

# ***EurekaLog 5.x.x user manual***

© 2001-2006 by Fabio Dell'Aria

# Table of Contents

|                                      |           |
|--------------------------------------|-----------|
| Foreword                             | 0         |
| <b>Part I Introduction</b>           | <b>2</b>  |
| 1 What is EurekaLog .....            | 2         |
| 2 Installation .....                 | 3         |
| 3 How to use EurekaLog .....         | 3         |
| 4 Features .....                     | 4         |
| <b>Part II New options</b>           | <b>8</b>  |
| 1 New menu option .....              | 8         |
| 2 Send Tab .....                     | 9         |
| 3 Log File Tab .....                 | 11        |
| 4 Exception Dialog Tab .....         | 12        |
| 5 Advanced Tab .....                 | 14        |
| 6 Messages Tab .....                 | 16        |
| 7 Exceptions Tab .....               | 17        |
| <b>Part III Types of Application</b> | <b>19</b> |
| 1 GUI .....                          | 19        |
| 2 Console .....                      | 23        |
| 3 Web .....                          | 25        |
| 4 Design-Time .....                  | 26        |
| <b>Part IV Events</b>                | <b>29</b> |
| 1 ExceptionNotify event .....        | 29        |
| How to use this event .....          | 29        |
| Examples .....                       | 30        |
| 2 ExceptionActionNotify event .....  | 30        |
| How to use this event .....          | 30        |
| Examples .....                       | 31        |
| 3 ExceptionErrorNotify event .....   | 32        |
| How to use this event .....          | 32        |
| Examples .....                       | 33        |
| 4 AttachedFilesRequest event .....   | 33        |
| How to use this event .....          | 33        |
| Examples .....                       | 34        |
| 5 CustomDataRequest event .....      | 34        |
| How to use this event .....          | 34        |
| Examples .....                       | 34        |
| 6 CustomFieldsRequest event .....    | 35        |
| How to use this event .....          | 35        |

|                                               |           |
|-----------------------------------------------|-----------|
| Examples .....                                | 35        |
| 7 PasswordRequest event .....                 | 36        |
| How to use this event .....                   | 36        |
| Examples .....                                | 36        |
| <b>Part V Generic functions</b>               | <b>39</b> |
| 1 StandardEurekaNotify function .....         | 39        |
| 2 IsEurekaLogInstalled function .....         | 39        |
| 3 CurrentEurekaLogOptions function .....      | 39        |
| 4 ForceApplicationTermination function .....  | 40        |
| 5 StandardEurekaError function .....          | 40        |
| 6 GetLastEurekaLogErrorCode function .....    | 41        |
| 7 GetLastEurekaLogErrorMsg function .....     | 41        |
| 8 SetEurekaLogInThread procedure .....        | 42        |
| 9 IsEurekaLogActiveInThread function .....    | 42        |
| 10 SetEurekaLogState procedure .....          | 42        |
| 11 IsEurekaLogActive function .....           | 43        |
| 12 EurekaLog version constants .....          | 43        |
| 13 SaveScreenshot procedure .....             | 43        |
| 14 SaveScreenshotToStream procedure .....     | 44        |
| 15 IsEurekaLogModule function .....           | 44        |
| 16 GetEurekaLogModuleVersion function .....   | 45        |
| 17 GetCompilationDate function .....          | 45        |
| 18 GenerateHTML function .....                | 46        |
| 19 SetCustomErrorMessage procedure .....      | 46        |
| 20 SetProxyServer procedure .....             | 46        |
| 21 SetProxyAuthenticationData procedure ..... | 47        |
| 22 GetLastExceptionAddress function .....     | 47        |
| 23 GetLastExceptionObject function .....      | 47        |
| 24 ShowLastExceptionData procedure .....      | 48        |
| 25 EurekaLogSendEmail function .....          | 49        |
| 26 CallStackToStrings procedure .....         | 49        |
| 27 GetCurrentCallStack function .....         | 50        |
| 28 GetLastExceptionCallStack function .....   | 50        |
| 29 GetCallStackByLevels function .....        | 50        |
| <b>Part VI TEurekaLog component</b>           | <b>53</b> |
| 1 How to use this component .....             | 53        |
| 2 Examples .....                              | 53        |
| <b>Part VII Types</b>                         | <b>57</b> |

|    |                                    |    |
|----|------------------------------------|----|
| 1  | TEurekaExceptionRecord type .....  | 57 |
| 2  | TEurekaModuleOptions type .....    | 57 |
| 3  | TEurekaStackList type .....        | 58 |
| 4  | TEurekaDebugInfo type .....        | 58 |
| 5  | TEurekaDebugDetail type .....      | 58 |
| 6  | TEurekaModuleInfo type .....       | 59 |
| 7  | TEurekaModulesList type .....      | 59 |
| 8  | TEurekaFunctionsList type .....    | 59 |
| 9  | TEurekaFunctionInfo type .....     | 59 |
| 10 | TEurekaModuleType type .....       | 60 |
| 11 | TEurekaExtraInformation type ..... | 60 |
| 12 | TEurekaAction type .....           | 60 |
| 13 | TForeground type .....             | 60 |
| 14 | TMessageType type .....            | 60 |
| 15 | TExceptionDialogOptions type ..... | 61 |
| 16 | TExceptionDialogOption type .....  | 61 |
| 17 | TCommonSendOptions type .....      | 61 |
| 18 | TCommonSendOption type .....       | 62 |
| 19 | TCallStackOptions type .....       | 62 |
| 20 | TCallStackOption type .....        | 62 |
| 21 | TBehaviourOptions type .....       | 62 |
| 22 | TBehaviourOption type .....        | 62 |
| 23 | TLogOptions type .....             | 62 |
| 24 | TLogOption type .....              | 62 |
| 25 | TShowOptions type .....            | 63 |
| 26 | TShowOption type .....             | 63 |
| 27 | TEmailSendMode type .....          | 63 |
| 28 | TWebSendMode .....                 | 63 |
| 29 | EFrozenApplication type .....      | 63 |
| 30 | TTerminateBtnOperation .....       | 64 |
| 31 | EEurekaLogGeneralError .....       | 64 |
| 32 | TEurekaLogErrorCodes type .....    | 64 |

## **Part VIII EurekaLog Viewer 66**

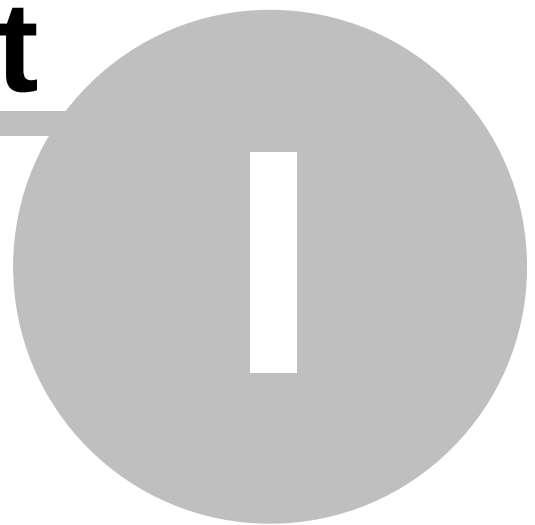
|   |                             |    |
|---|-----------------------------|----|
| 1 | How to use the Viewer ..... | 66 |
|---|-----------------------------|----|

## **Part IX Compatibility 68**

|   |                                          |    |
|---|------------------------------------------|----|
| 1 | Old 3.x version .....                    | 68 |
| 2 | Changed from the old 4.x version .....   | 68 |
| 3 | Changed from the old 4.5.x version ..... | 68 |

|                                           |           |
|-------------------------------------------|-----------|
| <b>Part X Enterprise version</b>          | <b>70</b> |
| 1 Debug .....                             | 70        |
| 2 Recompiling .....                       | 70        |
| <b>Part XI Command-line compiler</b>      | <b>72</b> |
| 1 To use the command-line compiler .....  | 72        |
| <b>Part XII Support</b>                   | <b>74</b> |
| 1 FAQ .....                               | 74        |
| 2 On-line support .....                   | 74        |
| <b>Part XIII Unsupported applications</b> | <b>76</b> |
| Index                                     | <b>77</b> |

**Part**



# 1 Introduction

## 1.1 What is EurekaLog

EurekaLog is the new add-in tool that gives your application (GUI, Console, Web, etc.) the ability to catch all exceptions, even those raised by memory leaks, and generate a detailed log of the call stack with unit, class, method and line-number information as shown in the image below. The information shown is also logged to a disk file and may optionally be forwarded to you by e-mail or via Web (HTTP/S - FTP).

EurekaLog helps you find infinite loops and deadlock bugs, raising an exception when the application is frozen for a specified time. The exception shows the same application state as it does for other exceptions.

EurekaLog is easy to use because it is fully integrated into the Delphi/CBuilder IDE. You just rebuild your application to add this new exception-logging capability.

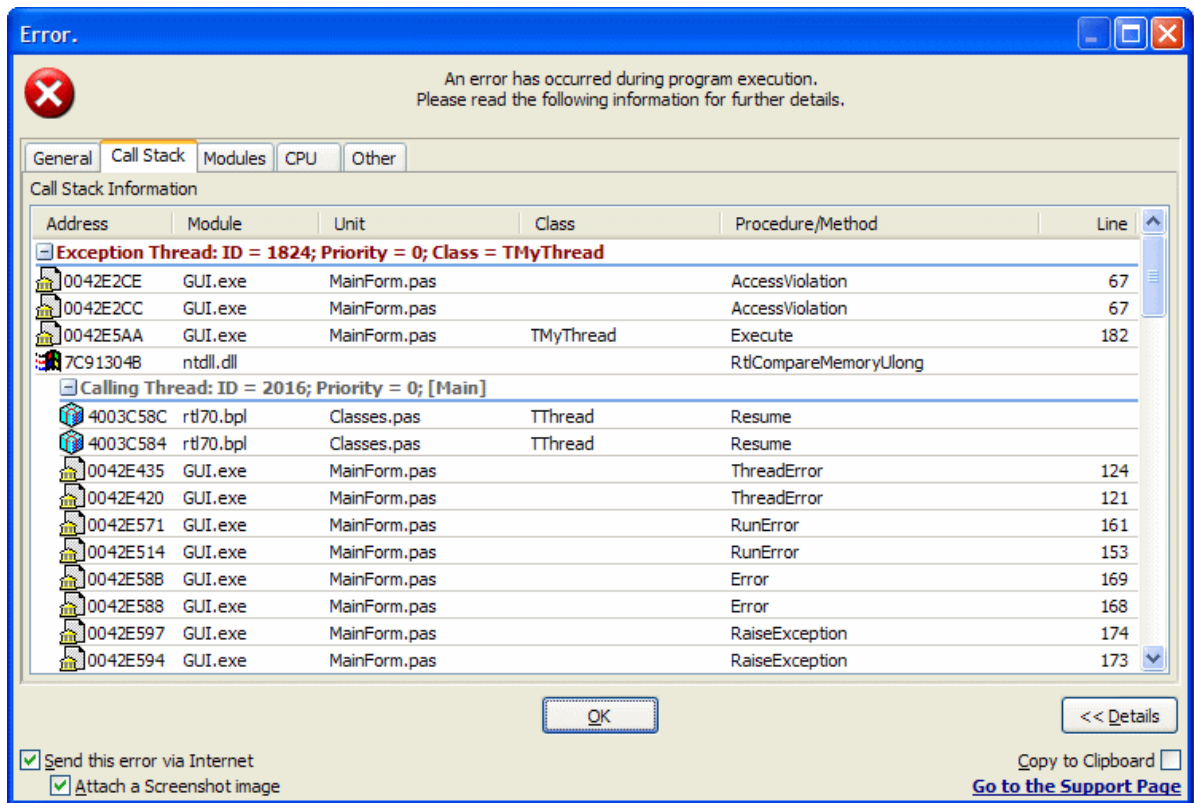
EurekaLog does not affect the performance of your application, as it only executes when an exception is raised. It increases the compiled file size by about 0.5% - 4% (this space is needed to store some additional, compressed and encoded, debugging information). To work EurekaLog needs only of the compiled file (not .map file).

EurekaLog is compatible with Delphi 3, 4, 5, 6, 7, 2005, 2006 and with CBuilder 5, 6, 2006 and works on all Windows platforms, from Win 95 to Win XP, Win 2003 server included.

**It comes with full source (only Enterprise version), full money back guarantee, is royalty free, and freely updatable!**

**Don't waste anymore time and money debugging your applications: now EurekaLog debugs them for you.**

See ["features"](#)  topic for further details.



A typical example of EurekaLog dialog.

## 1.2 Installation

Before installing EurekaLog, you must close all Delphi/CBuilder instances.

Run the EurekaLog installation program and select one or more Delphi/CBuilder versions you wish to install.

Start the installation.

When installation ends, open the Delphi/CBuilder IDE and you are ready to work with EurekaLog.

Note: EurekaLog install the [TEurekaLog](#) component into the EurekaLog component palette.

## 1.3 How to use EurekaLog

You must recompile your Delphi/CBuilder projects in order to use EurekaLog with them.

EurekaLog will automatically apply all necessary changes to make your projects intercept every exception.

To access to your project EurekaLog options, use the new "Project/EurekaLog Options..." menu item ([see here](#)).

To check if your project is compiled with EurekaLog you can check if the *EUREKALOG* directive exists.

Example:



```
{IFDEF EUREKALOG}
  WriteLn('Compiled with EurekaLog.');
```

```
{ELSE}
  WriteLn('Compiled without EurekaLog.');
```

```
{ENDIF}
```

To check if your project is compiled with the last EurekaLog 5.x version you can check if the `EUREKALOG_VER5` directive exists.

Example:

```
{IFDEF EUREKALOG}
  {IFDEF EUREKALOG_VER5}
    WriteLn('Compiled with EurekaLog 5.x.');
```

```
  {ELSE}
    WriteLn('Compiled with EurekaLog 4.x.');
```

```
  {ENDIF}
{ELSE}
  WriteLn('Compiled without EurekaLog.');
```

```
{ENDIF}
```

## 1.4 Features

This is a list of the main EurekaLog features:

| Compiled file                         | Pro   | Ent   |
|---------------------------------------|-------|-------|
| Kbytes added                          | ~150  | ~150  |
| Overall increase (% min - max)        | 0.5/4 | 0.5/4 |
| Decrease in Application's performance | none  | none  |
| Source code                           | Pro   | Ent   |
| Full included Source Code             | ✗     | ✓     |

| IDE                                                                                              | Pro | Ent |
|--------------------------------------------------------------------------------------------------|-----|-----|
| Full integration with Delphi/Cbuilder IDE                                                        | ✓   | ✓   |
| Display of custom messages for every kind of exception                                           | ✓   | ✓   |
| New Design-Time exception management ( <i>for debug packages at design-time</i> )                | ✓   | ✓   |
| Ability to save all modified files when an exception is trapped ( <i>ex: IDE crashes</i> )       | ✓   | ✓   |
| Integration with the Delphi/Cbuilder IDE help (in HTML format) <b>NEW</b>                        | ✓   | ✓   |
| New EurekaLog Viewer full integrated with the Delphi/CBuilder IDE <b>NEW</b>                     | ✓   | ✓   |
| New application capabilities                                                                     | Pro | Ent |
| Display of custom messages for every kind of exception                                           | ✓   | ✓   |
| Creation of a detailed Log for every exception ( <i>module, unit, class, method, line, ...</i> ) | ✓   | ✓   |
| Delivery of a customizable e-mail or Web message for every exception, with log-file              | ✓   | ✓   |
| Attach a PNG Screenshot to the message                                                           | ✓   | ✓   |
| Append text containing a user description on the bug reproducibility to the log                  | ✓   | ✓   |
| Fully customizable message texts ( <i>for multilanguage applications</i> )                       | ✓   | ✓   |
| Termination/restarting after a customizable number of exceptions                                 | ✓   | ✓   |

|                                                                                                      |   |   |
|------------------------------------------------------------------------------------------------------|---|---|
| Fully customizable exception management with new EurekaLog event                                     | ✓ | ✓ |
| Debug of third-party DLLs ( <i>showing address, module and procedure</i> )                           | ✓ | ✓ |
| Debug of third-party BPLs ( <i>showing address, module, unit, class and procedure</i> )              | ✓ | ✓ |
| Initialization/Finalization exceptions are trapped ( <i>only Delphi version</i> )                    | ✓ | ✓ |
| Creation of a detailed Log when the application freezes ( <i>as for any other exception</i> )        | ✓ | ✓ |
| Jump from Exception Dialog line to source code line ( <i>with a simple double-click</i> )            | ✓ | ✓ |
| Upload of log-file and attached files via HTTP/HTTPS and FTP protocol <b>NEW</b>                     | ✓ | ✓ |
| Send a log-file copy in XML format <b>NEW</b>                                                        | ✓ | ✓ |
| Send the last generated HTML page ( <i>only for the WEB applications</i> ) <b>NEW</b>                | ✓ | ✓ |
| Send the email/"upload files" in a separated thread <b>NEW</b>                                       | ✓ | ✓ |
| Compress all files to send in tar.gz format (.tgz) <b>NEW</b>                                        | ✓ | ✓ |
| Attach customizable files to the send message <b>NEW</b>                                             | ✓ | ✓ |
| Add customizables data to the log-file <b>NEW</b>                                                    | ✓ | ✓ |
| Add customizables fields to the uploading HTML page <b>NEW</b>                                       | ✓ | ✓ |
| New improved Exception-Dialog GUI interface <b>NEW</b>                                               | ✓ | ✓ |
| Full Exception-Dialog customization <b>NEW</b>                                                       | ✓ | ✓ |
| Adding customizable "Support Link" in the Exception-Dialog <b>NEW</b>                                | ✓ | ✓ |
| Customizable HTML error page via HTML template <b>NEW</b>                                            | ✓ | ✓ |
| Automatical Exception-Dialog closing after customizable time <b>NEW</b>                              | ✓ | ✓ |
| Customizable log-file section view ( <i>show/hide Modules/CPU sections</i> ) <b>NEW</b>              | ✓ | ✓ |
| Full Call-Stack customization <b>NEW</b>                                                             | ✓ | ✓ |
| Display detailed running Threads Call-Stack ( <i>calling Thread Call-Stack included</i> ) <b>NEW</b> | ✓ | ✓ |

| Type of supported applications                 | Pro | Ent |
|------------------------------------------------|-----|-----|
| MultiThread                                    | ✓   | ✓   |
| Indy threads/components                        | ✓   | ✓   |
| GUI ( <i>standard graphical applications</i> ) | ✓   | ✓   |
| Console                                        | ✓   | ✓   |
| BPL ( <i>Borland package library</i> )         | ✓   | ✓   |
| DLL ( <i>dynamic-link library</i> )            | ✓   | ✓   |
| NT Services                                    | ✓   | ✓   |
| Control Panel                                  | ✓   | ✓   |
| ActiveX-COM                                    | ✓   | ✓   |
| ISAPI                                          | ✓   | ✓   |
| CGI                                            | ✓   | ✓   |
| IntraWeb                                       | ✓   | ✓   |
| Applications compiled with Runtime Packages    | ✓   | ✓   |
| Other                                          | Pro | Ent |
| Delphi Command-line compiler                   | ✓   | ✓   |
| CBuilder Command-line compiler                 | ✓   | ✓   |
| FinalBuilder support                           | ✓   | ✓   |
| Exe compressors full compatibility             | ✓   | ✓   |

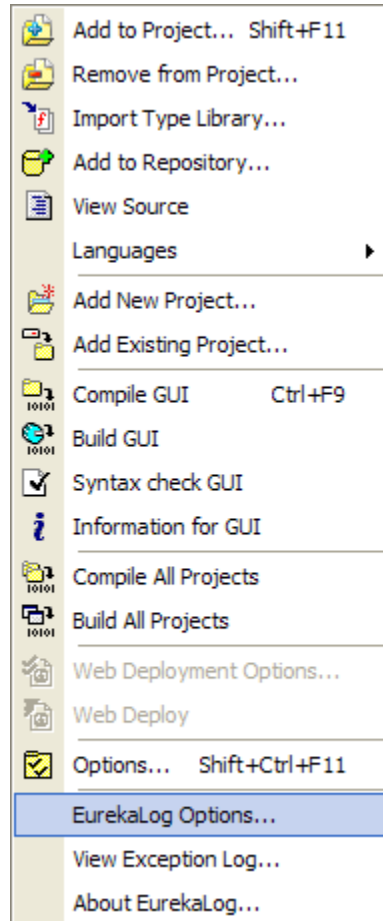
|                                                                        |   |   |
|------------------------------------------------------------------------|---|---|
| Exe protectors full compatibility                                      | ✓ | ✓ |
| New EurekaLog Viewer full integrated with the Windows Shell <b>NEW</b> | ✓ | ✓ |

**Part**



## 2 New options

### 2.1 New menu option



New option in IDE's main menu added by EurekaLog:  
click on it to access EurekaLog options.

## 2.2 Send Tab

Here you may tell the program to automatically send an email (multiple addresses are allowed) and a Web message whenever an exception is raised.

### **Email Send Options:**

You can choose to send the email via default email client, via SMTP client or via internal SMTP server.

If you select the SMTP client option then you must provide the relevant settings (email from address, host name/ip, host port and UserID/Password, the later only if the SMTP server requires log-in).

If you select the SMTP server option then you must provide the relevant settings (email from address).

For all Email options, you may request to "Append Logs in message text" adding the Log text at the bottom of the email message.

### **Web Send Options:**

Activating this options you may tell to the program to send the Log-File with all the other attached files to a Web location.

You can choose to send the message via [HTTP](#), [HTTPS](#) (secure HTTP) or [FTP](#) Internet protocol.

You must set a valid URL and optionally the User and Password fields.

To configure a custom Proxy setting see the relative [SetProxyServer](#)<sup>[46]</sup> and [SetProxyAuthenticationData](#)<sup>[47]</sup> procedures (by default EurekaLog use the System Internet Settings to get the Proxy configuration).

Follow a PHP Web server script capable to download all the HTTP/HTTPS uploaded files via the Web send feature.

#### PHP example:

```
<?
foreach ($_FILES as $key=>$value)
{
    $uploaded_file = $_FILES[$key]['tmp_name'];
    $server_dir = 'upload/'; // Upload folder
    $server_file = $server_dir.basename($_FILES[$key]['name']);

    // Move the uploaded file to the Server uploaded directory...
    if (move_uploaded_file($uploaded_file, $server_file))
    {
        // Here your code...
    }
}
?>
```

#### New Events:

To send to a Web HTML page a specified fields list, it's available the new [CustomFieldsRequest](#)<sup>[35]</sup> event.

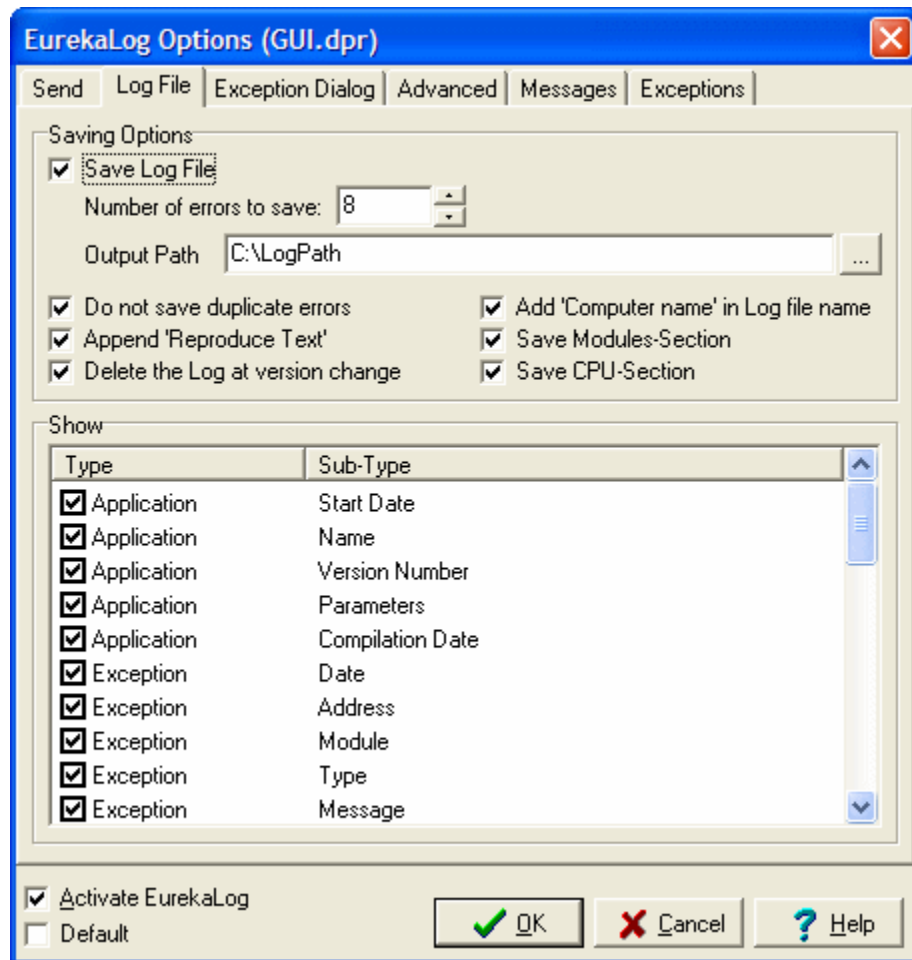
To add some custom data to the Log text it's available the new [CustomDataRequest](#)<sup>[34]</sup> event.

To send custom attached files with the message (Email and Web) it's available the new [AttachedFilesRequest](#)<sup>[33]</sup> event.

#### Common Send Options:

|                                 |                                                                                                       |
|---------------------------------|-------------------------------------------------------------------------------------------------------|
| "Show the send dialog"          | Show the send dialog                                                                                  |
| "Show success/failure msg."     | Show the succes or failure message after the send process                                             |
| "Send entire Log file"          | Send the entire log (by default only the log of the latest exception is sent)                         |
| "Send an XML Log's copy"        | Send an XML copy of the Log file                                                                      |
| "Send a screenshot"             | Attach a PNG screenshot to the message                                                                |
| "Use only active window"        | Catches a screenshot of active window instead that the entire desktop                                 |
| "Send last HTML page"           | Send the last generated HTML page (for Web application only)                                          |
| "Show in a separated thread"    | Run the send process in a separated thread (so the main process can continue without any other break) |
| "Add 'Date' in every file name" | Add the current date-time (in ISO format) to all attached file names                                  |
| "Compress all files to send"    | Send only one compressed file containing all attached file                                            |
| "Attached files"                | A list of files path to attach to the sending message                                                 |

## 2.3 Log File Tab



In this section you may customize the log file.

### **Saving Options:**

"Save Log File"

"Number of errors to save"

"Output Path"

"Do not save duplicate errors"

"Append 'Reproduce Text'"

"Delete the Log at version change"

"Add 'Computer name' in Log file..."

"Save Modules-Section"

Save the log file into the "Output path" folder

Max number of logged exceptions to save (min=1)

The full path of Log File (if is empty then the log is saved into the current program path - if the folder is not writable then the log is saved in the

"%UserProfile%\EurekaLog\ProjectName" folder)

Inhibit logging exceptions that have already been logged to the log file (*Inhibiting it's send too*)

Append to the Log File a text, entered by the user of your program, describing How to reproduce the error

Delete the Log file when the program version it's changed

Add the Computer-Name to the Log file name

Save the Modules section in the Log File (*displaying it in the Exception Dialog*)



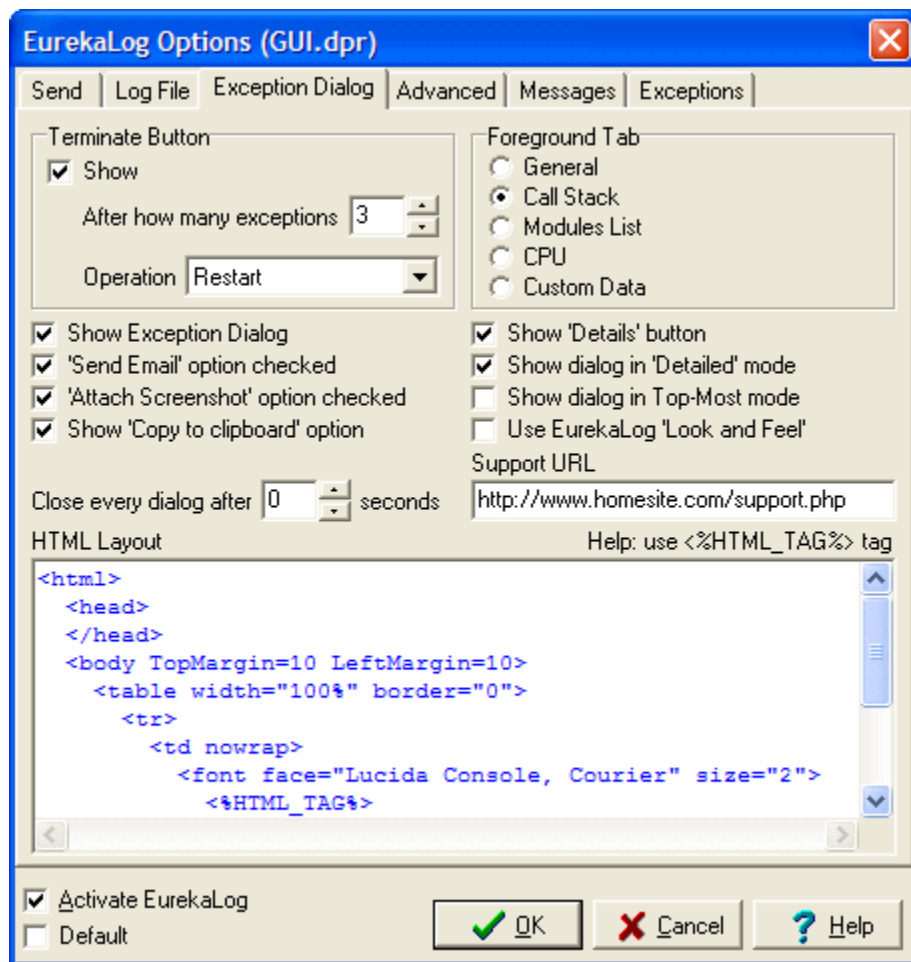
"Save CPU-Section"

Save the CPU section in the Log File (displaying it in the Exception Dialog)

### Show:

You may select the type (Application, Exception, etc.) and sub-type (Name, Date, Module, etc.) of the exceptions you wish to store in the logfile.

## 2.4 Exception Dialog Tab



In this section you may customize the Exception Dialog (or HTML error page for the Web application).

### Terminate button:

In this panel you can select to show a terminate or restart button after a specified number of exceptions.

### Foreground Tab:

In this panel you can indicate which Tab, General, Call-Stack, Modules List, CPU or Custom Data,

should be presented in the foreground when an exception is presented on screen.

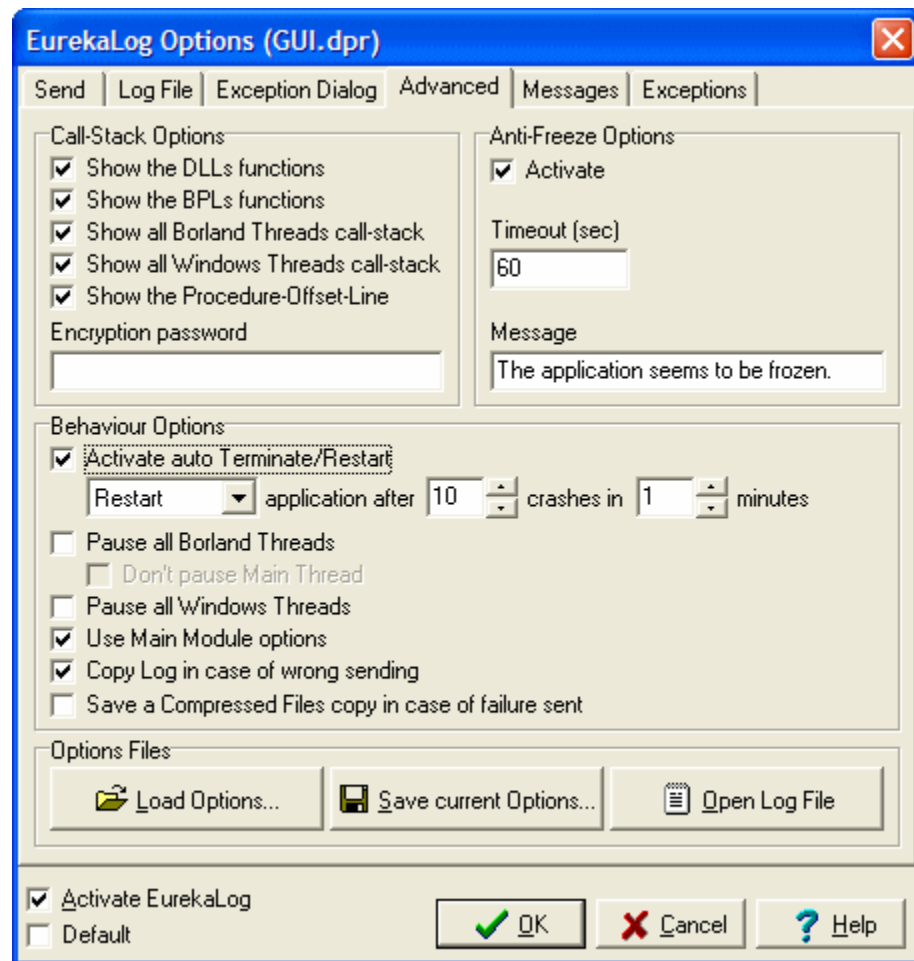
### **Common Options:**

|                                            |                                                                                                                           |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>"Show Exception Dialog"</i>             | Show the Exception Dialog                                                                                                 |
| <i>"Send Email' option checked"</i>        | Check the 'Send Email' option                                                                                             |
| <i>"Attach Screenshot' option checked"</i> | Check the 'Attach Screenshot' option                                                                                      |
| <i>"Show 'Copy to clipboard' option"</i>   | Show the 'Copy to clipboard' option                                                                                       |
| <i>"Show 'Details' button"</i>             | Show the 'Detail' button ( <i>used to display all the detailed exception data</i> )                                       |
| <i>"Show dialog in 'Detailed' mode"</i>    | Show the Exception-Dialog as the customer has pressed the 'Detail' button                                                 |
| <i>"Show dialog in Top-Most mode"</i>      | Show the Exception-Dialog in Top-Most mode ( <i>in Top to all displayed windows</i> )                                     |
| <i>"Use EurekaLog 'Look and Feel'"</i>     | Display the Exception-Dialog using it's personal 'Look and Feel' ( <i>ignoring the standard Windows 'Look and Feel'</i> ) |
| <i>"Close every dialog after..."</i>       | Automatically close every Exception-Dialog after a customizable time of unused                                            |
| <i>"Support URL"</i>                       | URL of the support product home-page ( <i>will be displayed at the Exception-Dialog left-bottom corner</i> )              |

### **HTML Layout:**

A custom HTML code used to show the HTML error page (Web application only).  
Use the <%HTML\_TAG%> to indicate where the error message will be show.

## 2.5 Advanced Tab



In this section you may customize all the available "Advanced Options".

### **Call-Stack Options:**

*"Show the DLLs functions"*

Display all the DLLs functions contained in call-stack

*"Show the BPLs functions"*

Display all the BPLs functions contained in call-stack

*"Show all Borland Threads call..."*

Display all the running Borland Threads functions contained in call-stack

*"Show all Windows Threads call..."*

Display all the running Windows Threads functions contained in call-stack

*"Show the Procedure-Offset-Line"*

Display after the Line # the Procedure Offset # (*the difference from the Line # to the first procedure Line #*).

It's possible use this value to obtain the exact Line # in sources modified after the software delivery

*"Encryption Password"*

Password use to encrypt all the debug data included in the compiled file (*the password isn't stored in the final compiled file so never hacker can use the debug information to crack your software*). You can use the [EurekaLog Viewer](#)<sup>[66]</sup> to decrypt the encrypted Exception Logs.

**NOTE:** See the [PasswordRequest](#)<sup>[36]</sup> event for further details about the encryption method.

### **Anti-Freeze Options:**

In this panel, you can select how EurekaLog detects that your application has frozen. You can choose to activate this feature, set the timeout needed to establish that the application is frozen and set the message text to be shown when the application freezes.

If EurekaLog detects a frozen application, it raises an [EFrozenApplication](#)<sup>[63]</sup> exception with your custom message text.

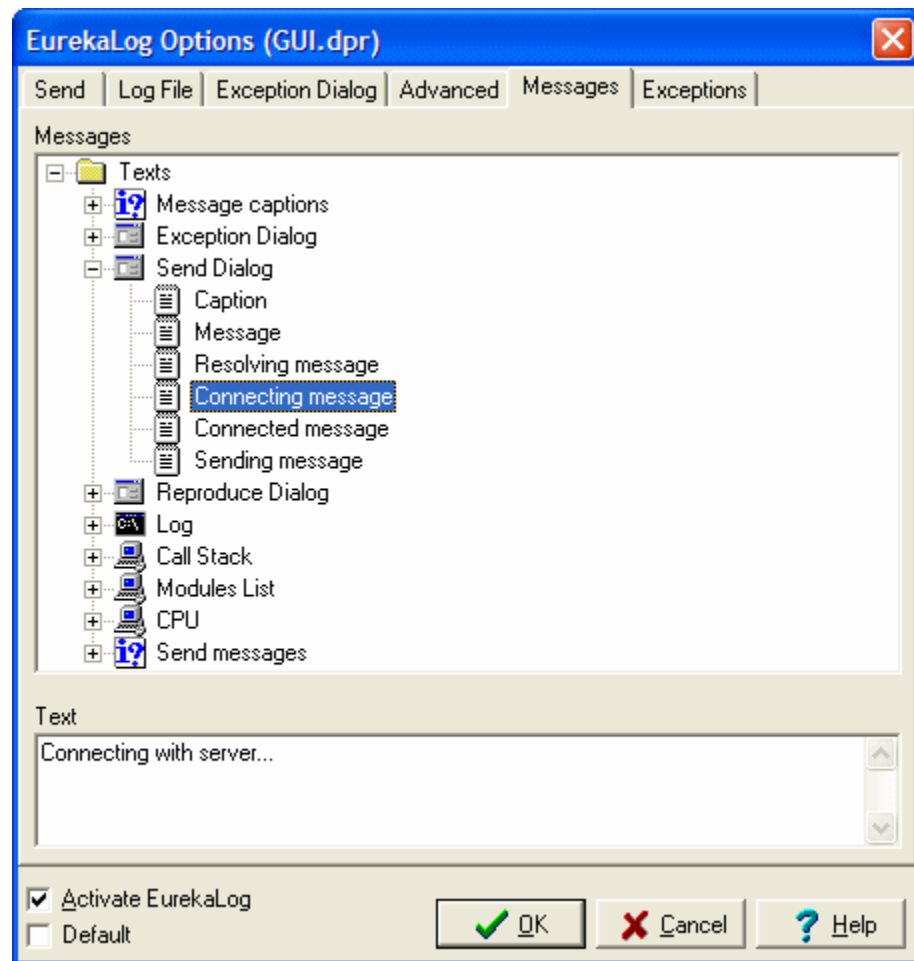
### **Behaviour Options:**

|                                             |                                                                                  |
|---------------------------------------------|----------------------------------------------------------------------------------|
| <i>"Auto Terminate/Restart"</i>             | Terminate/restart the application after a customizable crashed number            |
| <i>"Pause all Borland Threads"</i>          | Pause all Borland Threads during the Exception Dialog displaying                 |
| <i>"Don't pause Main Thread"</i>            | Do not pause the Main Thread during the Exception Dialog displaying              |
| <i>"Pause all Widnows Threads"</i>          | Pause all Windows Threads during the Exception Dialog displaying                 |
| <i>"Use Main Module options"</i>            | Use the EurekaLog options of Main Module ( <i>only for the library modules</i> ) |
| <i>"Copy Log in case of wrong sending"</i>  | Copy the Log text into the clipboard in case of wrong sending                    |
| <i>"Save a Compressed Files copy in..."</i> | Save a compressed copy of files to send in case of wrong sending                 |

### **Options Files:**

In this panel, you can save the current EurekaLog options to an options file, or restore the options saved in an options file. The "Open Log File" button lets you to examine the contents of the current Log File, if one exists.

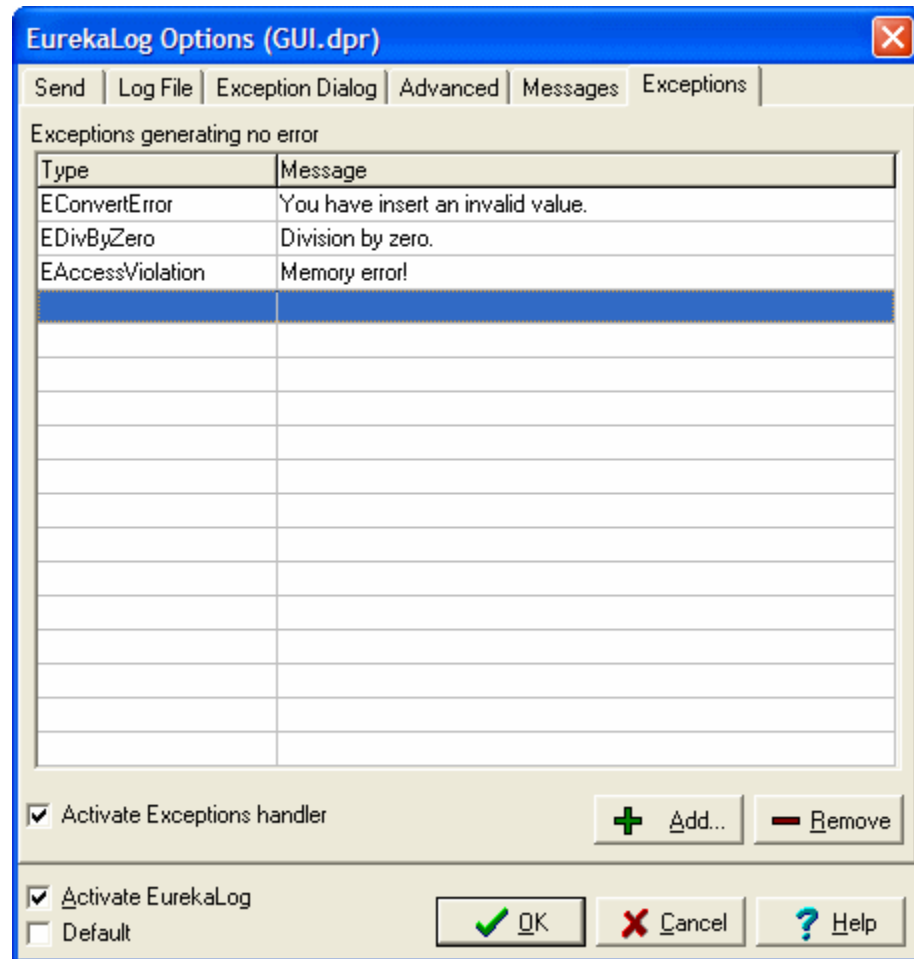
## 2.6 Messages Tab



Here you can customize the texts of EurekaLog so you can use EurekaLog in your favourite language, or just to show a customized text.

First select a message in the "Messages" window and then customize it in the "Text" box underneath.

## 2.7 Exceptions Tab



With the options on this tab you can customize the exception types that should be processed in a different way.

When an exception on this list is intercepted, it will not produce an error report but instead will display the indicated customized error text.

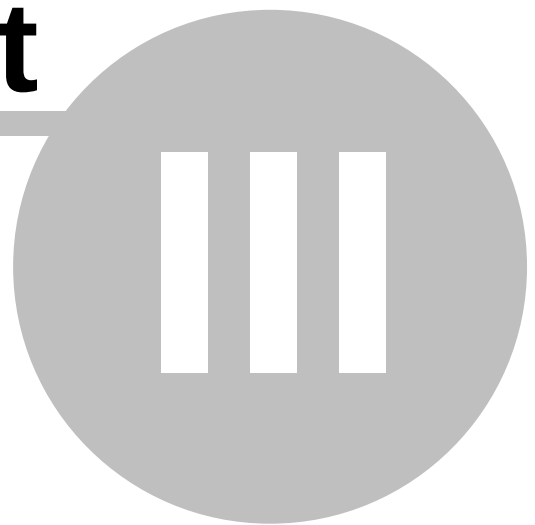
If EurekaLog finds an empty error text, then it ignores the exception.

Use the "Add" key to add a new exception type and its related custom message. Use the "Remove" key to remove the selected exception type.

Use the "Activate Exceptions handler" option to activate/deactivate this feature.

# Part

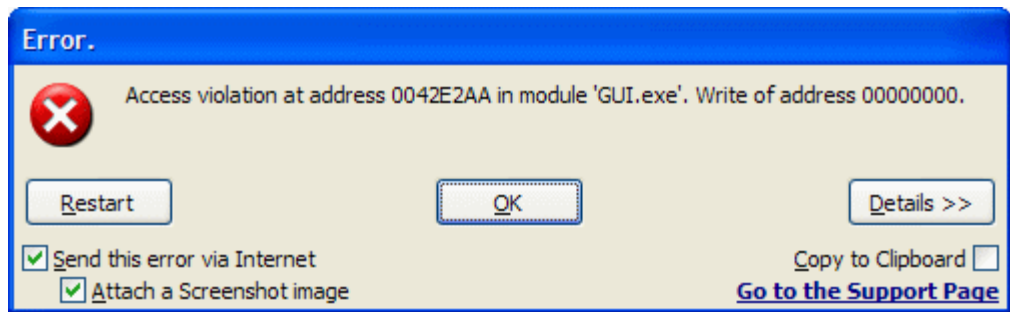
---



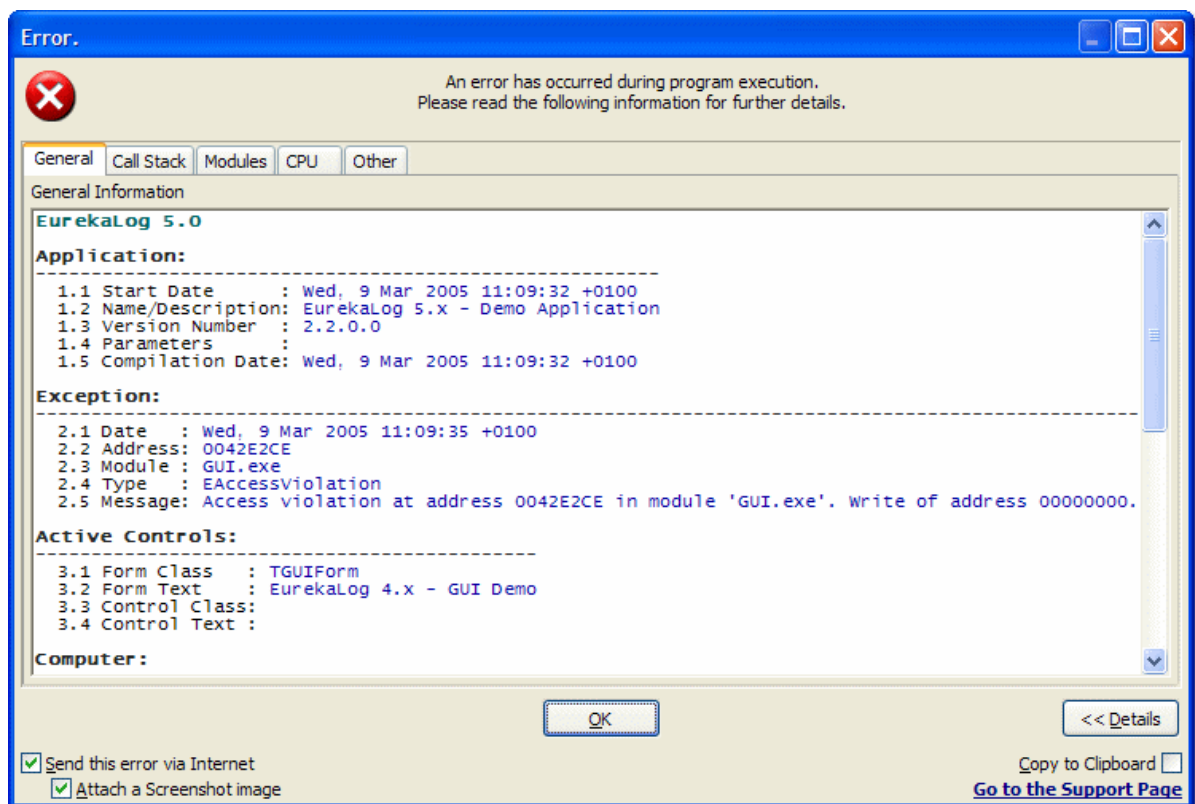
## 3 Types of Application

### 3.1 GUI

Information displayed when EurekaLog runs in a GUI (standard graphical) application:

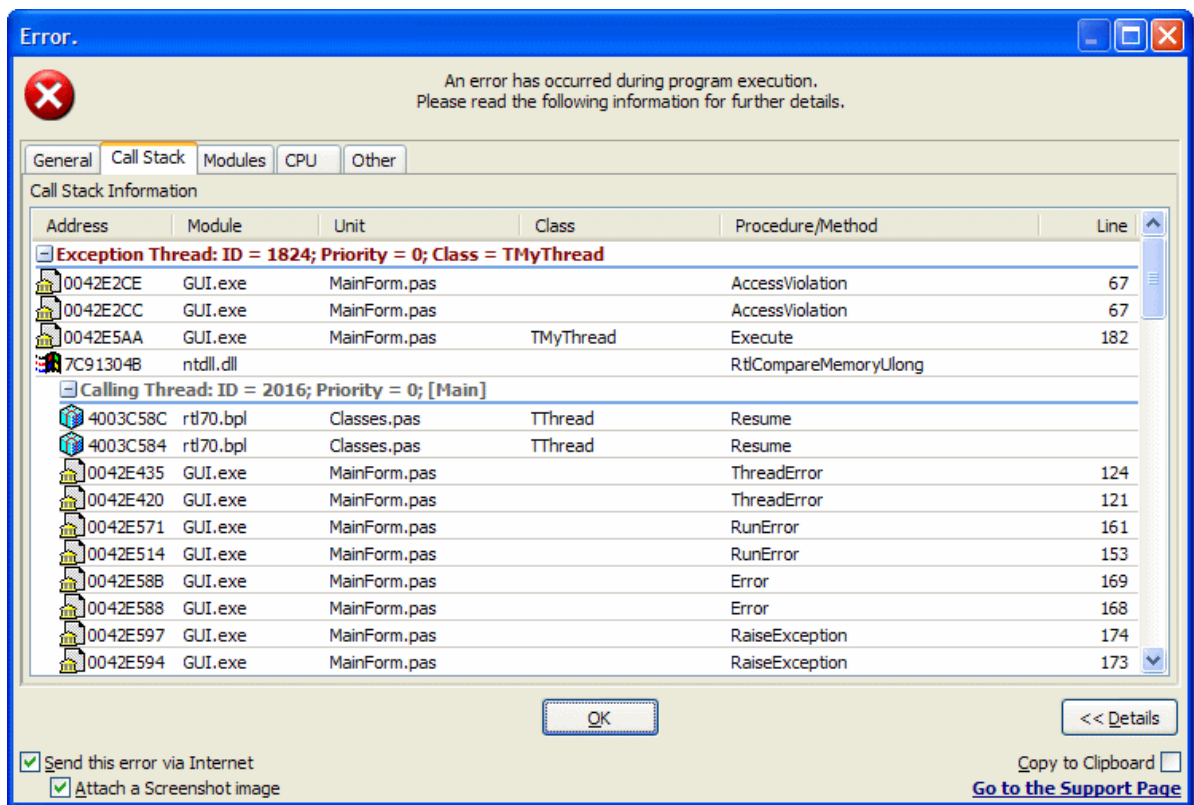


Pressing the "Details" button you can show detailed information about the exception.

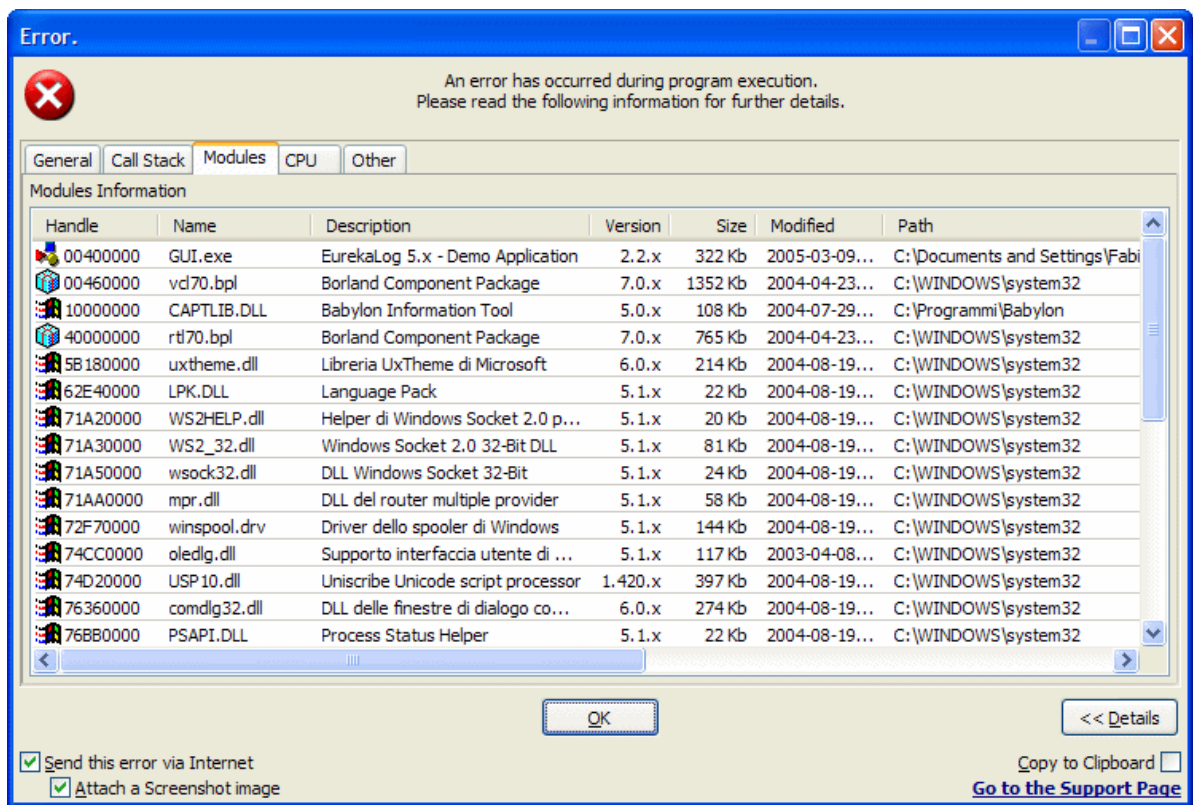


General information.

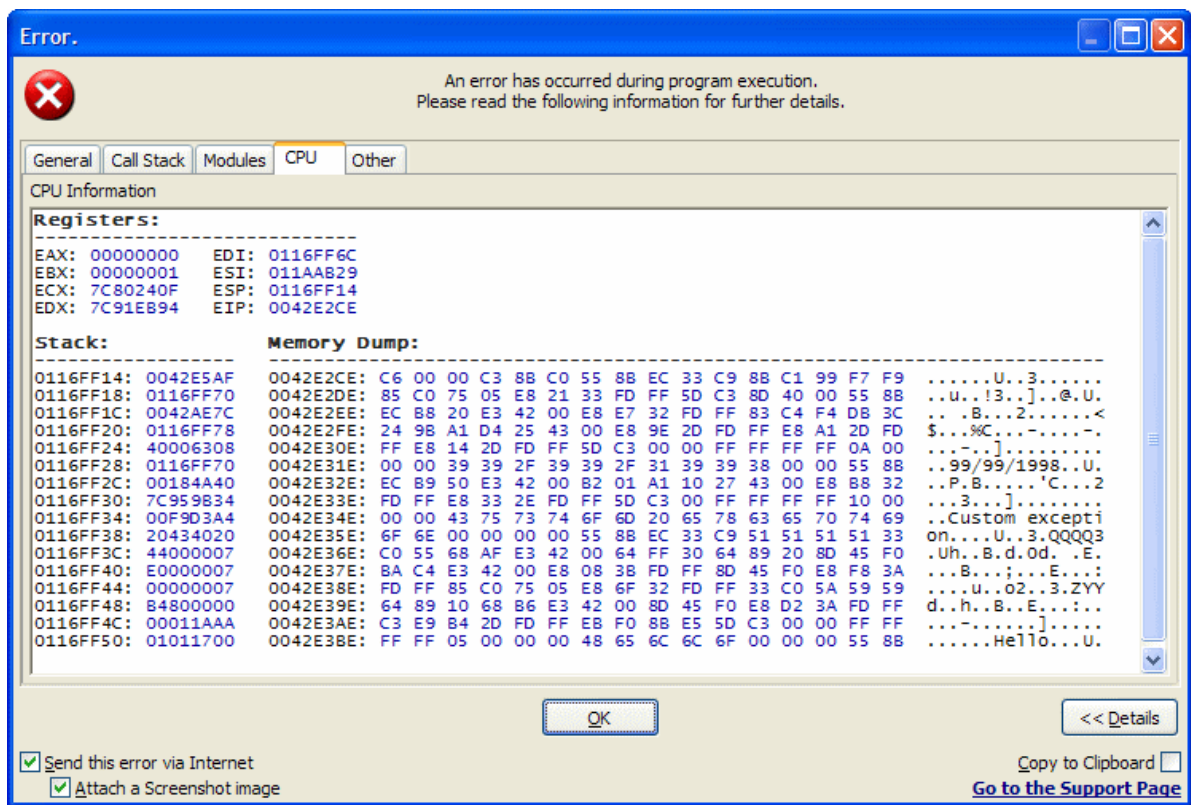




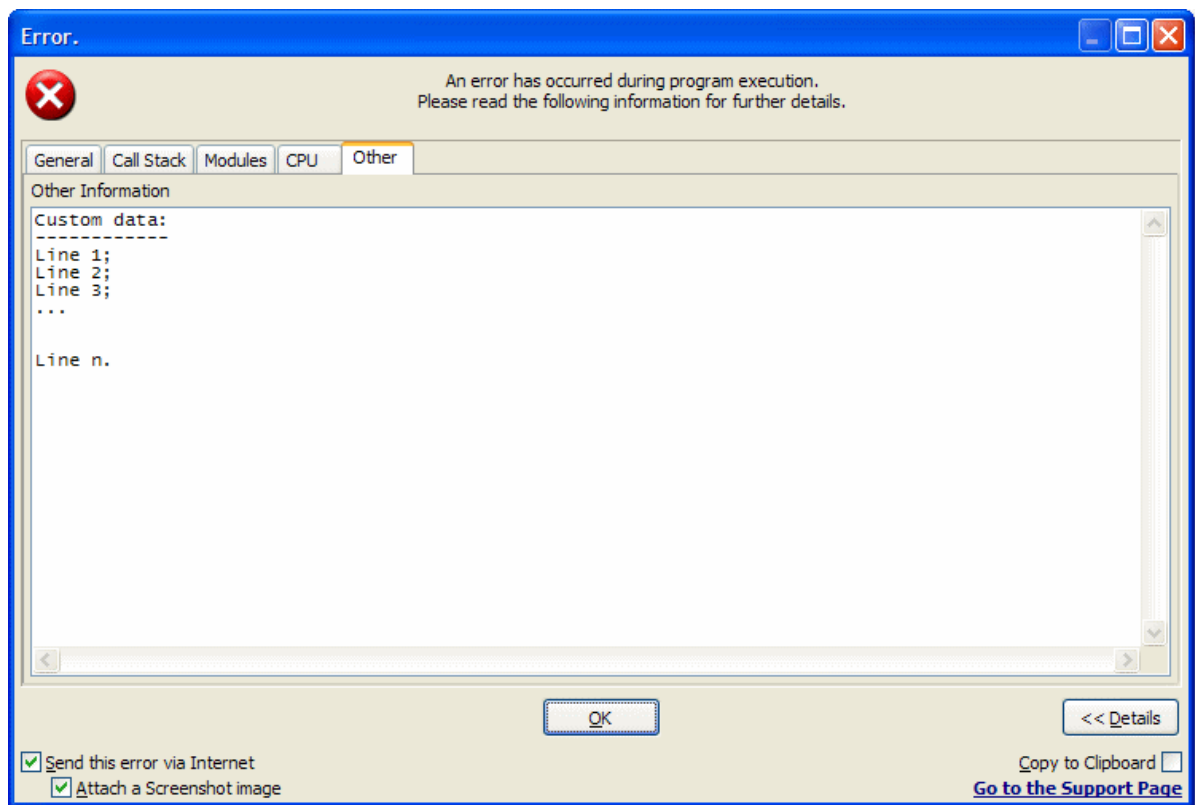
Call-Stack information for any running thread and relative calling thread.



The Modules list.



CPU details.



Custom information.

## 3.2 Console

EurekaLog displays this information when running in a Console application:

```

C:\WINDOWS\system32\cmd.exe
An error has occurred during program execution.
Please read the following information for further details.

EurekaLog 5.0

Application:
-----
1.1 Start Date      : Wed, 9 Mar 2005 11:46:33 +0100
1.2 Name/Description: Console.exe
1.3 Version Number  :
1.4 Parameters      :
1.5 Compilation Date: Wed, 9 Mar 2005 11:46:20 +0100

Exception:
-----
2.1 Date   : Wed, 9 Mar 2005 11:46:34 +0100
2.2 Address: 004146C8
2.3 Module  : Console.exe
2.4 Type    : EListError
2.5 Message : List index out of bounds (0).

Computer:
-----
4.1 Name       : Desktop
4.2 User        : Max
4.3 Total Memory : 511 Mb
4.4 Free Memory  : 309 Mb
4.5 Total Disk   : 55.89 Gb
4.6 Free Disk    : 47.53 Gb
4.7 System Up Time: 20:22:31:24
4.8 Processor    : AMD Athlon(tm) 64 Processor 3000+
4.9 Display Mode  : 1024 x 768 x 32

Operating System:
-----
5.1 Type       : Microsoft Windows XP
5.2 Build #    : 2600
5.3 Update     : Service Pack 2
5.4 Language   : English (USA)

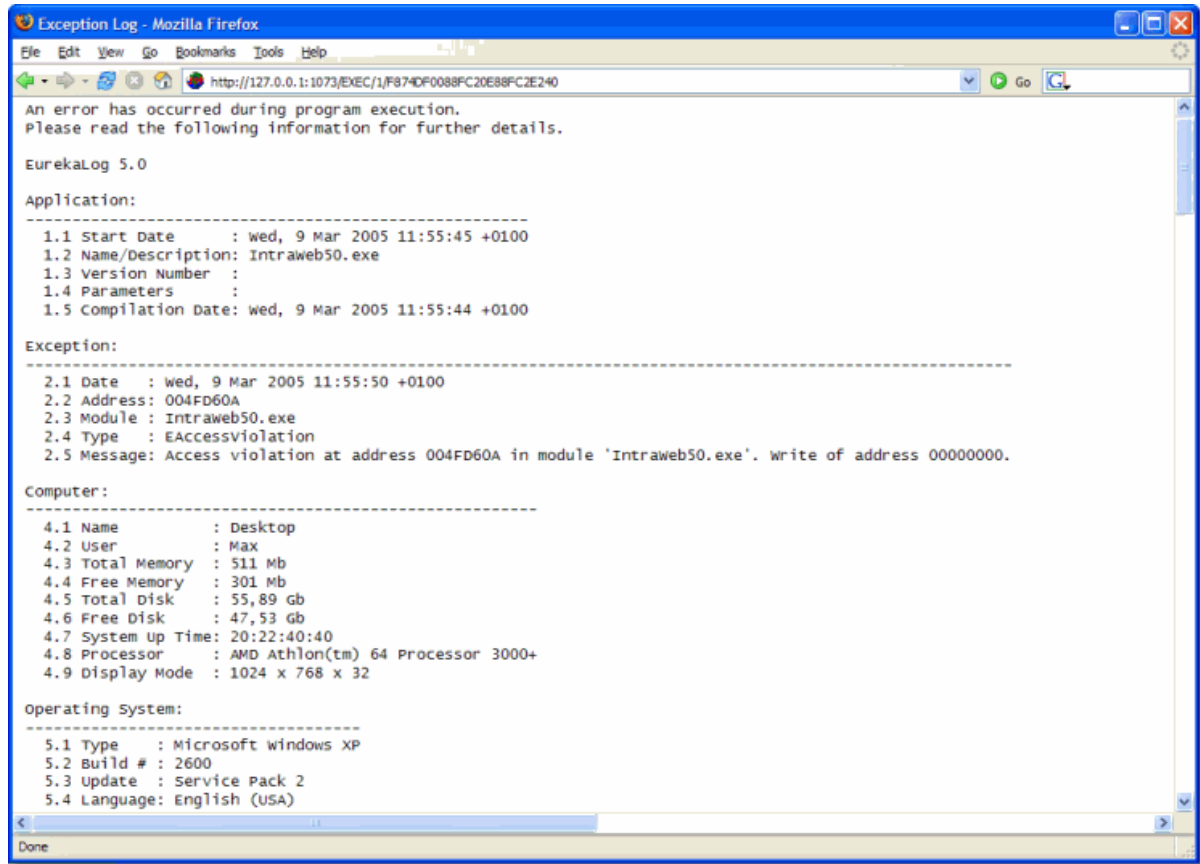
Network:
-----
6.1 IP Address: 0.0.0.0
6.2 Submask   : 0.0.0.0
6.3 Gateway    :
6.4 DNS 1     : 192.168.1.1
6.5 DNS 2     :
6.6 DHCP      : OFF

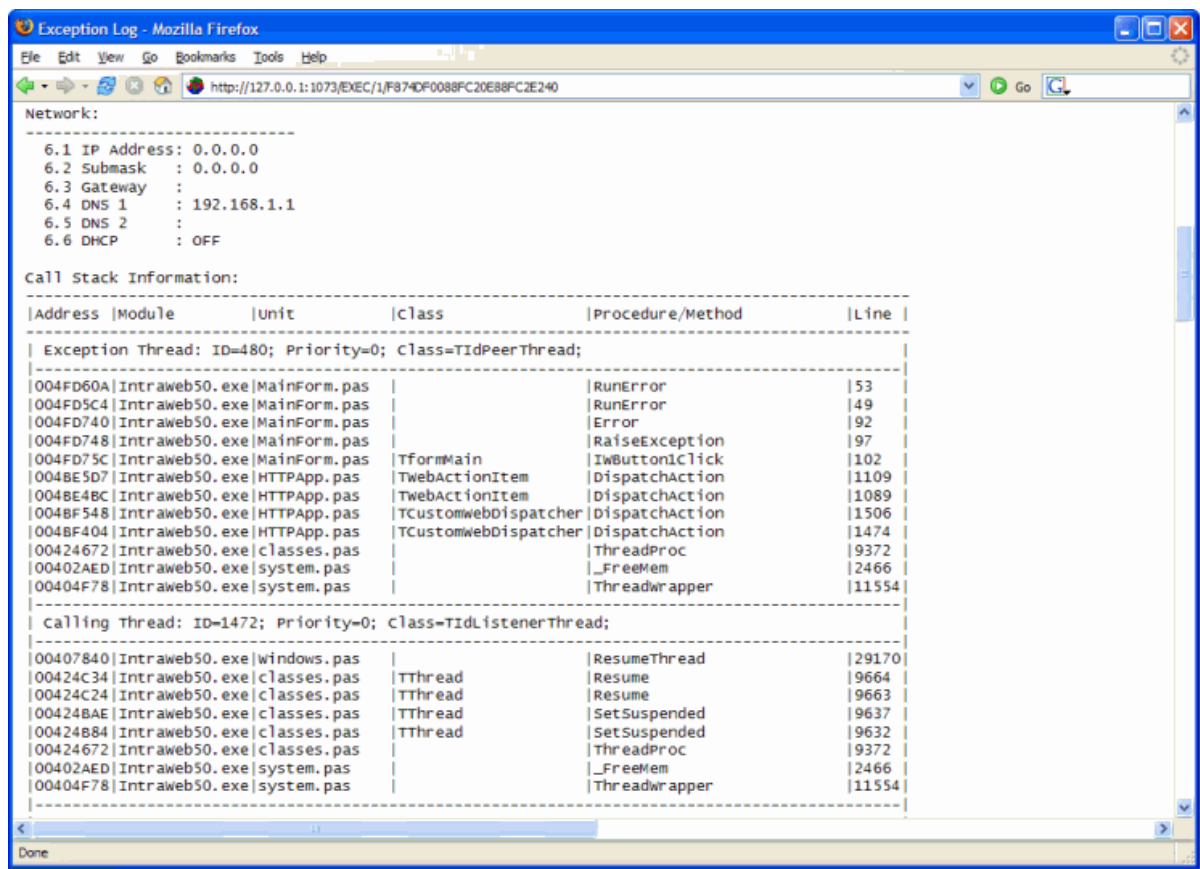
Call Stack Information:
-----
|Address|Module      |Unit      |Class|Procedure/Method |Line|
|-----|-----|-----|-----|-----|-----|
| Exception Thread: ID=1208; Priority=0; Class=; [Main]
|-----|-----|-----|-----|-----|-----|
|004146C8|Console.exe|classes.pas|TList|Error             |2790|
|004146B8|Console.exe|classes.pas|TList|Error             |2789|
|00414731|Console.exe|classes.pas|TList|Error             |2795|
|004146F8|Console.exe|classes.pas|TList|Error             |2794|
|00414783|Console.exe|classes.pas|TList|Get               |2826|
|00404247|Console.exe|system.pas |      |_AfterConstruction|9066|

```

### 3.3 Web

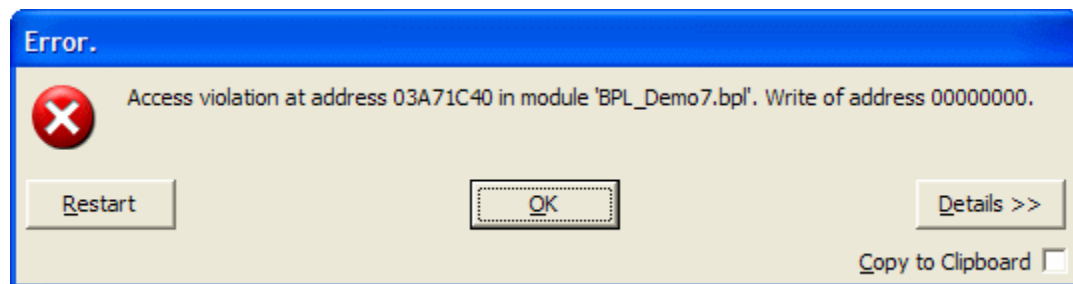
Information shown by EurekaLog when it runs in a Web application (only for Professional and Enterprise versions):



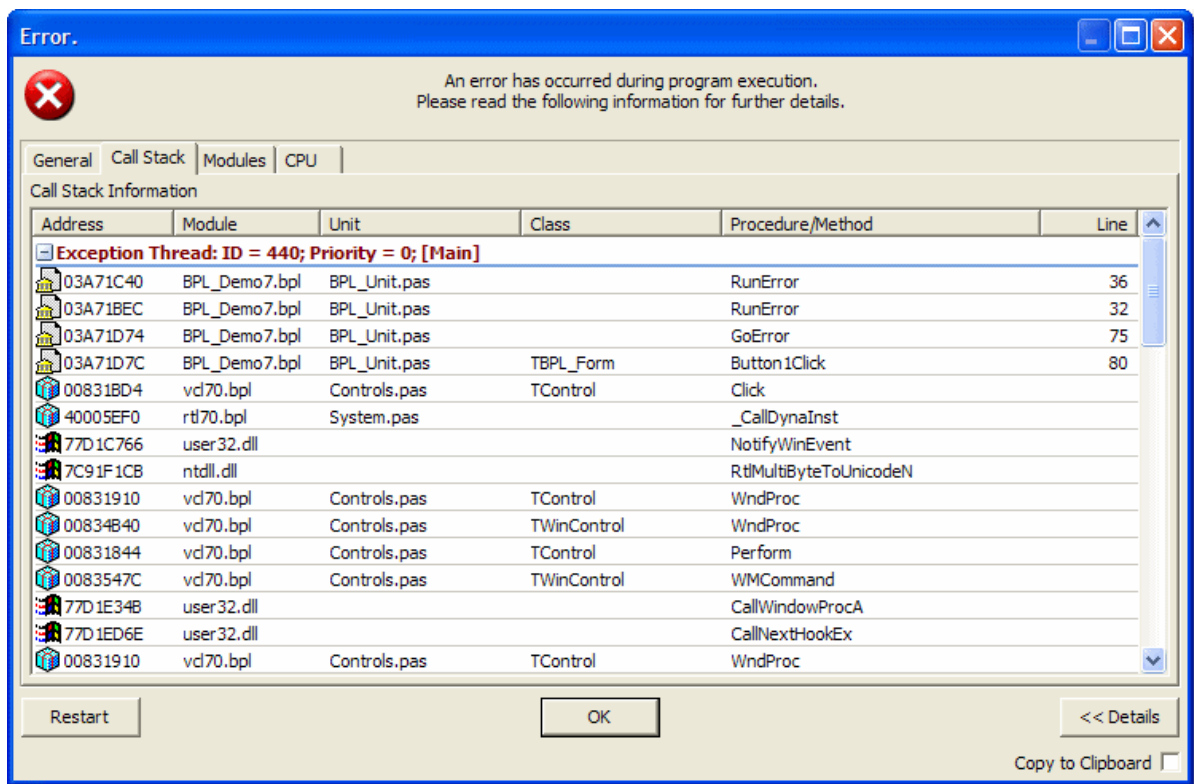


### 3.4 Design-Time

Dialog shown by EurekaLog when an exception is intercepted in the Delphi/CBuilder IDE:



Pressing the "Details" button shows detailed information about the exception. Pressing the "Terminate" button, you can terminate your Delphi/CBuilder session saving all modified files (very usefully when the IDE crashes).



This window shows all VCL and User Source Code points executed before the raised exception. You can debug Delphi/CBuilder BPLs with the IDE using this information.



**Part**

---

**IV**

## 4 Events

### 4.1 ExceptionNotify event

#### 4.1.1 How to use this event

EurekaLog includes an "ExceptionNotify" event for interaction with the user program when an exception is raised. The user-defined routine is called when EurekaLog traps an exception.

The syntax of this event is :

```
procedure MyNotify(ExcRecord: TEurekaExceptionRecord[57]; var Handled: Boolean);
```

To use this event, you must declare a procedure with those parameters and assign it to the ExceptionNotify EurekaLog variable, as shown in the following example:

```
uses ExceptionLog, ECore, ETypes; // The required units...  
// This is a normal procedure (not a method)...  
procedure MyNotify(ExcRecord: TEurekaExceptionRecord[57]; var Handled: Boolean);  
begin  
    ... // Your code...  
    ...  
end;  
  
begin  
    ExceptionNotify := MyNotify; // Assign ExceptionNotify variable to MyNotify procedure.  
end.
```

If you set the "Handled" parameter to *False* in your routine before returning, the exception will be reported by the standard Borland Exception Manager. If you set "Handled" to *True* (the default value) the exception will be handled by EurekaLog.

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnExceptionNotify](#)<sup>[53]</sup> event.

## 4.1.2 Examples

Some examples:

```
// Check unit name...
procedure MyNotify(ExcRecord: TEurekaExceptionRecord; var Handled: Boolean);
begin
    // Check for SourceCode information...
    if (ExcRecord.CallStack[0]^.DebugDetail = ddSourceCode) and
        (ExcRecord.CallStack[0]^.UnitName = 'System.pas') then Handled := False;
end;

// Show exception dialog?
procedure MyNotify(ExcRecord: TEurekaExceptionRecord; var Handled: Boolean);
begin
    with ExcRecord.CurrentModuleOptions do
    begin
        if (MessageBox(0, 'Do you want to show error details?', 'Question',
            MB_YESNO or MB_ICONQUESTION or MB_TASKMODAL) = ID_YES) then
            ExceptionDialogOptions := (ExceptionDialogOptions + [edoShowExceptionDialog])
        else
            ExceptionDialogOptions := (ExceptionDialogOptions - [edoShowExceptionDialog]);
        end;
    end;

// Don't save the Log File...
procedure MyNotify(ExcRecord: TEurekaExceptionRecord; var Handled: Boolean);
begin
    ExcRecord.CurrentModuleOptions.SaveLogFile := False;
end;

// Don't handle this exception with EurekaLog but with standard Borland Exception Manager...
procedure MyNotify(ExcRecord: TEurekaExceptionRecord; var Handled: Boolean);
begin
    Handled := False;
end;

// Check exception type...
procedure MyNotify(ExcRecord: TEurekaExceptionRecord; var Handled: Boolean);
var Error: Exception;
begin
    if ExcRecord.ExceptionObject is Exception then
    begin
        Error := Exception(ExcRecord.ExceptionObject); // Obtain exception type...
        MessageBox(0, PChar(Format('Error: %s', [Error.Message])), 'Error',
            MB_OK or MB_ICONSTOP or MB_TASKMODAL);
        ExcRecord.CurrentModuleOptions.ShowExceptionDialog := False; // Don't show exception dialog.
        end;
    end;
```

## 4.2 ExceptionActionNotify event

### 4.2.1 How to use this event

EurekaLog includes an "ExceptionActionNotify" event for interaction with the user program when an exception is raised and is being handled by EurekaLog. This user-supplied routine is called at each of several points within the EurekaLog system.

The syntax of this event is:

```
procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord57;
    EurekaAction: TEurekaActionType60; var Execute: Boolean);
```

To use this event you must create a routine with the indicated parameters and assign it to the `ExceptionActionNotify` EurekaLog variable, as shown in the following example:

```
uses ExceptionLog, ECore, ETypes; // The required units...

// This is a normal procedure (not a method)...
procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
    ... // Your code...
end;

begin
    // Assign ExceptionActionNotify variable to MyActionNotify procedure...
    ExceptionActionNotify := MyActionNotify;
end.
```

The user-supplied routine is called at each of seven action points within EurekaLog:

|                                     |                                                                                |
|-------------------------------------|--------------------------------------------------------------------------------|
| <code>atShowingExceptionInfo</code> | (before): call made before showing exception information.                      |
| <code>atShowedExceptionInfo</code>  | (after) : call made after showing exception information.                       |
| <code>atSavingLogFile</code>        | (before): call made before saving Log File.                                    |
| <code>atSavedLogFile</code>         | (after) : call made after saving Log File.                                     |
| <code>atSendingEmail</code>         | (before): call made before sending Email.                                      |
| <code>atSentEmail</code>            | (after) : call made after sending Email.                                       |
| <code>atSendingWebMessage</code>    | (before): call made before sending Web message                                 |
| <code>atSentWebMessage</code>       | (after) : call made after sending Web message                                  |
| <code>atTerminating</code>          | (before): call made before terminating application (click 'Terminate' button). |

If you set the *"Execute"* parameter to *False* in a **before** action, the standard EurekaLog action will not be executed when your routine returns control to EurekaLog. In **after** actions, the *"Execute"* parameter can be used to check if the action was performed successfully.

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnExceptionActionNotify](#)<sup>53</sup> event.

## 4.2.2 Examples

Some examples:

```
// Show exception dialog?
procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
    if EurekaAction = atShowingExceptionInfo then
    begin
        Execute := MessageBox(0, 'Do you want to show error details?', 'Question',
                               MB_YESNO or MB_ICONQUESTION or MB_TASKMODAL) <> ID_YES;
    end;
end;

// Don't save Log File...
procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
    if EurekaAction = atSavingLogFile then
        Execute := False;
end;

// Check that exception was correctly saved to Log File...
procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Execute: Boolean);
```

```

begin
  if EurekaAction = atSavedLogFile then
    begin
      if not Execute then
        MessageBox(0, 'Cannot write to Log File.', 'Error',
          MB_OK or MB_ICONSTOP or MB_TASKMODAL);
      end;
    end;
end;

// Check Email sent correctly...
procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
  EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
  if EurekaAction=atSentEmail then
    begin
      if not Execute then
        MessageBox(0, 'Could not send the Email.', 'Error',
          MB_OK or MB_ICONSTOP or MB_TASKMODAL);
      end;
    end;
end;

```

## 4.3 ExceptionErrorNotify event

### 4.3.1 How to use this event

EurekaLog includes an "ExceptionErrorNotify" event for interaction with the user program when a raised exception is being handled by EurekaLog and an error occurs.

The syntax of this event is:

```

procedure MyErrorNotify(EurekaExceptionRecord: TEurekaExceptionRecord[57];
  EurekaAction: TEurekaActionType[60]; var Retry: Boolean);

```

To use this event you must create a routine with the indicated parameters and assign it to the ExceptionErrorNotify EurekaLog variable, as shown in the following example:

```

uses ExceptionLog, ECore, ETypes; // The required units...

// This is a normal procedure (not a method)...
procedure MyErrorNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
  EurekaAction: TEurekaActionType; var Retry: Boolean);
begin
  ... // Your code...
end;

begin
  // Assign ExceptionErrorNotify variable to MyErrorNotify procedure...
  ExceptionErrorNotify := MyErrorNotify;
end.

```

The user-supplied routine is called if an error occurs during the processing of any of the following actions:

|                  |                                               |
|------------------|-----------------------------------------------|
| atSavedLogFile   | (after) : call made after saving Log File.    |
| atSentEmail      | (after) : call made after sending Email.      |
| atSentWebMessage | (after) : call made after sending Web message |

If you set "Retry" to *True*, EurekaLog will retry the last action.  
 You can use this event to change some EurekaLog settings in accord with the computer configuration.

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnExceptionErrorNotify](#)<sup>[53]</sup> event.

## 4.3.2 Examples

Some examples:

```
// Change email settings...
procedure MyErrorNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Retry: Boolean);
begin
    // An error occurred when sending the email...
    if EurekaAction = atSentEmail then
        begin
            // Try to send the email via SMTP client.
            EurekaExceptionRecord.CurrentModuleOptions.SMTPHost := 'localhost';
            Retry := True; // Force retry of this action.
        end;
    end;

// Change the log settings...
procedure MyErrorNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Retry: Boolean);
begin
    // An error occurred while saving the Log-File...
    if EurekaAction = atSavedLogFile then
        begin
            // Try to send again after changing the Log-File path.
            EurekaExceptionRecord.CurrentModuleOptions.OutputPath := 'c:\';
            Retry := True; // Force to retry of this action.
        end;
    end;
```

## 4.4 AttachedFilesRequest event

### 4.4.1 How to use this event

EurekaLog includes an "AttachedFilesRequest" event for interaction with the user program when a raised exception is handled by EurekaLog and the user want send custom files with the EurekaLog message (via Email and Web).

The syntax of this event is:

```
procedure MyAttachedFiles(EurekaExceptionRecord: TEurekaExceptionRecord[57]; AttachedFiles: TStrings);
```

To use this event you must create a routine with the indicated parameters and assign it to the AttachedFilesRequest EurekaLog variable, as shown in the following example:

```
uses ExceptionLog, ECore, ETypes; // The required units...

// This is a normal procedure (not a method)...
procedure MyAttachedFiles(EurekaExceptionRecord: TEurekaExceptionRecord; AttachedFiles: TStrings);
begin
    ...
    // Your code...
    ...
end;

begin
    // Assign AttachedFilesRequest variable to MyAttachedFiles procedure...
    AttachedFilesRequest := MyAttachedFiles;
end;
```

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnAttachedFilesRequest](#)<sup>[53]</sup> event.

## 4.4.2 Examples

Some examples:

```
// Attach the "C:\Windows\notepad.exe" and "C:\AutoExec.bat" files to the sending message.
procedure MyAttachedFiles(AttachedFiles: TStrings);
begin
    AttachedFiles.Add('C:\Windows\notepad.exe');
    AttachedFiles.Add('C:\AutoExec.bat');
end;

// Clear all attached files (screenshot last HTML page, ...) and add the "C:\Config.sys" file.
procedure MyAttachedFiles(AttachedFiles: TStrings);
begin
    AttachedFiles.Clear; // Clear all EurekaLog attached files (as screenshot, last HTML page,...)
    AttachedFiles.Add('C:\Config.sys');
end;
```

## 4.5 CustomDataRequest event

### 4.5.1 How to use this event

EurekaLog includes an "CustomDataRequest" event for interaction with the user program when a raised exception is handled by EurekaLog and the user want add some custom information to the Log text.

The syntax of this event is:

```
procedure MyCustomData(EurekaExceptionRecord: TEurekaExceptionRecord[57]; var CustomData: String);
```

To use this event you must create a routine with the indicated parameters and assign it to the CustomDataRequest EurekaLog variable, as shown in the following example:

```
uses ExceptionLog, ECore, ETypes; // The required units...

// This is a normal procedure (not a method)...
procedure MyCustomData(EurekaExceptionRecord: TEurekaExceptionRecord; var CustomData: String);
begin
    ...
    ... // Your code...
    ...
end;

begin
    // Assign CustomDataRequest variable to MyCustomData procedure...
    CustomDataRequest := MyCustomData;
end;
```

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnCustomDataRequest](#)<sup>[53]</sup> event.

## 4.5.2 Examples

Some examples:

```
// Add the "C:\MyData.txt" text file data to the Log data.
procedure MyCustomData(var CustomData: String);
var
    TextFile: TStrings;
begin
```

```

TextFiles := TStringList.Create;
try
  TextFiles.LoadFromFile('C:\MyData.txt'); // Load the file content
  CustomData := TextFiles.Text; // Add the file context to the Log data
finally
  TextFiles.Free;
end;
end;

// Add a global variable (MainSQL) to the Log data.
// Global declaration...
var MainSQL: string;

procedure MyCustomData(var CustomData: String);
begin
  CustomData := MainSQL; // Add the MainSQL variable to the Log data
end;

```

## 4.6 CustomFieldsRequest event

### 4.6.1 How to use this event

EurekaLog includes an "CustomFieldsRequest" event for interaction with the user program when a raised exception is handled by EurekaLog and the user want send a message to a Web server sending custom HTTP fields (via post) together the upload files.

The syntax of this event is:

```

procedure MyCustomFields(EurekaExceptionRecord: TEurekaExceptionRecord[57]; FieldsName, FieldsValue: TString

```

To use this event you must create a routine with the indicated parameters and assign it to the CustomFieldsRequest EurekaLog variable, as shown in the following example:

```

uses ExceptionLog, ECore, ETypes; // The required units...

// This is a normal procedure (not a method)...
procedure MyCustomFields(EurekaExceptionRecord: TEurekaExceptionRecord[57]; FieldsName, FieldsValue: TString
begin
  ...
  ... // Your code...
  ...
end;

begin
  // Assign CustomFieldRequest variable to MyCustomField procedure...
  CustomFieldsRequest := MyCustomFields;
end.

```

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnCustomFieldsRequest](#)<sup>[53]</sup> event.

### 4.6.2 Examples

Some examples:

```

// Add userName and Password fields to the sending HTML page.
procedure MyCustomFields(FieldsName, FieldsValue: TString);
begin
  // First HTTP field (UserName)...
  FieldsName.Add('UserName');
  FieldsValue.Add('MyUserName');

  // Second HTTP field (Password)...
  FieldsName.Add('Password');

```



```

    FieldsValue.Add('xyz');
end;

// Add the User Email field to the sending HTML page.
procedure MyCustomFields(FieldsName, FieldsValue: TStrings);
begin
    FieldsName.Add('Email');
    FieldsValue.Add('myemail@email.com');
end;

```

## 4.7 PasswordRequest event

### 4.7.1 How to use this event

EurekaLog includes an "PasswordRequest" event for interaction with the user program when a raised exception is handled by EurekaLog and all the Debug data (Unit, Class and Procedure names) are saved as encrypted values.

This event allow to obtain the exact password required to can decrypt all the encrypted data, otherwise all the data will be store in the Log text in encrypted form (the software author will be decrypt this data using the decrypt capabilities of [EurekaLog Viewer](#)<sup>[66]</sup> software).

The syntax of this event is:

```

procedure MyPasswordRequest(EurekaExceptionRecord: TEurekaExceptionRecord[57]; var Password: String);

```

To use this event you must create a routine with the indicated parameters and assign it to the PasswordRequest EurekaLog variable, as shown in the following example:

```

uses ExceptionLog, ECore, ETypes; // The required units...

// This is a normal procedure (not a method)...
procedure MyPasswordRequest(EurekaExceptionRecord: TEurekaExceptionRecord; var Password: String);
begin
    ...
    ... // Your code...
    ...
end;

begin
    // Assign PasswordRequest variable to MyPasswordRequest procedure...
    PasswordRequest := MyPasswordRequest;
end.

```

**Note:** to simplify management of your EurekaLog events, you can use the [TEurekaLog.OnPasswordRequest](#)<sup>[53]</sup> event.

### 4.7.2 Examples

Some examples:

```

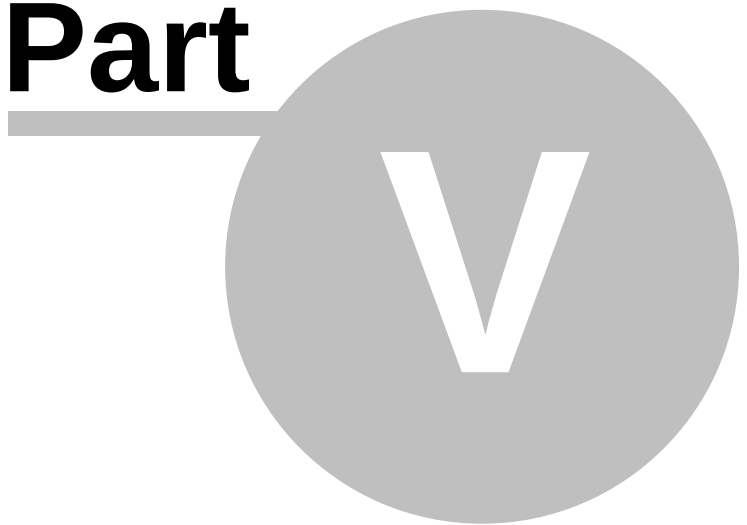
// Obtain the required password asking it to the final user.
procedure MyPasswordRequest(var Password: String);
begin
    InputQuery('Request', 'Insert the BUG report password', Password);
end;

// Obtain the required password by configuration INI file.
procedure MyPasswordRequest(var Password: String);
var
    INI: TIniFile;
begin
    INI := TiniFile.Create('Config.ini');

```

```
try
  Password := INI.ReadString('Main', 'Password', '');
finally
  INI.Free;
end;
end;
```

**Part**



## 5 Generic functions

### 5.1 StandardEurekaNotify function

EurekaLog includes a function called StandardEurekaNotify, that enables you to call EurekaLog "manually".

Syntax of this function is as follows:

```
function StandardEurekaNotify(Obj: TObject; Addr: pointer): Boolean;
```

When EurekaLog is active the Result is *True*, *False* otherwise (i.e. when the *ExceptionLog.pas* unit is included in your project but you have disabled EurekaLog from the "Project/EurekaLog Options..." IDE menu).

Example:

```
uses ExceptionLog; // The required unit...  
begin  
    ... := StandardEurekaNotify(ExceptObject, ExceptAddr);  
    ...  
end;
```

### 5.2 IsEurekaLogInstalled function

EurekaLog includes a function named IsEurekaLogInstalled, that enables you to check if EurekaLog is installed in your application.

Syntax of this function is as follows:

```
function IsEurekaLogInstalled: Boolean;
```

When EurekaLog is active the Result is *True*, *False* otherwise (when the *ExceptionLog.pas* unit is included in your project but you have disabled EurekaLog from the "Project/Exceptions Log Options..." IDE menu).

Example:

```
uses ExceptionLog; // The required unit...  
begin  
    ...  
    ok := IsEurekaLogInstalled;  
    ...  
end;
```

### 5.3 CurrentEurekaLogOptions function

EurekaLog includes a function named CurrentEurekaLogOptions, that provides access to the EurekaLog options of the current module.

Syntax of this function is as follows:

```
function CurrentEurekaLogOptions: TEurekaModuleOptions .
```

When EurekaLog is active the Result is the EurekaLog options of the calling module, otherwise this function raises an exception (you can first check that EurekaLog is installed by calling the [IsEurekaLogInstalled](#)<sup>[39]</sup> function).

**Note:**

Don't use this function within EurekaLog events. Use instead the [TEurekaExceptionRecord.CurrentModuleOptions](#)<sup>[57]</sup> parameter.

Example:

```
uses ExceptionLog, ECore, ETypes; // The required units...

begin
    ...
    CurrentEurekaLogOptions.EmailAddresses := 'bugs@eurekalog.com';
    CurrentEurekaLogOptions.ShowExceptionDialog := False;
    CurrentEurekaLogOptions.OutputPath := ExtractFilePath(ParamStr(0));
    ...
end;
```

## 5.4 ForceApplicationTermination function

EurekaLog includes a function named [ForceApplicationTermination](#), that enables you to force your application to terminate/restart after handling the current exception.

Syntax of this function is as follows:

```
function ForceApplicationTermination(TrmType: TTerminateBtnOperation[64]): Boolean;
```

Use the "TrmType" parameter to choose the termination type (simple termination / termination with restart).

When EurekaLog is active the Result is True, False otherwise (when the ExceptionLog.pas unit is included in your project but you have disabled EurekaLog from the "Project/Exceptions Log Options..." IDE menu).

Example:

```
uses ExceptionLog, ECore; // The required units...

begin
    ...
    ok := ForceApplicationTermination(tbRestart); // Force the application to restart.
    ...
end;
```

## 5.5 StandardEurekaError function

EurekaLog includes a function named [StandardEurekaError](#), that enables you to call EurekaLog "manually" simulating an exception.

Syntax of this function is as follows:

```
function StandardEurekaError(const Error: string): Boolean;
```

When EurekaLog is active the Result is *True*, otherwise it returns *False* (when the *ExceptionLog.pas*

unit is included in your project but you have EurekaLog disabled in the "Project/Exceptions Log Options..." IDE menu).

With this function you can obtain the current call-stack without raising an exception.

This function simulates the raising of an [EEurekaLogGeneralError](#)<sup>[64]</sup> exception with the Exception.Message = Error.

Example:

```
uses ExceptionLog; // The required unit...

begin
    ok := StandardEurekaError('General error!');
end;
```

## 5.6 GetLastEurekaLogErrorCode function

EurekaLog includes a function named [GetLastEurekaLogErrorCode](#), that enables you to obtain the last internal EurekaLog error code.

Syntax of this function is as follows:

**function** GetLastEurekaLogErrorCode: [TEurekaLogErrorCode](#)<sup>[64]</sup>;

You should use this function in the EurekaLog [ExceptionActionNotify](#)<sup>[29]</sup> event to check if for errors in the email sending procedure (or other).

Example:

```
uses ExceptionLog, ECore, ETypes; // The required units...

procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
    EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
    if (EurekaAction = atSentEmail) and (GetLastEurekaLogErrorCode = eeEmailMAPIError) then
    begin
        MessageBox(0, 'Cannot connect with the email client using the MAPI protocol.', 'Error',
            MB_OK or MB_ICONERROR);
    end;
end;
```

## 5.7 GetLastEurekaLogErrorMsg function

EurekaLog includes a function named [GetLastEurekaLogErrorMsg](#), that enables you to obtain the last internal EurekaLog error message.

Syntax of this function is as follows:

**function** GetLastEurekaLogErrorMsg: **string**;

You should use this function in the EurekaLog [ExceptionActionNotify](#)<sup>[29]</sup> event to check if for errors in the email sending procedure (or other).

Example:

```

uses ExceptionLog, ECore, ETypes; // The required units...

procedure MyActionNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
                        EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
    if (EurekaAction = atSentEmail) and
        (GetLastEurekaLogErrorCode in [eeEmailMAPIError, eeEmailShellError, eeEmailSMTPError]) then
        begin
            MessageBox(0, PChar(GetLastEurekaLogErrorMsg), 'Email error', MB_OK or MB_ICONERROR);
        end;
end;

```

## 5.8 SetEurekaLogInThread procedure

EurekaLog includes a procedure called *SetEurekaLogInThread*, that enables you to activate/deactivate EurekaLog in a specified thread.

Syntax of this function is as follows:

```

procedure SetEurekaLogInThread(ThreadID: DWord; Activate: Boolean);

```

The *ThreadID* parameter indicates the Thread ID and *Activate* indicates the EurekaLog state (Active/Disactive).

Example:

```

uses ExceptionLog; // The required unit...

begin
    ...
    SetEurekaLogInThread(GetCurrentThreadID, False); // Disable EurekaLog in the current Thread.
    ...
end;

```

## 5.9 IsEurekaLogActiveInThread function

EurekaLog includes a function called *IsEurekaLogActiveInThread*, that enables you to check the EurekaLog state in a specified thread.

Syntax of this function is as follows:

```

function IsEurekaLogActiveInThread(ThreadID: DWord): Boolean;

```

The *ThreadID* parameter indicates the Thread ID.

Example:

```

uses ExceptionLog; // The required unit...

begin
    ...
    OK := IsEurekaLogActiveInThread(GetCurrentThreadID); // Get the state of EurekaLog in current Thread.
    ...
end;

```

## 5.10 SetEurekaLogState procedure

EurekaLog includes a procedure called *SetEurekaLogState*, that enables you to activate/deactivate EurekaLog.

Syntax of this function is as follows:

```
procedure SetEurekaLogState(Activate: Boolean);
```

The Activate parameter indicates the EurekaLog state (Active/Disactive).

Example:

```
uses ExceptionLog; // The required unit...
begin
    ...
    SetEurekaLogState(False); // Disable EurekaLog.
    ...
end;
```

## 5.11 IsEurekaLogActive function

EurekaLog includes a function called IsEurekaLogActive, that enables you to check the EurekaLog state.

Syntax of this function is as follows:

```
function IsEurekaLogActive: Boolean;
```

Example:

```
uses ExceptionLog; // The required unit...
begin
    ...
    OK := IsEurekaLogActive; // Get the state of EurekaLog.
    ...
end;
```

## 5.12 EurekaLog version constants

EurekaLog includes two constants, EurekaLogCurrentVersion (Word) and EurekaLogVersion (String) that enables you to check the current EurekaLog version.

Example:

```
uses EConsts; // The required unit...
begin
    ...
    // Write 4.5.0 for the EurekaLog 4.5.0 version.
    WriteLn(EurekaLogVersion);

    // Write 'OK' if EurekaLog version if 4.2.0 or higher.
    if (EurekaLogCurrentVersion >= 420) then WriteLn('OK');
    ...
end;
```

## 5.13 SaveScreenshot procedure

EurekaLog includes a function called SaveScreenshot, that enables you to save the current screenshot to a file (.PNG format). The saving options are gets by the EurekaLog options.

Syntax of this function is as follows:



```
procedure SaveScreenshot(const Filename: string);
```

Example:

```
uses ExceptionLog; // The required unit...

begin
    ...
    SaveScreenshot('C:\Screenshots\MyFile.png');
    ...
end;
```

## 5.14 SaveScreenshotToStream procedure

EurekaLog includes a function called *SaveScreenshotToStream*, that enables you to save the current screenshot to a stream (.PNG format). The saving options are gets by the EurekaLog options.

Syntax of this function is as follows:

```
procedure SaveScreenshotToStream(Stream: TStream);
```

Example:

```
uses ExceptionLog; // The required unit...

var
    FS: TFileStream;
begin
    FS := TFileStream.Create('Screenshot.png', fmCreate);
    try
        SaveScreenshotToStream(FS);
    finally
        FS.Free;
    end;
end;
```

## 5.15 IsEurekaLogModule function

EurekaLog includes a function called *IsEurekaLogModule*, that enables you to check if a module has been compiled with the EurekaLog support.

Syntax of this function is as follows:

```
function IsEurekaLogModule(HModule: THandle): Boolean;
```

Example:

```
uses ExceptionLog; // The required unit...

procedure LoadLib;
var
    Lib: THandle;
    Res: Boolean;
begin
    Lib := LoadLibrary('C:\Library.dll');
    if (Lib <> 0) then
        try
            Res := IsEurekaLogModule(Lib);
            if (Res) then
```

```

        MessageBox(0, 'The "Lybrary.dll" module has been compiled WITH the EurekaLog support.',
        'Information', MB_OK or MB_ICONINFORMATION)
    else
        MessageBox(0, 'The "Lybrary.dll" module has been compiled WITHOUT the EurekaLog support.',
        'Information', MB_OK or MB_ICONINFORMATION)
    finally
        FreeLibrary(Lib);
    end;
end;
end;

```

## 5.16 GetEurekaLogModuleVersion function

EurekaLog includes a function called *GetEurekaLogModuleVersion*, that enables you to obtain the EurekaLog version used to compile a specified module (0 if the module it isn't compiled with EurekaLog, 400 if is compiled with EurekaLog 4.0.0, 512 if is compiled with EurekaLog 5.1.2, ...).

Syntax of this function is as follows:

```
function GetEurekaLogModuleVersion(HModule: THandle): Word;
```

Example:

```

uses ExceptionLog; // The required unit...

// This function returns the EurekaLog version used to compile a specified file...
function GetEurekaLogFileVersion(const FileName: string): Word;
var
    Module: THandle;
begin
    Result := 0;
    // Load the file to obtain its Handle...
    Module := LoadLibrary(PChar(FileName));
    if (Module <> 0) then
        try
            Result := GetEurekaLogModuleVersion(Module);
        finally
            // Unload the file...
            FreeLibrary(Module)
        end;
    end;

```

## 5.17 GetCompilationDate function

EurekaLog includes a function called *GetCompilationDate*, that enables you to obtain the compilation date of a specified module (in local or global-GMT time coordinates).

Syntax of this function is as follows:

```
function GetCompilationDate(HModule: THandle; LocalTime: Boolean; var Date: TDateTime): Boolean;
```

The first parameter indicate the module handle (HInstance for the current module), the second indicate the result format and the last parameter indicate the result value. The function returns True if the required module was compiled with EurekaLog and False otherwise.

Example:

```

uses ExceptionLog; // The required unit...

var
    Date: TDateTime;
begin
    ...
    if GetCompilationDate(HInstance, True, Date) then

```

```

    MessageBox(0, PChar(FormatDateTime('dd/mm/yyyy hh:nn:ss', Date)), '', 0);
...
end;

```

## 5.18 GenerateHTML function

EurekaLog includes a function called *GenerateHTML*, that enables you to convert a custom plain text in a formatted HTML code (the formatting process uses the HTML layout defined in the EurekaLog options).

**NOTE:** You can use this function to show the EurekaLog error page in [unsupported Web applications types](#)<sup>[76]</sup>.

Syntax of this function is as follows:

```
function GenerateHTML(const Text: string; AddOKButton: Boolean): string;
```

Example:

```

uses ExceptionLog; // The required unit...

var
    HTMLCode: string;
begin
    ...
    HTMLCode := GenerateHTML('Custom text...', True {Add a JavaScript 'back' button at the page bottom} );
    ...
end;

```

## 5.19 SetCustomErrorMessage procedure

EurekaLog includes a procedure called *SetCustomErrorMessage*, that enables you to set a custom error message to display in the Exception Dialog instead of exception message (*only when the dialog is not displayed in detailed mode*).

**NOTE:** Use an empty string to remove any previous custom error message (*Example: SetCustomErrorMessage('');*).

Syntax of this function is as follows:

```
procedure SetCustomErrorMessage(const Value: string);
```

Example:

```

uses ExceptionLog; // The required unit...

begin
    ...
    SetCustomErrorMessage('An error has occurred. ');
    ...
end;

```

## 5.20 SetProxyServer procedure

EurekaLog includes a procedure called *SetProxyServer*, that enables you to set a custom Proxy server. By default EurekaLog use the System Internet Settings to get the Proxy configuration but using this procedure you can force the use of custom Proxy settings.

Syntax of this function is as follows:

```
procedure SetProxyServer(const Server: string; Port: Word);
```

Example:

```
uses EWebTools; // The required unit...
begin
    ...
    SetProxyServer('www.my-proxy.com', 8080);
    ...
end;
```

## 5.21 SetProxyAuthenticationData procedure

EurekaLog includes a procedure called SetProxyAuthenticationData, that enables you to set the Proxy UserID and Password.

Syntax of this function is as follows:

```
procedure SetProxyAuthenticationData(const UserID, Password: string);
```

Example:

```
uses EWebTools; // The required unit...
begin
    ...
    SetProxyAuthenticationData('proxy_username', 'proxy_password');
    ...
end;
```

## 5.22 GetLastExceptionAddress function

EurekaLog includes a function called GetLastExceptionAddress, that enables you to obtain the memory address of the last raised exception.

Syntax of this function is as follows:

```
function GetLastExceptionAddress: Pointer;
```

Example:

```
uses ExceptionLog; // The required unit...
begin
    ...
    WriteLn(Format('Last Exception address: %p', [GetLastExceptionAddress]));
    ...
end;
```

## 5.23 GetLastExceptionObject function

EurekaLog includes a function called GetLastExceptionObject, that enables you to obtain the object instance (*TObject*) of the last raised exception (only if valid too, otherwise *nil*).

Syntax of this function is as follows:

```
function GetLastExceptionObject: TObject;
```

Example:

```
uses ExceptionLog; // The required unit...

begin
    ...
    if (GetLastExceptionObject <> nil) then
        WriteLn(Format('Last Exception class name: "%s"', [GetLastExceptionObject.ClassName]))
    else
        WriteLn('Unknown last exception class name!');
    ...
end;
```

## 5.24 ShowLastExceptionData procedure

EurekaLog includes a procedure called ShowLastExceptionData, that enables you to display the data of the last raised exception.

This procedure is usefully if used in a "re-raised block" (a raise command in an except/end block) without lost the original exception line.

Syntax of this procedure is as follows:

```
procedure ShowLastExceptionData;
```

Example (with raise command):

```
uses ExceptionLog; // The required unit...

begin
    ...
    try
        raise Exception.Create('Original exception!');
    except
        raise; // <-- WARNING - Will be displayed this line as exception line,
               losing the original exception line!!!
    end;
    ...
end;
```

Example (with ShowLastExceptionData procedure):

```
uses ExceptionLog; // The required unit...

begin
    ...
    try
        raise Exception.Create('Original exception!'); // <-- Will be displayed this line as exception line!
    except
        ShowLastExceptionData;
        Abort;
    end;
    ...
end;
```

## 5.25 EurekaLogSendEmail function

EurekaLog includes a function called *EurekaLogSendEmail*, that enables you to send an email to one or more recipients using the default email client installed on your PC.

Syntax of this function is as follows:

```
function EurekaLogSendEmail(const AMailTo, ASubject, ABody, AAttachments: string): Boolean;
```

**Note:**

You can insert more recipients and attachments, separating them with a ';' or a ',' char.

Example:

```
uses ExceptionLog; // The required unit...

var
    Sent: boolean;
begin
    Sent := EurekaLogSendEmail('first@email.com;second@supportemail.com', 'Test email',
        'This is a test email!', 'C:\Log.txt;C:\Windows\notepad.exe');
end;
```

## 5.26 CallStackToStrings procedure

EurekaLog includes a procedure called *CallStackToStrings*, that enables you to obtain a textual representation of a call-stack.

This procedure is usefully to work with all the EurekaLog routines able to return a call-stack type ([TEurekaStackList](#)<sup>[58]</sup>).

Syntax of this procedure is as follows:

```
procedure CallStackToStrings(CallStack: TEurekaStackList[58]; Strings: TStrings);
```

**Note:**

The text output format is full compatible with the EurekaLog log-file (.ELF) call-stack format.

Example:

```
uses ExceptionLog; // The required unit...

var
    CallStackList: TStringList;
begin
    CallStackList := TStringList.Create;
    try
        CallStackToStrings(GetCurrentCallStack, CallStackList);
        // ...
        // Use the CallStackList here...
        // ...
    finally
        CallStackList.Free;
    end;
```

## 5.27 GetCurrentCallStack function

EurekaLog includes a function named GetCurrentCallStack, that enables you to obtain the current call-stack.

Syntax of this function is as follows:

```
function GetCurrentCallStack: TEurekaStackList[58];
```

Example:

```
uses ExceptionLog; // The required unit...

var
  CallStackList: TStringList;
begin
  CallStackList := TStringList.Create;
  try
    CallStackToStrings(GetCurrentCallStack, CallStackList);
    // ...
    // Use the CallStackList here...
    // ...
  finally
    CallStackList.Free;
  end;
end;
```

## 5.28 GetLastExceptionCallStack function

EurekaLog includes a function called GetLastExceptionCallStack, that enables you to obtain the last Exception Call-Stack (it works with handled and unhandled exceptions).

Syntax of this function is as follows:

```
function GetLastExceptionCallStack: TEurekaStackList[58];
```

Example:

```
uses ExceptionLog; // The required unit...

var
  CallStackList: TStringList;
begin
  CallStackList := TStringList.Create;
  try
    CallStackToStrings(GetLastExceptionCallStack, CallStackList);
    // ...
    // Use the CallStackList here...
    // ...
  finally
    CallStackList.Free;
  end;
end;
```

## 5.29 GetCallStackByLevels function

EurekaLog includes a function called GetCallStackByLevels, that enables you to obtain only a custom subset of the current call-stack. You can choose what levels obtains (the level=0 indicate the current running routine, the level=1 the caller of the current routine, ...).

Syntax of this function is as follows:

**function** GetCallStackByLevels(StartLevel, LevelsNumber: Integer): [TEurekaStackList](#)<sup>58</sup>;

Example:

```
uses ExceptionLog; // The required unit...

var
  CallStackList: TStringList;
begin
  CallStackList := TStringList.Create;
  try
    CallStackToStrings(GetCallStackByLevels(0, 99), CallStackList);
    // ...
    // Use the CallStackList here...
    // ...
  finally
    CallStackList.Free;
  end;
end;
```



**Part**

---

**VI**

## 6 TEurekaLog component

### 6.1 How to use this component

EurekaLog install the TEurekaLog component into the EurekaLog component palette.



This component has seven events:

- OnExceptionNotify - a wrapper for the [ExceptionNotify](#)<sup>[29]</sup> EurekaLog event (called when EurekaLog traps an exception)
- OnExceptionActionNotify - a wrapper for the [ExceptionActionNotify](#)<sup>[30]</sup> EurekaLog event.
- OnExceptionErrorNotify - a wrapper for the [ExceptionErrorNotify](#)<sup>[32]</sup> EurekaLog event.
- OnAttachedFilesRequest - a wrapper for the [AttachedFilesRequest](#)<sup>[33]</sup> EurekaLog event
- OnCustomDataRequest - a wrapper for the [CustomDataRequest](#)<sup>[34]</sup> EurekaLog event.
- OnCustomFieldsRequest - a wrapper for the [CustomFieldsRequest](#)<sup>[35]</sup> EurekaLog event.
- OnPasswordRequest - a wrapper for the [PasswordRequest](#)<sup>[36]</sup> EurekaLog event.

#### **Warning:**

You can use this component to simplify management of your EurekaLog events, but it's more reliable use the original events because TEurekaLog component is created only when its parent form is created did not allowing to handled EurekaLog events before its creation.

See the ["Examples"](#)<sup>[53]</sup> topic for further details.

### 6.2 Examples

Some examples:

```
uses ECore, ETypes; // The required units...

// OnExceptionNotify example...
procedure TForm1.EurekaLog1ExceptionNotify(
  EurekaExceptionRecord: TEurekaExceptionRecord; var Handled: Boolean);
begin
  // Don't save the Log File...
  EurekaExceptionRecord.CurrentModuleOptions.SaveLogFile := False;
end;
```

See the ["ExceptionNotify-Examples"](#)<sup>[30]</sup> topic for further details.

```
uses ECore, ETypes; // The required units...

// OnExceptionActionNotify example...
procedure TForm1.EurekaLog1ExceptionActionNotify(
  EurekaExceptionRecord: TEurekaExceptionRecord;
  EurekaAction: TEurekaActionType; var Execute: Boolean);
begin
  // Show exception dialog?
```

```

if EurekaAction = atShowingExceptionInfo then
  begin
    Execute := MessageBox(0, 'Do you want to show error details?', 'Question',
      MB_YESNO or MB_ICONQUESTION or MB_TASKMODAL) <> ID_YES;
  end;
end;

```

See the ["ExceptionActionNotify-Examples"](#)<sup>[33]</sup> topic for further details.

```

uses ECore, ETypes; // The required units...

// OnExceptionErrorNotify example...
procedure TForm1.EurekaLog1ErrorNotify(EurekaExceptionRecord: TEurekaExceptionRecord;
  EurekaAction: TEurekaActionType; var Retry: Boolean);
begin
  // An error occurred when sending the email...
  if EurekaAction = atSentEmail then
    begin
      // Try to send the email via SMTP client.
      EurekaExceptionRecord.CurrentModuleOptions.SMTPHost := 'localhost';
      Retry := True; // Force retry of this action.
    end;
end;

```

See the ["ExceptionErrorNotify-Examples"](#)<sup>[33]</sup> topic for further details.

```

uses ECore, ETypes; // The required units...

// OnAttachedFilesRequest example...
procedure TForm1.EurekaLog1AttachedFilesRequest(AttachedFiles: TStrings);
var
  Path: string;
begin
  // Attach a file chose by user...
  Path := InputBox('Request', 'Insert the path of the file to send', '');
  AttachedFiles.Add(Path);
end;

```

See the ["AttachedFilesRequest-Examples"](#)<sup>[34]</sup> topic for further details.

```

uses ECore, ETypes; // The required units...

// OnCustomDataRequest example...
procedure TForm1.EurekaLog1CustomDataRequest(var CustomData: String);
begin
  // Add custom data to the Log text...
  CustomData := 'My custom data...';
end;

```

See the ["CustomDataRequest-Examples"](#)<sup>[34]</sup> topic for further details.

```

uses ECore, ETypes; // The required units...

// OnCustomFieldsRequest example...
procedure TForm1.EurekaLog1CustomFieldsRequest(FieldsName, FieldsValue: TStrings);
begin
  // Send the software identification ID to the Web server
  FieldsName.Add('Software_ID');
  FieldsValue.Add('F0E0D0');
end;

```

See the ["CustomFieldsRequest-Examples"](#)<sup>[35]</sup> topic for further details.

```
uses ECore, ETypes; // The required units...

// OnPasswordRequest example...
procedure TForm1.EurekaLog1PasswordRequest(var Password: string);
begin
    // Require the password to the user...
    InputQuery('Request', 'Insert the BUG report password', Password);
end;
```

See the ["PasswordRequest-Examples"](#)<sup>[36]</sup> topic for further details.

**Part**

---

**VII**

## 7 Types

### 7.1 TEurekaExceptionRecord type

This one is the structure of TEurekaExceptionRecord:

Unit ExceptionLog.

```
TEurekaExceptionRecord = packed record
  ExceptionObject:      TObject;
  ExceptionAddress:     Pointer;
  ExceptionThreadID:    DWord;
  LogText:              string;
  ModulesList:          TEurekaModulesList[59];
  CallStack:            TEurekaStackList[58];
  CurrentModuleOptions: TEurekaModuleOptions[57];
end;
```

**Note:** if the ExceptionObject is an Exception than is possible change its "Message" property text to show a most friendly error message, but into the log-file will be stored the original Exception Message.

### 7.2 TEurekaModuleOptions type

This is the TEurekaModuleOptions type:

Unit ECore.

```
TEurekaModuleOptions = class(TObject)
...
public
  // Methods...
  constructor Create(ModuleName: string ['']; SaveSharedData: Boolean [False]);
  procedure Assign(Source: TEurekaModuleOptions[57]);
  procedure SetToDefaultOptions;
  procedure SaveToStream(Stream: TStream);
  procedure LoadFromStream(Stream: TStream);
  // Properties...
  property ModuleName: string ...;
  property SMTPFrom: string ...;
  property SMTPHost: string ...;
  property SMTPPort: Word ...;
  property SMTPUserID: string ...;
  property SMTPPassword: string ...;
  property FreezeActivate: Boolean ...;
  property FreezeTimeout: Integer ...;
  property FreezeMessage: string ...;
  property ExceptionClassName: TStringList ...;
  property ExceptionMessage: TStringList ...;
  property CustomizedTexts[Index: TMessageType[60]]: string ...;
  property ActivateLog: Boolean ...;
  property SaveLogFile: Boolean ...;
  property ActivateHandle: Boolean ...;
  property ForegroundTab: TForegroundType[60] ...;
  property AppendLogs: Boolean ...;
  property ShowTerminateBtn: Boolean ...;
  property TerminateBtnOperation: TTerminateBtnOperation[64] ...;
  property LogOptions: TLogOptions[62] ...;
  property ShowOptions: TShowOptions[63] ...;
  property ErrorsNumberToSave: Integer ...;
  property ErrorsNumberToShowTerminateBtn: Integer ...;
  property OutputPath: string ...;
  property EmailAddresses: string ...;
```

```

property EmailSubject:
property EmailMessage:
property EmailSendMode:
property WebSendMode:
property WebURL:
property WebUserID:
property WebPassword:
property WebPort:
property CommonSendOptions:
property AttachedFiles:
property ExceptionDialogOptions:
property AutoCloseDialogSecs:
property SupportURL:
property HTMLLayout:
property AutoCrashOperation:
property AutoCrashNumber:
property AutoCrashMinutes:
property CallStackOptions:
property BehaviourOptions:
end;

```

```

string ...;
string ...;
TEmailSendMode[63] ...;
TWebSendMode[63] ...;
string ...;
string ...;
string ...;
Integer ...;
TCommonSendOptions[61] ...;
string ...;
TExceptionDialogOptions[61] ...;
Integer ...;
string ...;
string ...;
TTerminateBtnOperation[64] ...;
Integer ...;
Integer ...;
TCallStackOptions[62] ...;
TBehaviourOptions[62] ...;

```

### 7.3 TEurekaStackList type

This is the TEurekaStackList type:

Unit ExceptionLog.

```

TEurekaStackList = class(TList)
...
public
...
property Items[Index: Integer]: PEurekaDebugInfo[58] read GetItem; default;
end;

```

### 7.4 TEurekaDebugInfo type

This is the PEurekaDebugInfo type:

Unit ExceptionLog.

```

TEurekaDebugInfo = packed record
DebugDetail: TEurekaDebugDetail[58];
ModuleInfo: PEurekaModuleInfo[59];
ThreadID: DWord;
RunningThread: Boolean;
Addr: DWord;
UnitName: string;
ClassName: string;
ProcedureName: string;
Line: DWord;
ProcOffsetLine: DWord;
end;
PEurekaDebugInfo = ^TEurekaDebugInfo[58];

```

### 7.5 TEurekaDebugDetail type

This is the TEurekaDebugDetail type:

Unit ExceptionLog.

```

TEurekaDebugDetail = (ddNone, ddModule, ddProcedure, ddUnitAndProcedure, ddSourceCode);

```

## 7.6 TEurekaModuleInfo type

This is the TEurekaModuleInfo type:

Unit ExceptionLog.

```
TEurekaModuleInfo = packed record
  Handle:          THandle;
  Name:            string;
  Description:     string;
  Version:         string;
  Size:            DWord;
  FunctionsList:   TEurekaFunctionsList[59];
  ModuleType:      TEurekaModuleType[60];
  ExtraInformation: TEurekaExtraInformation[60];
  LastModified:    TDateTime;
  EncryptPassword: string;
  IsValidEncryptPassword: Boolean;
end;
PEurekaModuleInfo = ^TEurekaModuleInfo[59];
```

## 7.7 TEurekaModulesList type

This is the TEurekaModuleList type:

Unit ExceptionLog.

```
TEurekaModulesList = class(TList)
...
public
...
  property Items[Index: Integer]: PEurekaModuleInfo[59] read GetItem; default;
end;
```

## 7.8 TEurekaFunctionsList type

This is the TEurekaFunctionsList type:

Unit ExceptionLog.

```
TEurekaFunctionsList = class(TList)
...
public
...
  property Items[Index: Integer]: PEurekaFunctionInfo[59] read GetItem; default;
end;
```

## 7.9 TEurekaFunctionInfo type

This is the TEurekaFunctionInfo type:

Unit ExceptionLog.

```
TEurekaFunctionInfo = packed record
  Name: string;
  Addr: DWord;
  Size: DWord;
end;
PEurekaFunctionInfo = ^TEurekaFunctionInfo[59];
```



## 7.10 TEurekaModuleType type

This is the TEurekaModuleType type:

Unit ExceptionLog.

```
TEurekaModuleType = (mtUnknown, mtMainModule, mtBorlandPackage, mtOSLibrary);
```

## 7.11 TEurekaExtraInformation type

This is the TEurekaExtraInformation type:

Unit ExceptionLog.

```
TEurekaExtraInformation = packed record
  EurekaVersion:      Word;
  CompilationDate:    TDateTime;
  Options:            TEurekaModuleOptions57;
  DebugInformation:   TMemoryStream;
end;
```

## 7.12 TEurekaAction type

This is the TEurekaActionType type:

Unit ExceptionLog.

```
TEurekaActionType = (atShowingExceptionInfo, atShowedExceptionInfo, atSavingLogFile,
  atSavedLogFile, atSendingEmail, atSentEmail, atSendingWebMessage,
  atSentWebMessage, atTerminating);
```

## 7.13 TForeground type

This is the TForegroundType type:

Unit ECore.

```
TForegroundType = (ftGeneral, ftCallStack, ftModules, ftCPU, ftCustom);
```

## 7.14 TMessageType type

This is the TMessageType type:

Unit ETypes.

```
TMessageType = (
  // General Messages...
  mtInformationMsgCaption, mtQuestionMsgCaption,
  // Exception Dialog...
  mtDialog_Caption, mtDialog_ErrorMsgCaption,
  mtDialog_GeneralCaption, mtDialog_GeneralHeader,
  mtDialog_CallStackCaption, mtDialog_CallStackHeader,
  mtDialog_ModulesCaption, mtDialog_ModulesHeader,
  mtDialog_CPUCaption, mtDialog_CPUHeader,
  mtDialog_CustomDataCaption, mtDialog_CustomDataHeader,
  mtDialog_OKButtonCaption, mtDialog_TerminateButtonCaption,
  mtDialog_RestartButtonCaption, mtDialog_DetailsButtonCaption,
  mtDialog_SendMessage, mtDialog_ScreenshotMessage, mtDialog_CopyMessage,
  mtDialog_SupportMessage,
  // Log...
  mtLog_AppHeader, // Application...
  mtLog_AppStartDate, mtLog_AppName, mtLog_AppVersionNumber,
  mtLog_AppParameters, mtLog_AppCompilationDate,
```

```

mtLog_ExcHeader, // Exception...
mtLog_ExcDate, mtLog_ExcAddress, mtLog_ExcModule,
mtLog_ExcType, mtLog_ExcMessage,
mtLog_ActCtrlsHeader, // Active Controls...
mtLog_ActCtrlsFormClass, mtLog_ActCtrlsFormText,
mtLog_ActCtrlsControlClass, mtLog_ActCtrlsControlText,
mtLog_CmpHeader, // Computer...
mtLog_CmpName, mtLog_CmpUser, mtLog_CmpTotalMemory, mtLog_CmpFreeMemory,
mtLog_CmpTotalDisk, mtLog_CmpFreeDisk, mtLog_CmpSystemUpTime,
mtLog_CmpProcessor, mtLog_CmpDisplayMode,
mtLog_OSHeader, // Operating System...
mtLog_OSType, mtLog_OSBuildN, mtLog_OSUpdate, mtLog_OSLanguage,
mtLog_NetHeader, // Network...
mtLog_NetIP, mtLog_NetSubmask, mtLog_NetGateway, mtLog_NetDNS1,
mtLog_NetDNS2, mtLog_NetDHCP,
mtLog_CustInfoHeader, // Custom Information...
// Call Stack...
mtCallStack_Address, mtCallStack_Name, mtCallStack_Unit,
mtCallStack_Class, mtCallStack_Procedure, mtCallStack_Line,
mtCallStack_MainThread, mtCallStack_ExceptionThread, mtCallStack_RunningThread,
mtCallStack_CallingThread, mtCallStack_ThreadID, mtCallStack_ThreadPriority,
mtCallStack_ThreadClass,
// Send Dialog...
mtSendDialog_Caption, mtSendDialog_Message, mtSendDialog_Resolving,
mtSendDialog_Connecting, mtSendDialog_Connected, mtSendDialog_Sending,
// Reproduce Dialog...
mtReproduceDialog_Caption, mtReproduceDialog_Request,
mtReproduceDialog_OKButtonCaption,
// Modules List...
mtModules_Handle, mtModules_Name, mtModules_Description,
mtModules_Version, mtModules_Size, mtModules_LastModified, mtModules_Path,
// CPU...
mtCPU_Registers, mtCPU_Stack, mtCPU_MemoryDump,
// Send messages...
mtSend_SuccessMsg, mtSend_FailureMsg);

```

## 7.15 TExceptionDialogOptions type

This is the TExceptionDialogOptions type:

Unit ECore.

```
TExceptionDialogOptions = set of TExceptionDialogOption[61];
```

## 7.16 TExceptionDialogOption type

This is the TExceptionDialogOption type:

Unit ECore.

```

TExceptionDialogOption = (edoShowExceptionDialog, edoSendEmailChecked,
                           edoAttachScreenshotChecked, edoShowCopyToClipOption,
edoShowDetailsButton,
                           edoShowInDetailedMode, edoShowInTopMostMode,
edoUseEurekaLogLookAndFeel);

```

## 7.17 TCommonSendOptions type

This is the TCommonSendOptions type:

Unit ECore.

```
TCommonSendOptions = set of TCommonSendOption[62];
```

## 7.18 TCommonSendOption type

This is the TCommonSendOption type:

Unit ECore.

```
TCommonSendOption = (sndShowSendDialog, sndShowSuccessFailureMsg, sndSendEntireLog,  
                      sndSendXMLLogCopy, sndSendScreenshot, sndUseOnlyActiveWindow,  
                      sndSendLastHTMLPage,  
                      sndSendInSeparatedThread, sndAddDateInFileName, sndCompressAllFiles);
```

## 7.19 TCallStackOptions type

This is the TCallStackOptions type:

Unit ECore.

```
TCallStackOptions = set of TCallStackOption[62];
```

## 7.20 TCallStackOption type

This is the TCallStackOption type:

Unit ECore.

```
TCallStackOption = (csoShowDLLs, csoShowBPLs, csoShowBorlandThreads,  
                   csoShowWindowsThreads, csoShowProcedureOffset);
```

## 7.21 TBehaviourOptions type

This is the TBehaviourOptions type:

Unit ECore.

```
TBehaviourOptions = set of TBehaviourOption[62];
```

## 7.22 TBehaviourOption type

This is the TBehaviourOption type:

Unit ECore.

```
TBehaviourOption = (boActivateCrashDetection, boPauseBorlandThreads,  
                   boDoNotPauseMainThread, boPauseWindowsThreads, boUseMainModuleOptions,  
                   boCopyLogInCaseOfError, boSaveCompressedCopyInCaseOfError);
```

## 7.23 TLogOptions type

This is the TLogOptions type:

Unit ECore.

```
TLogOptions = set of TLogOption[62];
```

## 7.24 TLogOption type

This is the TLogOption type:

Unit ECore.

```
TLogOption = (loNoDuplicateErrors, loAppendReproduceText,
              loDeleteLogAtVersionChange, loAddComputerNameInLogFileName,
              loSaveModulesSection, loSaveCPUSection);
```

## 7.25 TShowOptions type

This is the TShowOptions type:

Unit ECore.

```
TShowOptions = set of TShowOption;
```

## 7.26 TShowOption type

This is the TShowOption type:

Unit ETypes.

```
TShowOption = (
  // Application
  soAppStartDate, soAppName, soAppVersionNumber, soAppParameters, soAppCompilationDate,
  // Exception
  soExcDate, soExcAddress, soExcModule, soExcType, soExcMessage,
  // Active Controls
  soActCtlsFormClass, soActCtlsFormText, soActCtlsControlClass, soActCtlsControlText,
  // Computer
  soCmpName, soCmpUser, soCmpTotalMemory, soCmpFreeMemory, soCmpTotalDisk, soCmpFreeDisk,
  soCmpSysUpTime, soCmpProcessor, soCmpDisplayMode,
  // Operating System
  soOSType, soOSBuildN, soOSUpdate, soOSLanguage,
  // Network
  soNetIP, soNetSubmask, soNetGateway, soNetDNS1, soNetDNS2, soNetDHCP);
```

## 7.27 TEmailSendMode type

This is the TEmailSendOptions type:

Unit ECore.

```
TEmailSendMode = (esmNoSend, esmEmailClient, esmSMTPClient, esmSMTPServer);
```

## 7.28 TWebSendMode

This is the TEmailSendOptions type:

Unit ECore.

```
TWebSendMode = (wsmNoSend, wsmHTTP, wsmHTTPS, wsmFTP);
```

## 7.29 EFrozenApplication type

This is the EFrozenApplication type:

Unit ExceptionLog.

```
EFrozenApplication = class(Exception);
```

### 7.30 TTerminateBtnOperation

This is the TTerminateBtnOperation type:

Unit ECore.

```
TTerminateBtnOperation = (tbTerminate, tbRestart);
```

### 7.31 EEurekaLogGeneralError

This is the EEurekaLogGeneralError type:

Unit ExceptionLog.

```
EEurekaLogGeneralError = class(Exception);
```

### 7.32 TEurekaLogErrorCodes type

This is the TEurekaLogErrorCodes type:

Unit ExceptionLog.

```
TEurekaLogErrorCode = (eeNone, eeShowError, eeLogError, eeEmailMAPIError,  
                        eeEmailShellError, eeEmailSMTPError, eeWebHTTPError,  
                        eeWebHTTPSError, eeWebFTPError);
```

**Part**



## 8 EurekaLog Viewer

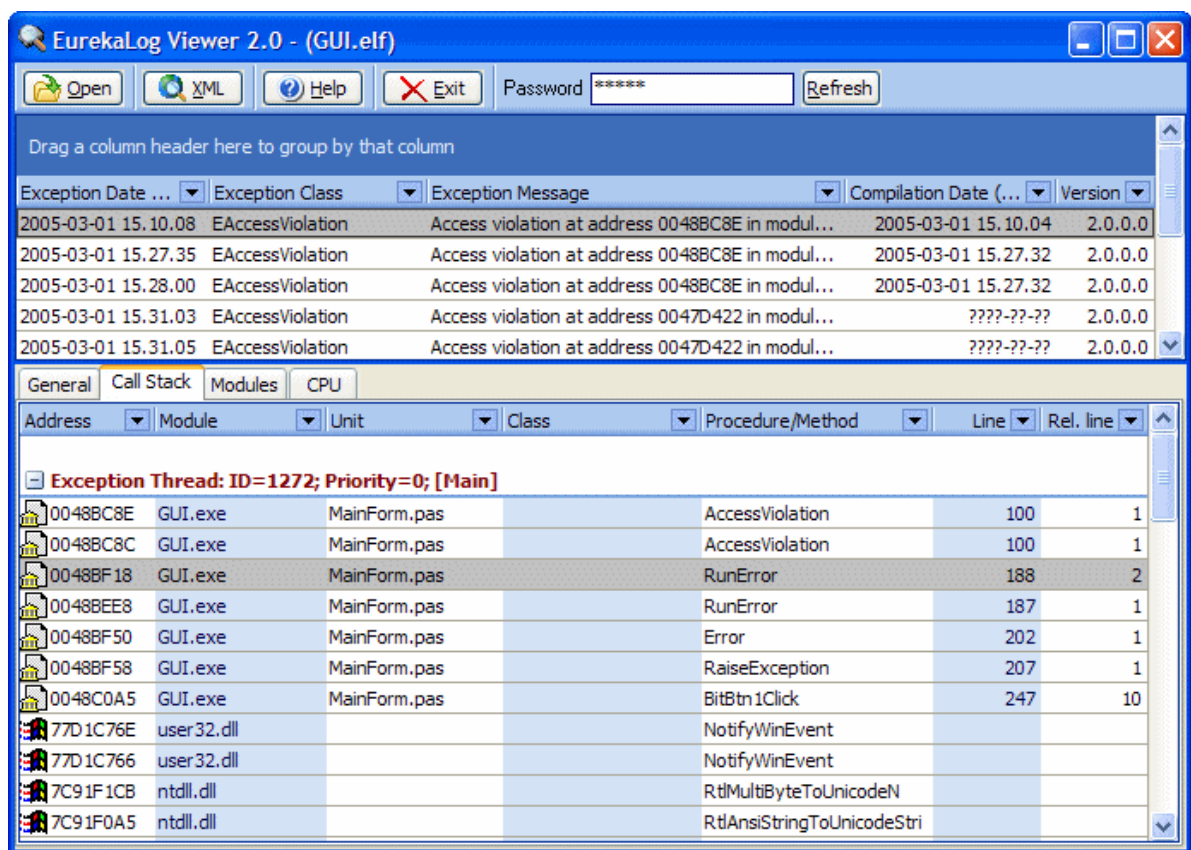
### 8.1 How to use the Viewer

Use this Viewer to analyze the log file generated by EurekaLog (.elf).

You can open this program with a double-click on the Log file (.elf) or using the ["View Exception Log..."](#) new item in the Project IDE menu.

To analyze an encrypted Log file you must enter the password in the relative "Password" field pressing the "Refresh" button.

Use the "XML" button to export the Log file in XML format.



**Part**

---

**IX**



## 9 Compatibility

### 9.1 Old 3.x version

Please read the following instructions if you wish to install EurekaLog 4.x as an upgrade from EurekaLog 3.x:

1. Check the new "Exception Log Options". Some of the old options are no longer available. Save your old options before continuing.
2. Replace the old "ExceptionHandler.OnException" event with the new [ExceptionNotify](#)<sup>[29]</sup> event.
3. Don't use the "TEurekaThread.EurekaHandleException" method for managing thread exceptions. EurekaLog now has fully-automatic support for multithreaded applications.
4. Change all references to the ExceptionLog2 unit to ExceptionLog. The ExceptionLog2 unit was used in the old version for managing Console applications.

That's all.

### 9.2 Changed from the old 4.x version

These are the changed from the old 4.x version to the new 4.5 version:

1. Changed the EmailObject property in [EmailSubject](#)<sup>[57]</sup>
2. Changed the [TEmailSendOptions](#)<sup>[63]</sup> type to "(esoNoSend, esoEmailClient, esoSMTPClient, esoSMTPServer)"
3. Changed the [TLogOption](#)<sup>[62]</sup> type to "(loNoDuplicateErrors, loAppendReproduceText)"
4. Changed the AppendToLog property to [AppendLogs](#)<sup>[57]</sup>
5. Changed the "MuteMode" property to "[ShowExceptionDialog](#)<sup>[57]</sup>" (with inverted sense)
6. Changed the [TForegroundType](#)<sup>[60]</sup>
7. Changed the [TMessageType](#)<sup>[60]</sup> type
8. Changed the [TShowOption](#)<sup>[63]</sup> type
9. EMailSendConfirm property removed
10. EResFile.pas removed

That's all.

### 9.3 Changed from the old 4.5.x version

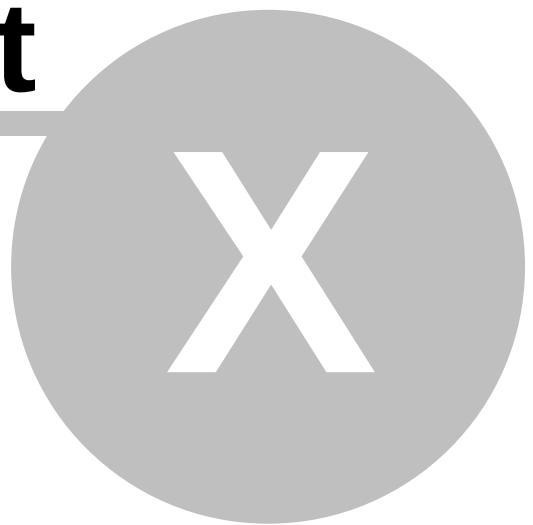
These are the changed from the old 4.5.x version to the new 5.0 version:

1. SMTPShowDialog property removed (replaced by [CommonSendOptions.sndShowSendDialog](#)<sup>[62]</sup>)
2. SendEntireLog property removed (replaced by [CommonSendOptions.sndSendEntireLog](#)<sup>[62]</sup>)
3. EurekaLogLook property removed (replaced by [ExceptionDialogOptions.edoUseEurekaLogStyle](#)<sup>[61]</sup>)
4. ShowExceptionDialog property removed (replaced by [ExceptionDialogOptions.edoShowExceptionDialog](#)<sup>[61]</sup>)
5. EmailSendOptions property renamed in [EmailSendMode](#)<sup>[63]</sup> (with suffix replaced from 'eso' to 'esm')
6. TEurekaExtraInformation.CompiledDate replaced with [TEurekaExtraInformation.CompilationDate](#)<sup>[60]</sup>
7. Added the 'so' prefix at every [TShowOption](#)<sup>[63]</sup> items

That's all.

**Part**

---



## 10 Enterprise version

### 10.1 Debug

If you want to debug EurekaLog source code (only supplied in the Enterprise version), you need to change the *Exceptions.inc* file.

Open the file and comment out the line:

```
{SD-} // Disable Debug-Infos generation (comment it ONLY for debug).
```

Then rebuild your software and run it.

**Note:** When on, debug shows in EurekaLog source codes, into call-stack points, some internal EurekaLog calls.

### 10.2 Recompiling

Enterprise version is delivered with full source code.

If you want to recompile EurekaLog, you must close all open projects and then open the ExceptionExpertX.dpk (or CExceptionExpertX.dpk for CBuilder) package, where X is the Delphi/CBuilder version number, and then recompile the package.

**Part**

---

**XI**

## 11 Command-line compiler

### 11.1 To use the command-line compiler

As a command-line compiler, EurekaLog provides the **ecc32.exe** program for Delphi and **emake.exe** program for CBuilder: you can find them in the same directory as the **dcc32.exe/make.exe** compiler.

This new compiler integrates the standard **dcc32.exe/make.exe** features plus new EurekaLog features, so now you can use just the new **ecc32.exe/emake.exe** command-line compiler instead of the standard **dcc32.exe/make.exe** compiler.

Using the new command-line compiler, you can compile your projects just like when you use the standard compiler, adding to your projects new EurekaLog features.

EurekaLog command line compiler add the following new options:

- **"--el\_config"**, use it to compile your project with a different EurekaLog option file;  
*Example:* `--el_config"new_eurekalog_optionfile"`
- **"--el\_alter\_exe"**, use it not to compile your project but only to add the EurekaLog debug info into a just compiled project;  
*Delphi example:* `--el_alter_exe"ProjectFile.dpr"`  
*C++Builder example:* `--el_alter_exe"MakeFile.mak" or --el_alter_exe"ProjectFile.bpr"`

Don't use the standard command-line compiler anymore: use only the new EurekaLog command-line compiler.

Examples:

```
ecc32.exe "ProjectFile.dpr" // Delphi command-line compiler
emake.exe -B -f"MakeFile.mak" // C++Builder command-line compiler
```

Note: the **emake.exe** command-line compiler did not support ".bpr" files (supported only ".mak" file). To generate the ".mak" file from the ".bpr" you can use the **bpr2mak.exe** command-line program.

**Part**

---

**XIII**

## 12 Support

### 12.1 FAQ

**Question:** How does EurekaLog work?

**Answer:** EurekaLog perfectly integrates itself with the Delphi/CBuilder IDE and builds (with the normal compile command) your applications (GUI, Console, Web, etc.) with the new capability to intercept every exception and trace a detailed and customizable log of every code point executed between the application's start and the point where an exception is raised (showing address, module, unit, class, method, line number and much more).

**Question:** How much does EurekaLog increase compiled file size?

**Answer:** EurekaLog increases the compiled file size by about 50 Kbyte plus a variable percentage that ranges from 0.5% to a maximum of 4%.

**Question:** Can EurekaLog decrease an application's performance?

**Answer:** Definitely NOT, because it works only when exceptions are raised.

**Question:** Why does compiled file size increase by more than 4%? (Happens sometimes, but very rarely).

**Answer:** When you set the "Project/Options/Debugging/Use Debug DCUs" project option, Delphi includes all VCL source-code data in the EurekaLog debug information. This increases the final size of compiled file.

**Question:** Is EurekaLog compatible with exe compression/protection software?

**Answer:** Yes, EurekaLog is FULLY compatible with exe compression/protection software, so you can use those tools without any restrictions.

**Question:** Does EurekaLog work with DLLs and BPLs?

**Answer:** Yes it does, but only in Professional and Enterprise versions.

**Question:** What does EurekaLog do when it intercepts an error ?

**Answer:** It displays a window to communicate the error in a graphic (GUI) application, it shows a text in a Console application and shows an HTML document in a Web application.

**Question:** What is the log file's extension?

**Answer:** It is \*.ELF (EurekaLog File).

**Question:** Does EurekaLog intercept every type of exception-based error?

**Answer:** Yes it does. EurekaLog is able to intercept every type of exception-based error, Thread, Console and Internet Web exceptions included.

**Question:** How many errors can EurekaLog store in the file log ?

**Answer:** There are no limits.

**Question:** When there are errors already verified and saved in the log file, is it possible to avoid further savings?

**Answer:** Yes it is. You must set the "Do not include duplicate errors" option in the "Exceptions Log Options..." menu.

### 12.2 On-line support

Feel free to visit <http://www.eurekalog.com/support.php> or send an e-mail to [support@eurekalog.com](mailto:support@eurekalog.com)

**Part**





## 13 Unsupported applications

EurekaLog support natively more applications types (see ["features"](#)<sup>[4]</sup> topic for further details), but not all exists applications types, so the scope of this little tutorial is explain how to can use EurekaLog in unsupported applications types.

Basically the steps to do are two:

- create a OnException event for the application type (*used to call EurekaLog manually*);
- create a [ExceptionNotify](#)<sup>[29]</sup> event for EurekaLog (*used to customize the EurekaLog behavior*).

**NOTE:** handling Web application you must disable the ["Show Exception Dialog"](#)<sup>[12]</sup> option to inhibit the Exception Dialog showing.

So a little example (*to manage unsupported Web modules*) can become as follow:

```
// Event uses to call manually EurekaLog...
procedure TWebModule1.WebModuleException(Sender: TObject; E: Exception; var Handled: Boolean);
begin
    Handled := True;
    StandardEurekaNotify(E, ExceptAddr); // Call manually EurekaLog
end;

// Event used to send to Browser the EurekaLog error page...
procedure TWebModule1.EurekaLog1ExceptionNotify(EurekaExceptionRecord:
TEurekaExceptionRecord; var Handled: Boolean);
begin
    Response.Content := GenerateHTML(
        EurekaExceptionRecord.LogText, // Exception Log plain text
        True);                        // Add the JavaScript 'back' button at the page bottom
end;
```

# Index

## - A -

Advanced Tab 14  
 AttachedFilesRequest - Examples 34  
 AttachedFilesRequest - How to use this event 33

## - C -

CallStackToStrings procedure 49  
 Changed from the old 4.5.x version 68  
 Changed from the old 4.x version 68  
 Console 23  
 CurrentEurekaLogOptions function 39  
 CustomDataRequest - Examples 34  
 CustomDataRequest - How to use this event 34  
 CustomFieldsRequest - Examples 35  
 CustomFieldsRequest - How to use this event 35

## - D -

Debug 70  
 Design-Time 26

## - E -

EEurekaLogGeneralError type 64  
 EFrozenApplication type 63  
 EurekaLog version constants 43  
 EurekaLog Viewer 66  
 EurekaLogSendEmail function 49  
 Exception Dialog Tab 12  
 ExceptionActionNotify - Examples 31  
 ExceptionActionNotify - How to use this event 30  
 ExceptionErrorNotify - Examples 33  
 ExceptionErrorNotify - How to use this event 32  
 ExceptionNotify - Examples 30  
 ExceptionNotify - How to use this event 29  
 Exceptions Tab 17

## - F -

FAQ 74

Features 4  
 ForceApplicationTermination function 40

## - G -

GenerateHTML function 46  
 GetCallStackByLevels function 50  
 GetCompilationDate function 45  
 GetCurrentCallStack function 50  
 GetEurekaLogModuleVersion function 45  
 GetLastEurekaLogErrorCode function 41  
 GetLastEurekaLogErrorMsg function 41  
 GetLastExceptionAddress function 47  
 GetLastExceptionCallStack function 50  
 GetLastExceptionObject function 47  
 GUI 19

## - H -

How to use EurekaLog 3

## - I -

Installation 3  
 IsEurekaLogActive function 43  
 IsEurekaLogActiveInThread function 42  
 IsEurekaLogInstalled function 39  
 IsEurekaLogModule function 44

## - L -

Log File Tab 11

## - M -

Messages Tab 16

## - N -

New menu option 8

## - O -

Old 3.x version 68  
 On-line support 74

## - P -

PasswordRequest - Examples 36  
PasswordRequest - How to use this event 36

## - R -

Recompiling 70

## - S -

SaveScreenshot procedure 43  
SaveScreenshotToStream procedure 44  
Send Tab 9  
SetCustomErrorMessage procedure 46  
SetEurekaLogInThread procedure 42  
SetEurekaLogState procedure 42  
SetProxyAuthenticationData procedure 47  
SetProxyServer procedure 46  
ShowLastExceptionData procedure 48  
StandardEurekaError function 40  
StandardEurekaNotify function 39

## - T -

TBehaviourOption type 62  
TBehaviourOptions type 62  
TCallStackOption type 62  
TCallStackOptions type 62  
TCommonSendOption type 62  
TCommonSendOptions type 61  
TEmailSendMode type 63  
TEurekaAction type 60  
TEurekaDebugDetail type 58  
TEurekaDebugInfo type 58  
TEurekaExceptionRecord type 57  
TEurekaExtraInformation type 60  
TEurekaFunctionInfo type 59  
TEurekaFunctionsList type 59  
TEurekaLog 53  
TEurekaLog - Examples 53  
TEurekaLogErrorCode type 64  
TEurekaModuleInfo type 59  
TEurekaModuleOptions type 57  
TEurekaModuleType type 60  
TEurekaStackList type 58

TExceptionDialogOption type 61  
TExceptionDialogOptions type 61  
TForeground type 60  
TLogOption type 62  
TLogOptions type 62  
TMessageType type 60  
To use the command-line compiler 72  
TShowOption type 63  
TShowOptions type 63  
TTerminateBtnOperation type 64  
TWebSendMode type 63

## - U -

Unsupported applications 76

## - W -

Web 25  
What is EurekaLog 2