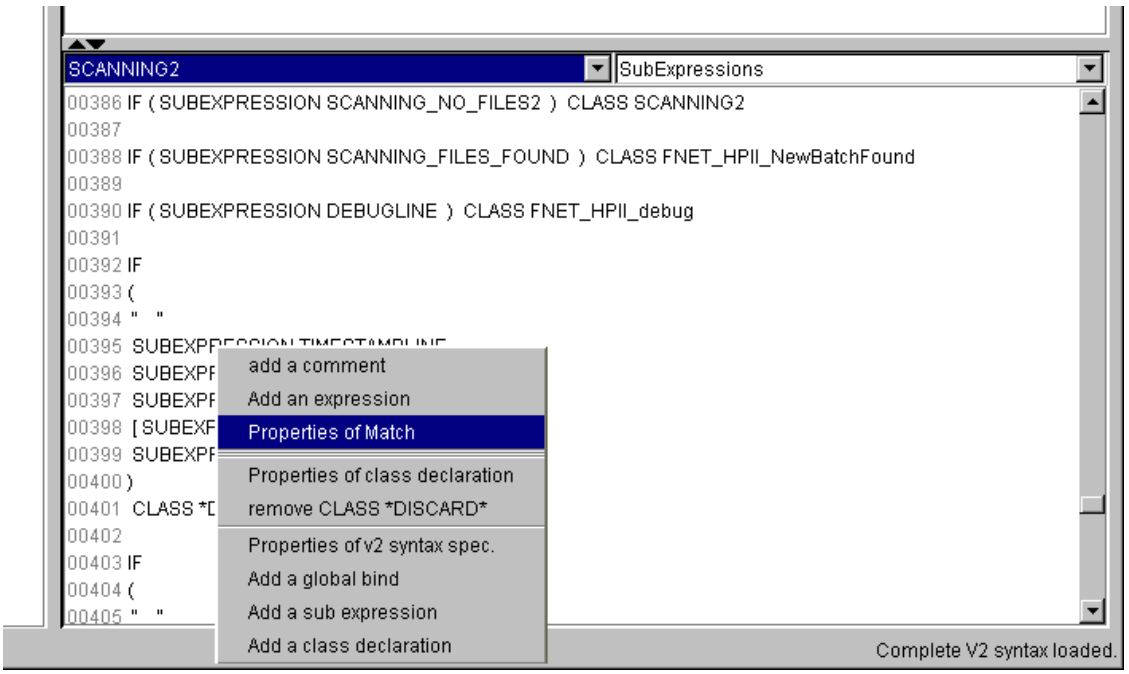
The properties of any expression can be edited by selecting the *Properties* item in the popup-menu. The popup-menu appears when the mouse is placed over an expression and the right mouse button is pressed.

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.

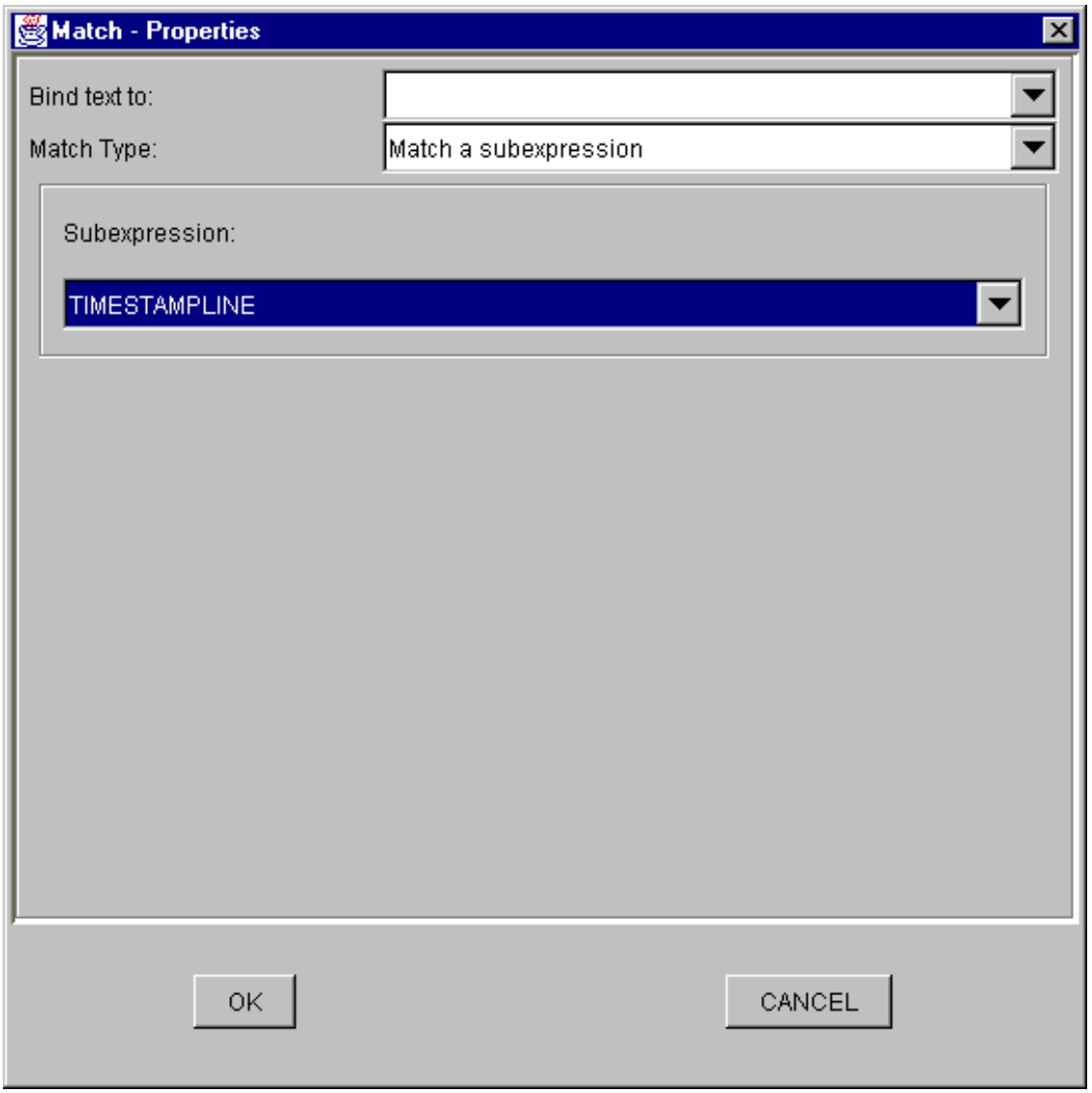
The screenshots below shows an example property change: the match for the subexpression *TIMESTAMPLINE* is changed into a match for the subexpression *VERSIONLINE*.



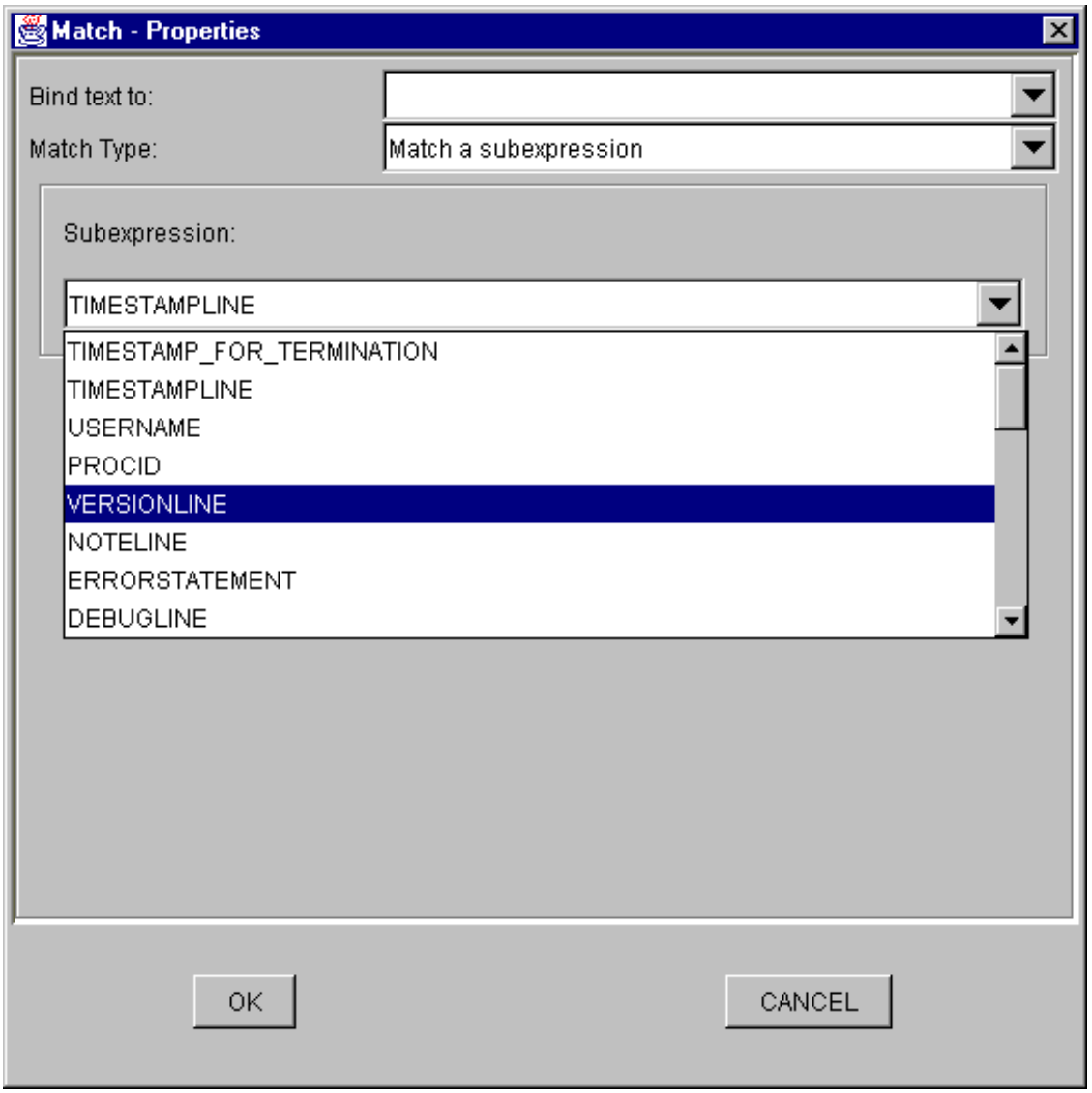
Editing expression properties: The sub expression *DISCARD* before the change



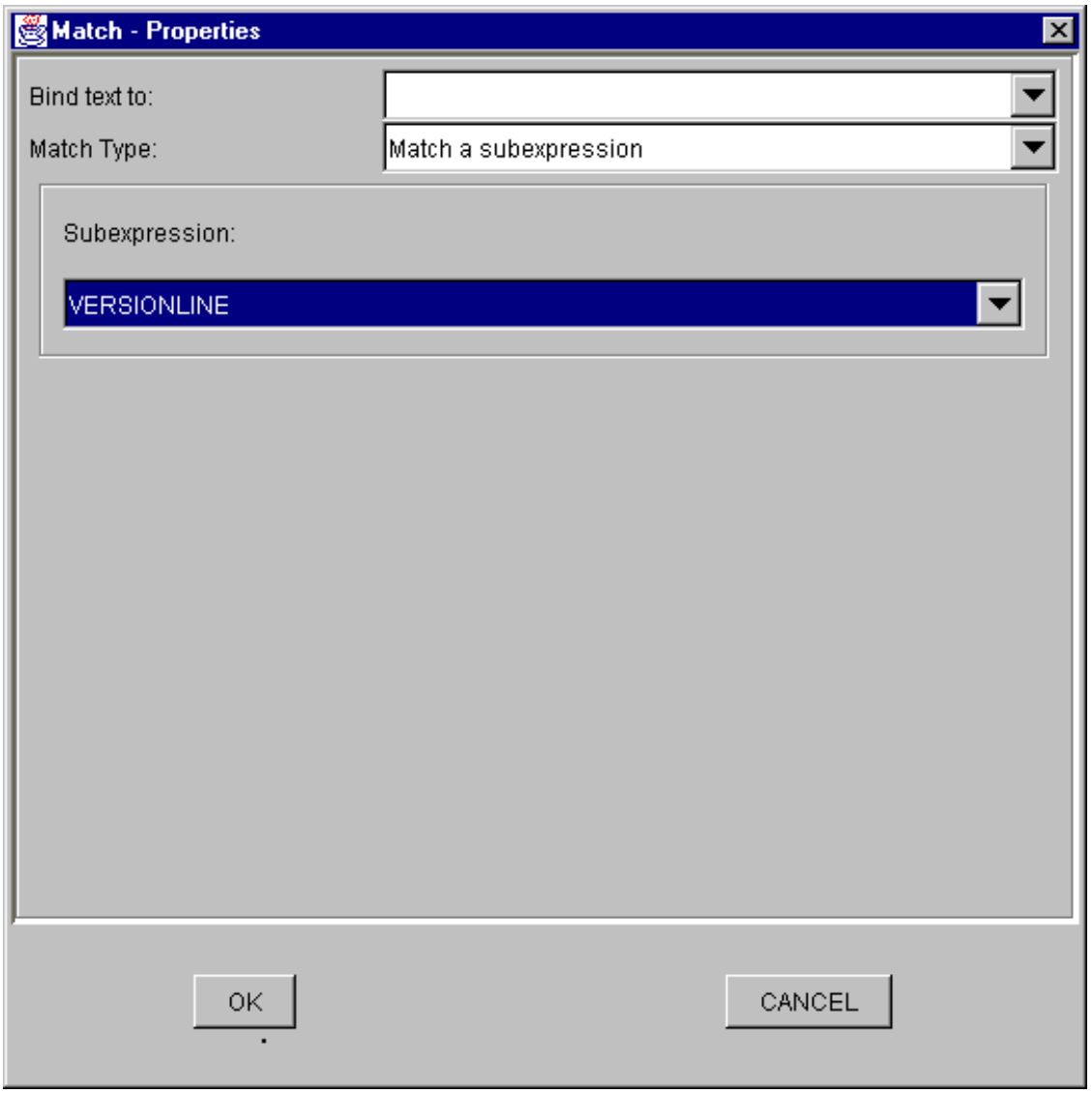
Editing expression properties: Opening the properties window.



Editing expression properties: The properties window before the change.

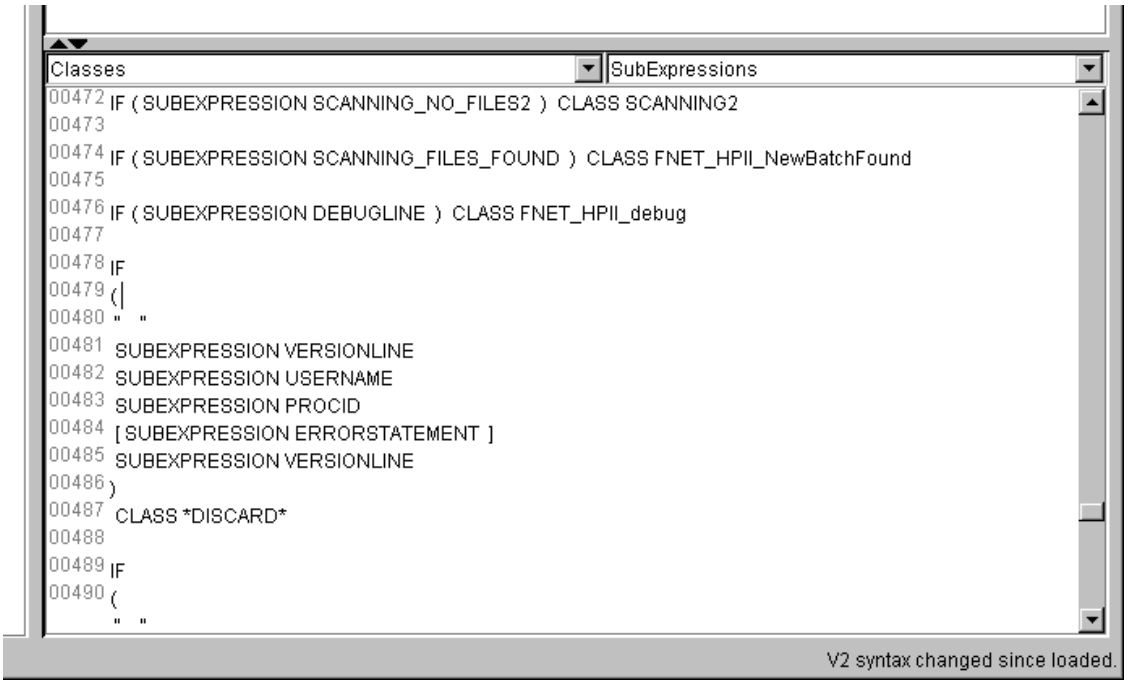


Editing expression properties: Changing TIMESTAMPLINE to VERSIONLINE



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

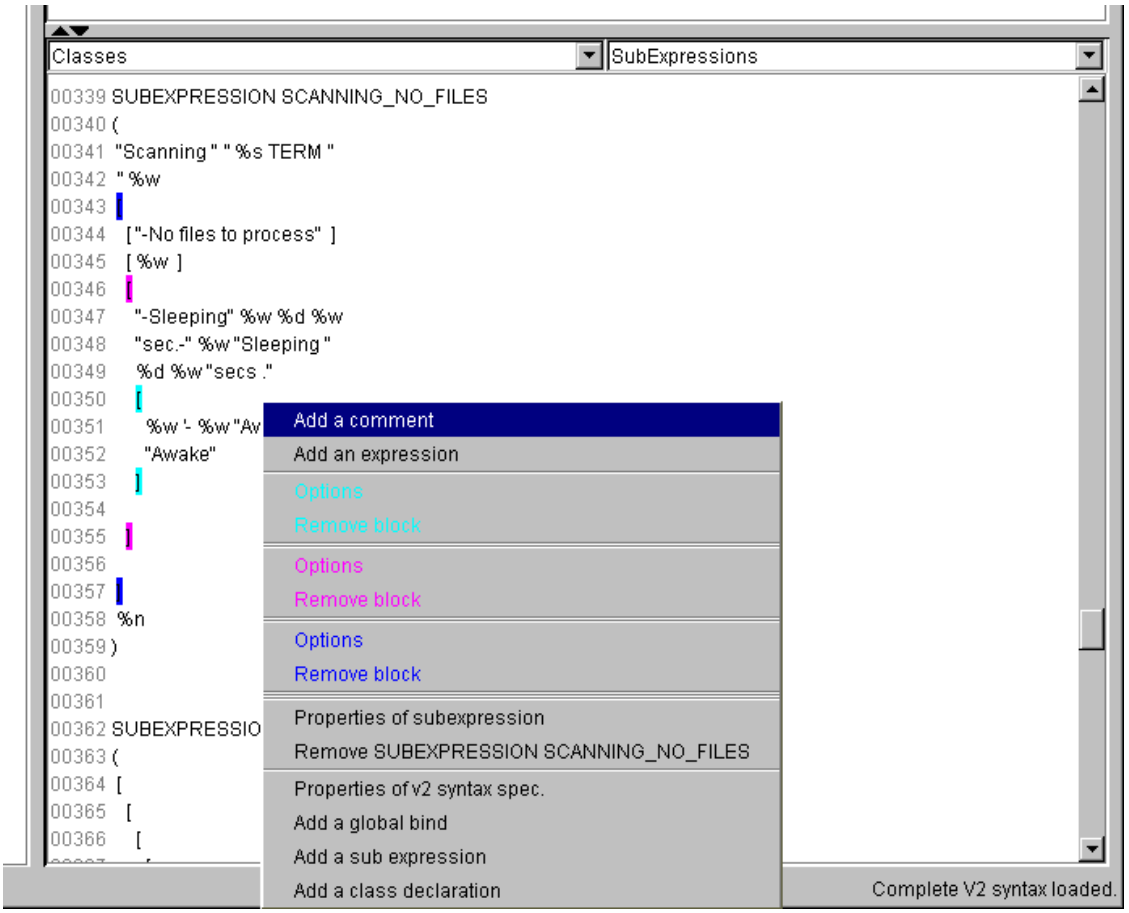
After pressing **OK**, the modifications are applied to the v2 specification.



Editing expression properties: The modified class deinition.

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.

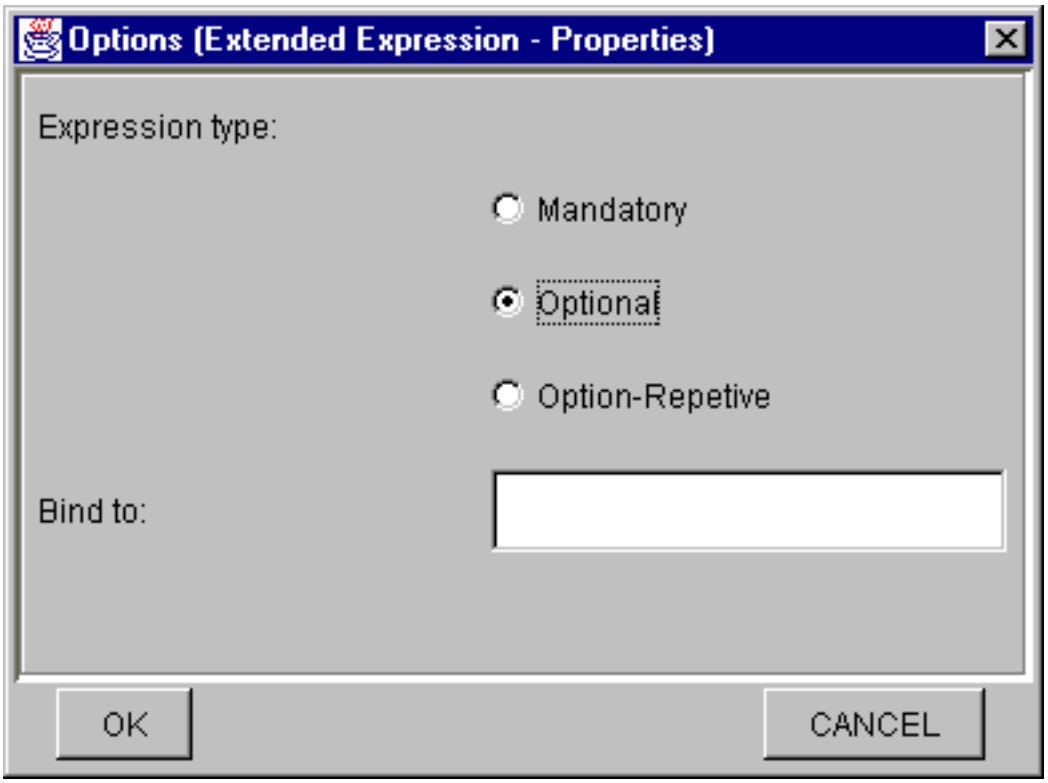


The v2s area popup window

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.



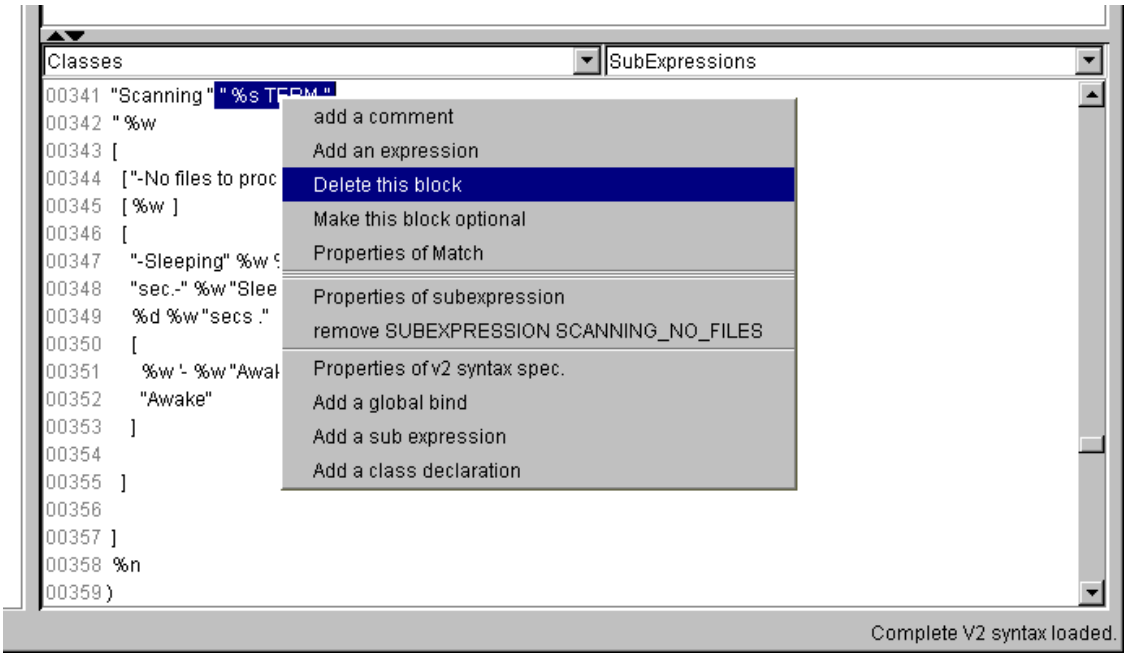
The block options dialog

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 choosen properties.

Deleting expressions

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



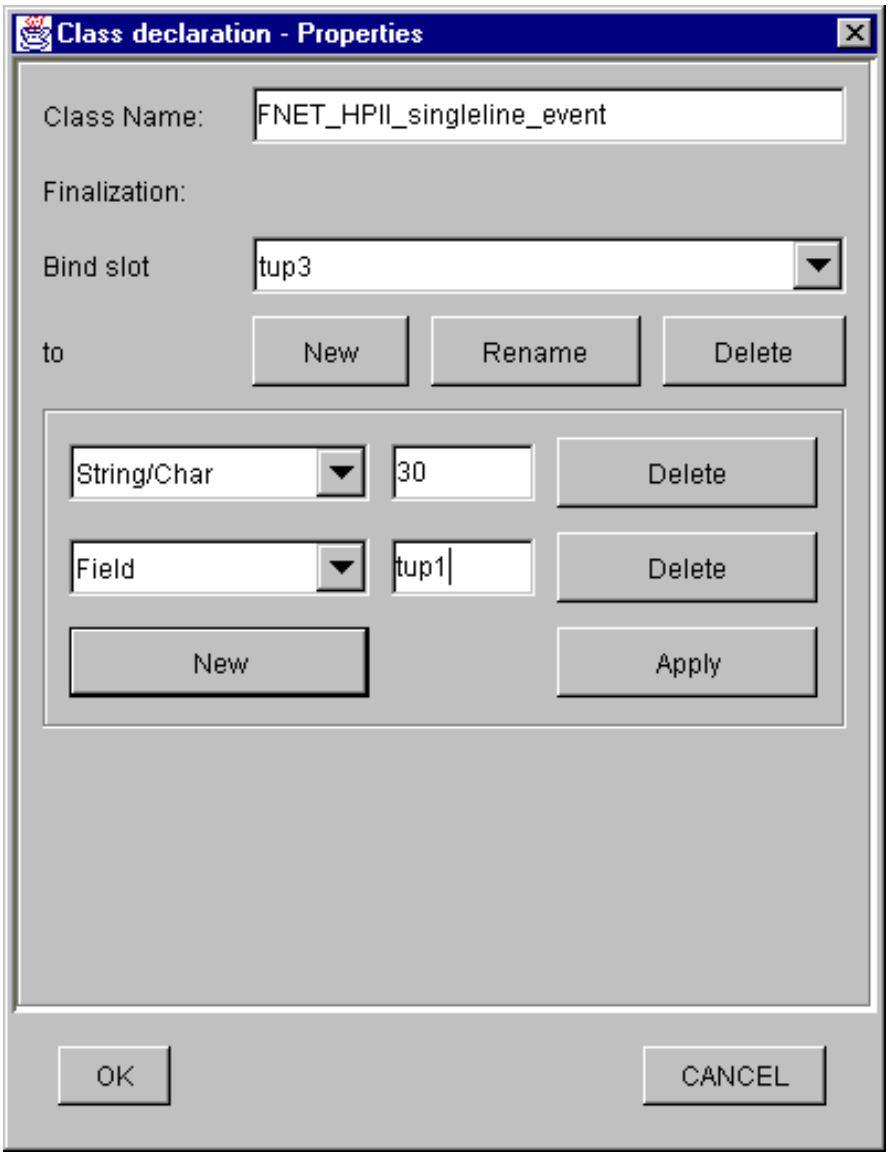
Deleting an expression block

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.

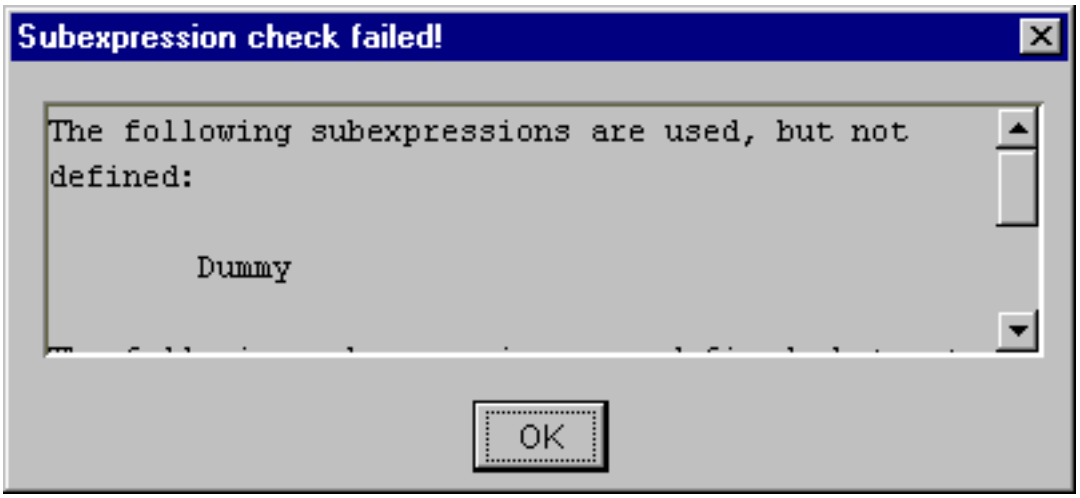
Checking the V2S file for correctness

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.

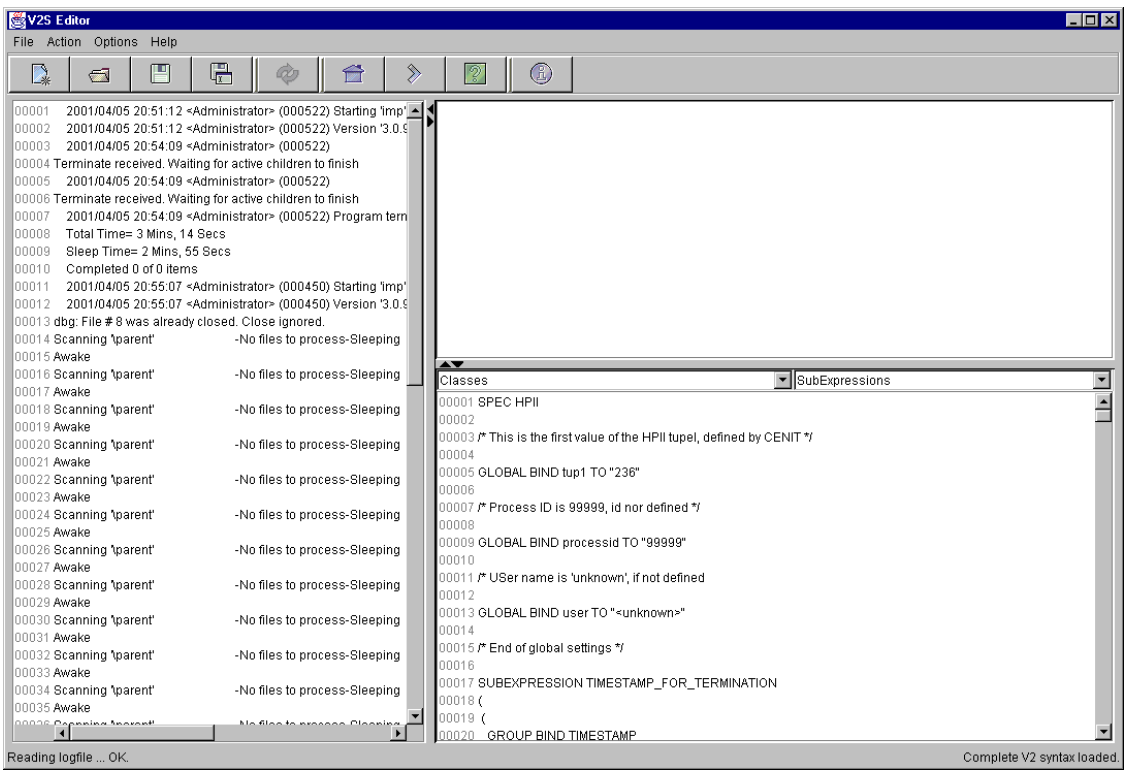
Each new v2 specification has one standard class for Unknown events. In most cases you should leave this class in the specification (as the last class declaration) to match events which are not matched by any other event class.

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

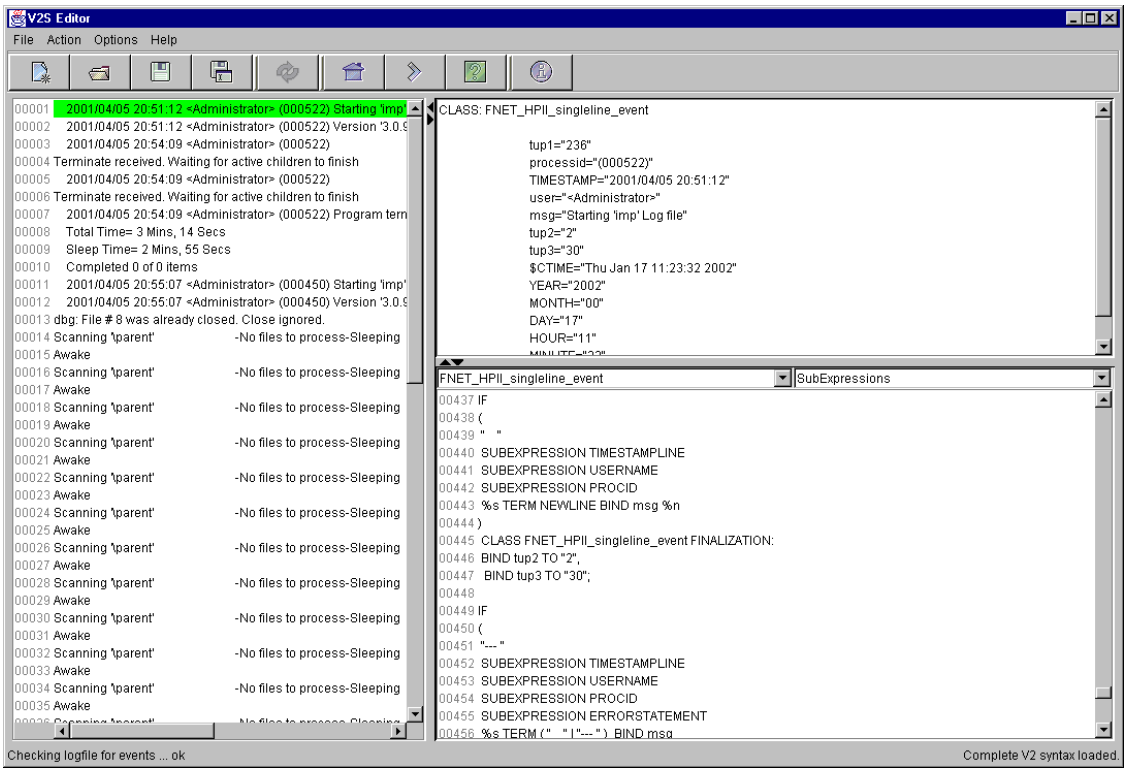
Parsing the logfile with the v2 specification

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 choosen, 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.

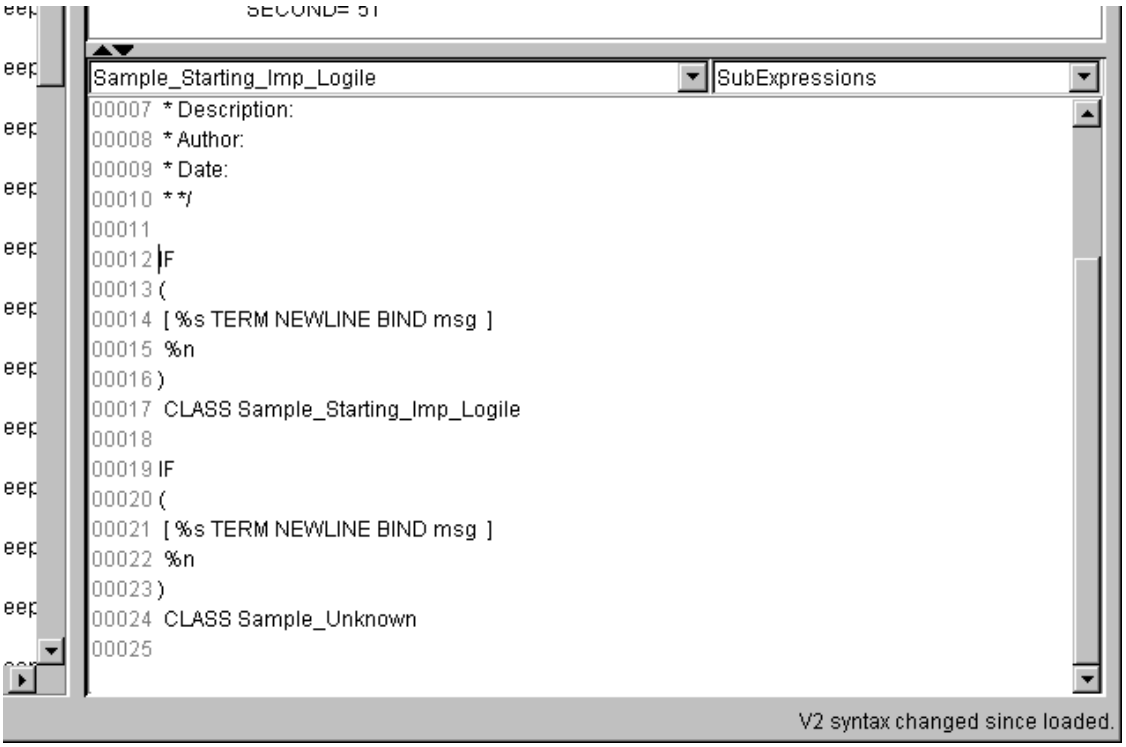
After selecting this menu item, a dialog box appears, where the classes to load can be selected. To load the full specification back into the parser, the v2 specification has to be reloaded (menu item *Reload V2S*).

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.

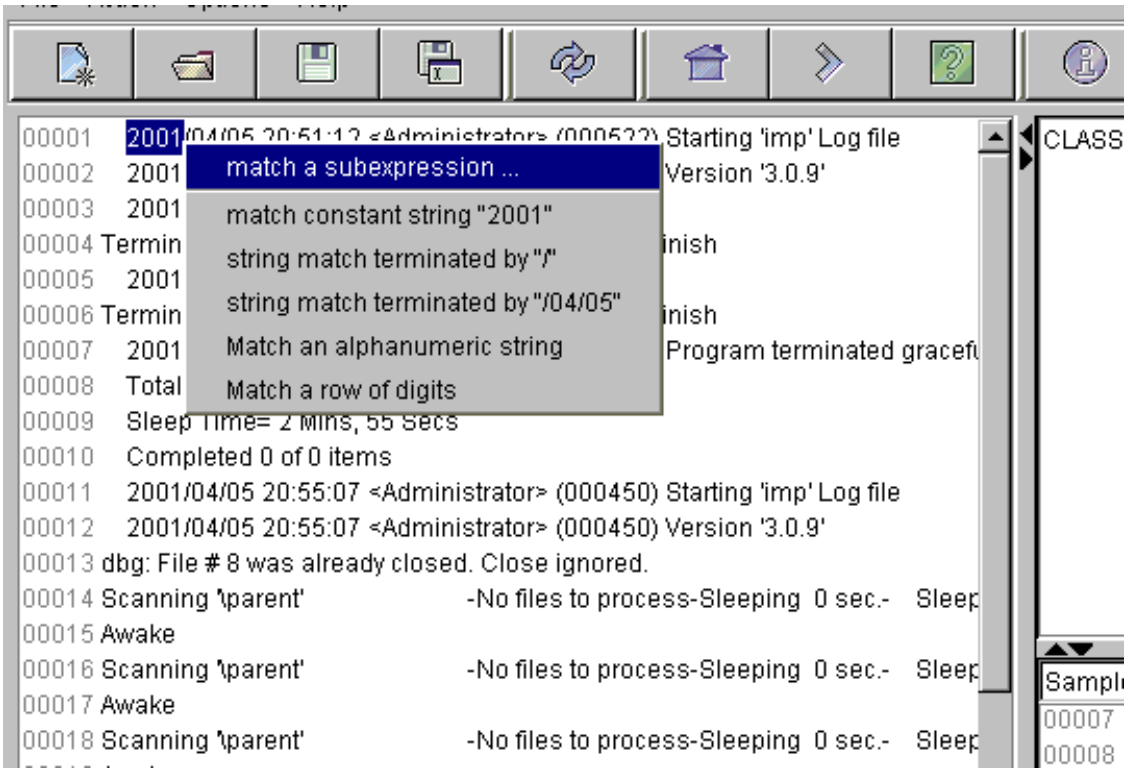
(See class `Sample_Starting_Imp_Logfile` in the screenshot)



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.

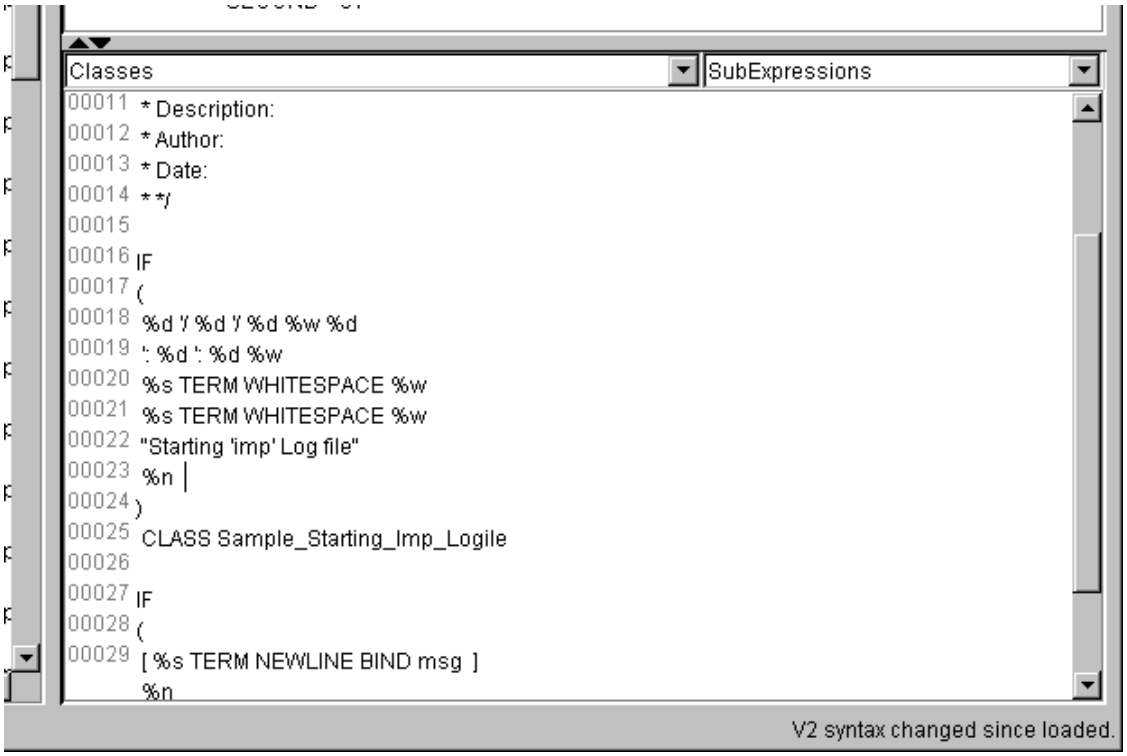


Creating an event description from the log file - step 2

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

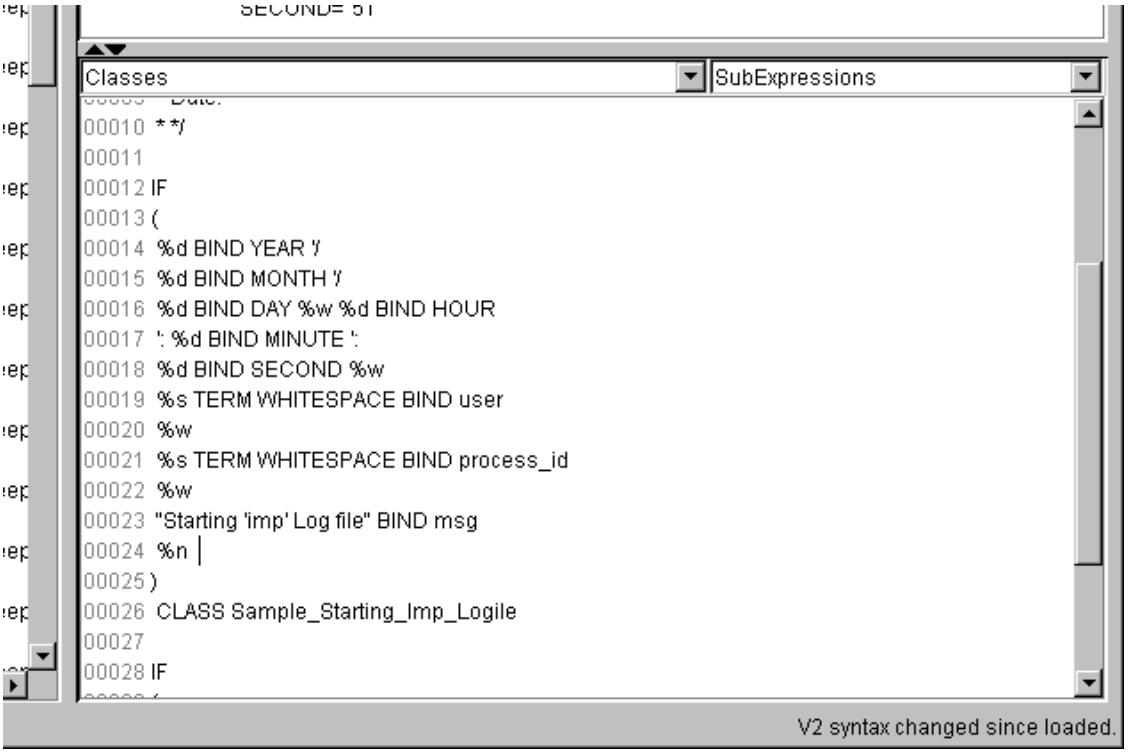
There is one special item in this popup menu the item *Match a subexpression* checks the marked text against all defined subexpressions and gives you a list of the ones which match. After selecting the a subexpression, a match for it is inserted into the v2s file.

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

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.

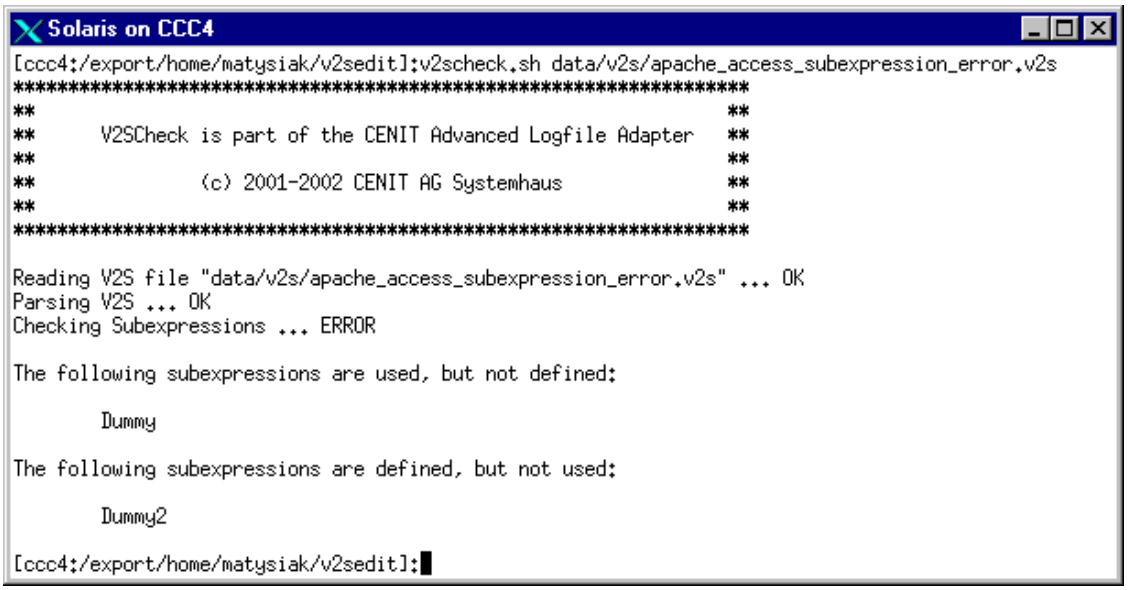
On all systems a java runtime environment >= JRE 1.3 must be reachable over the `PATH` environment variable to execute the programs.

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 failes, detailed error messages are printed on the screen.

Usage

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



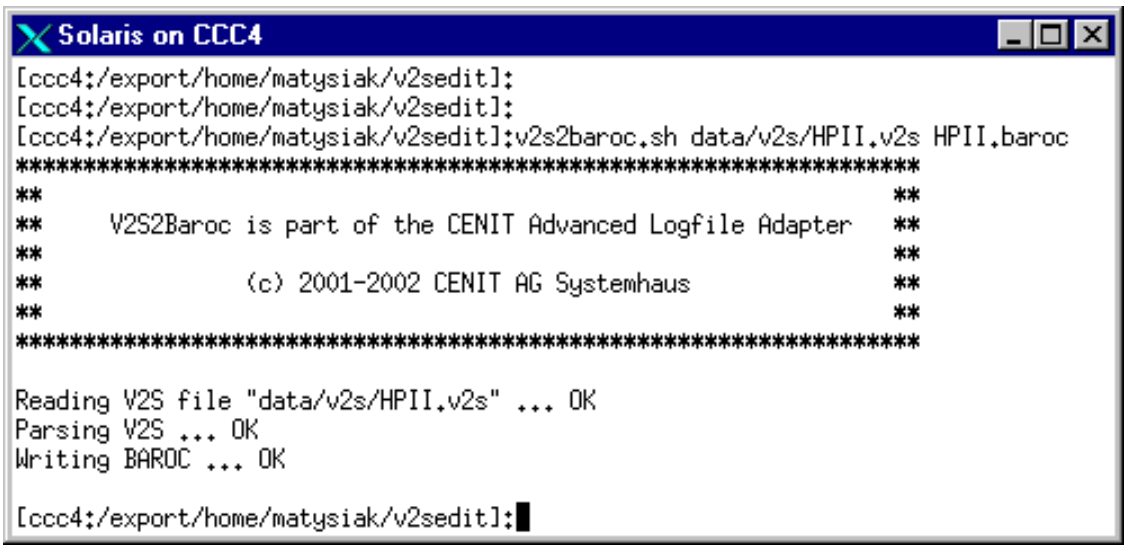
Using the v2scheck program

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.



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++.

```
0001 /* <comment> */
0002 // <comment terminated by new line>
```

Example of comments in v2s

Header

The header has the format:

```
0001 SPEC <name>
```

Figure: Format of v2s spec expression

`<name>` is any identifier for the format specification.

```
0001 SPEC SNA
```

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.

```
0001 GLOBAL BIND <slot name> TO "<string>"
```

Figure: Format of v2s global bind expression

```
0001 GLOBAL BIND source to cala
```

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
$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.

```
0001 IF expression CLASS name [ FINALIZATION: BIND <slot> TO <any sequence ↵
     of V2S expressions > ]
```

Figure: Format of v2s class expression

```
0001 IF (
0002 SUBEXPRESSION TIMESTAMP
0003 "myprocess shut down"
0004 ) CLASS MYPROCSHUTDOWN
```

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.

```
0001 SUBEXPRESSION name ( expression )
```

Figure: Format of v2s subexpression definition

```
0001 SUBEXPRESSION IPADDREXPR ( GROUP BIND IPADDR %d { '. %d } )
```

Example of a v2s subexpression definition

The "Macro" is called using

```
0001 SUBEXPRESSION name
```

Figure: Format of v2s subexpression call

```
0001 "Text" SUBEXPRESSION IPADDR "Text"
```

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.

```
0001 \65
```

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).

```
0001 %a BIND FIELD1
```

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

```
0001 "any text"
```

Figure: 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:

```
0001 (" alt string1" |  "alt string2" |  "alt string 3" )
```

Figure: 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

```
0001 SUBEXPRESSION name
```

Figure: 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.

```
0001 ( 'A %d )
```

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.

```
0001 'A %d [ '. %d ]
```

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.

```
0001 'A %d { '. %d }
```

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.

```
0001 ( GROUP BIND IPADDR %d { '. %d } )
```

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;

```

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 *TIMES-TAMPLINE* 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

```
0001 // Filename: V2SEdit_personal.properties
0002 //
0003 // Date: 12.12.2001
0004 //
0005 // Personal settings for V2S editor
0006 //
0007 // The settings in this file are user definable.
0008 // Things which are not configured here (like button and dialog texts)
0009 // are defined in V2SEdit.properties file or any of it's parents.
0010 // The V2SEdit.properties file also contains a default value for each
0011 // of  the following parameters, which is used if the parameter is
0012 // not configured here.
0013 //
0014 // The settings in this file are given like this:
0015 //
0016 //  [property name]=[value]
0017 //
0018 // For properties expecting color values, the following colors are
0019 // supported:
0020 //
0021 // black
0022 // blue
0023 // cyan
0024 // darkGray
0025 // gray
0026 // green
0027 // lightGray
0028 // magenta
0029 // orange
0030 // pink
0031 // red
0032 // white
0033 // yellow
0034 //
0035 // All lines starting with // are comments.
0036 // Look and Feel: The user interface manager
0037 //
0038 // 0 = use system default (motif on Unix, windows on Windows)
0039 // 1 = motif
0040 // 2 = windows (only in MS operating systems)
0041 // 3 = metal
0042 //
0043 // default: 0
0044 v2sedit.menu.options.lookandfeel.defaultmanager=3
0045 // Line number mode at startup
0046 //
0047 // The line no. mode can be switched on for each
0048 // of the text areas in the main window.
0049 // Setting the property to 0 turns the line no. mode off,
0050 // each other value turns it on.
0051 //
0052 // Line no. mode is switched on for logfile and v2s by default.
0053 //
0054 v2sedit.menu.options.linenumbers.logfile.startup=1
0055 v2sedit.menu.options.linenumbers.v2s.startup=1
0056 v2sedit.menu.options.linenumbers.result.startup=0
0057 // Text font and size for the logfile area in the main window.
0058 //
0059 // Default: SansSerif, 12
0060 //
0061 v2sedit.logarea.font=SansSerif
```

```
0062 v2sedit.logarea.fontsize=12
0063 // Highlight color for found events.
0064 v2sedit.logarea.eventcolor=green
0065 // Text font and size for the v2s area in the main window.
0066 //
0067 // Default: SansSerif, 12
0068 //
0069 v2sedit.v2sarea.font=SansSerif
0070 v2sedit.v2sarea.fontsize=12
0071 // Highlight color for errors in v2 syntax
0072 v2sedit.v2sarea.errorcolor=red
0073 // Show subexpression as tooltip? (0=no, 1=yes)
0074 v2sedit.v2sarea.showtooltips=1
0075 // Text font and size for the result area.
0076 // (This is the area where the FIRs are displayed.)
0077 //
0078 // Default: SansSerif, 12
0079 //
0080 v2sedit.resultarea.font=SansSerif
0081 v2sedit.resultarea.fontsize=12
0082 // Logfile path and extensions
0083 v2sedit.openlogfile.filechooser.path=data/logs
0084 v2sedit.openlogfile.filechooser.extension=.log
0085 // V2S file path and extensions
0086 v2sedit.openv2sfile.filechooser.extension=.v2s
0087 v2sedit.openv2sfile.filechooser.path=data/v2s
0088 // Baroc file path and extensions
0089 v2sedit.exportbaroc.filechooser.extension=.baroc
0090 v2sedit.exportbaroc.filechooser.path=data/baroc
0091 // baroc export: header and footer
0092 //
0093 // This text is added at the beginning (header) or end (footer) of
0094 // the BAROC file. {0} is the name of the v2 syntax.
0095 //
0096 v2sspec.barocexport.header=//\n// Filename:\n//\n//\t {0}.baroc\n//\n// ↵
     Language:\n//\n//\t Tivoli BAROC\n//\n// Description:\n//\n//\t Created by the ↵
     V2 format editor.\n//\n//\n\n
0097 v2sspec.barocexport.footer=
0098 // Display an "are you sure" dailog before exit? (0=no, 1=yes)
0099 v2sedit.exit.displaydialog=0
0100 // split lines in v2s windows at this length
0101 v2sspec.splitLinesAtLenght=30
// Creating a new V2S ->0102  default names
0103 v2sedit.specdecl.defaultname=Unknown
0104 // Comment at the beginning of the syntax
0105 // {0} is the syntax name
0106 v2sedit.specdecl.defaultheadcomment=V2 definition {0}\n *\n * (c) 2002 CENIT AG ↵
     Systemhaus, Stuttgart (Germany)\n *\n * Description:\n * Author:\n * Date:\n *
0107 // Default class name
0108 //{0} is the syntax name
0109 v2sedit.specdecl.addclass.defaultclass={0}_Unknown
0110 // Default subexpression name
0111 //{0} is the syntax name
0112 v2sedit.specdecl.addsubexpression.defaultepxression={0}_UNKOWN
0113 // remove redundant mandatory expressions?
0114 //
0115 // If set to 1, the expression
0116 //
0117 // IF
0118 // (
0119 //   ( %n  )
0120 //    %s TERM NEWLINE
0121 // )
0122 // CLASS mini_Unknown
```

```
0123 //
0124 // is replaced by
0125 //
0126 // IF
0127 // (
0128 //    %n
0129 //    %s TERM NEWLINE
0130 // )
0131 // CLASS mini_Unknown
0132 //
0133 // (the mandatory expression is no longer needed)
0134 v2sspec.extendedexpression.removeredundant=1
0135 // colors for colored popup menu in v2s area
0136 // 1 = inner
0137 // 10 = outer
0138 v2sspec.extendedexpression.color.1=blue
0139 v2sspec.extendedexpression.color.2=magenta
0140 v2sspec.extendedexpression.color.3=cyan
0141 v2sspec.extendedexpression.color.4=green
0142 v2sspec.extendedexpression.color.5=yellow
0143 v2sspec.extendedexpression.color.6=pink
0144 v2sspec.extendedexpression.color.7=red
0145 v2sspec.extendedexpression.color.8=orange
0146 v2sspec.extendedexpression.color.9=darkGray
0147 v2sspec.extendedexpression.color.10=gray
```

An example personal properties file

Appendix C. Copyright notice

IBM Enterprise Content Management System Monitor (December 2016)

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502

Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published.

Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Product Number: 5724-R91

Printed in USA

SC27-4910-04