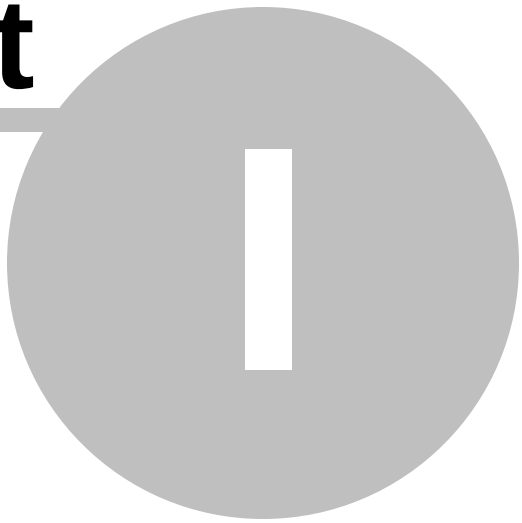
EurekaLog.NET 6.5.x

user manual

Table of Contents

Part



1 Introduction

1.1 What is EurekaLog.NET

EurekaLog.NET is the new add-in tool that gives your application (GUI, Console, Web, etc.) the ability to catch all exceptions, 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.NET is easy to use because it is fully integrated into the Visual Studio IDE. You just rebuild your application to add this new exception-logging capability.

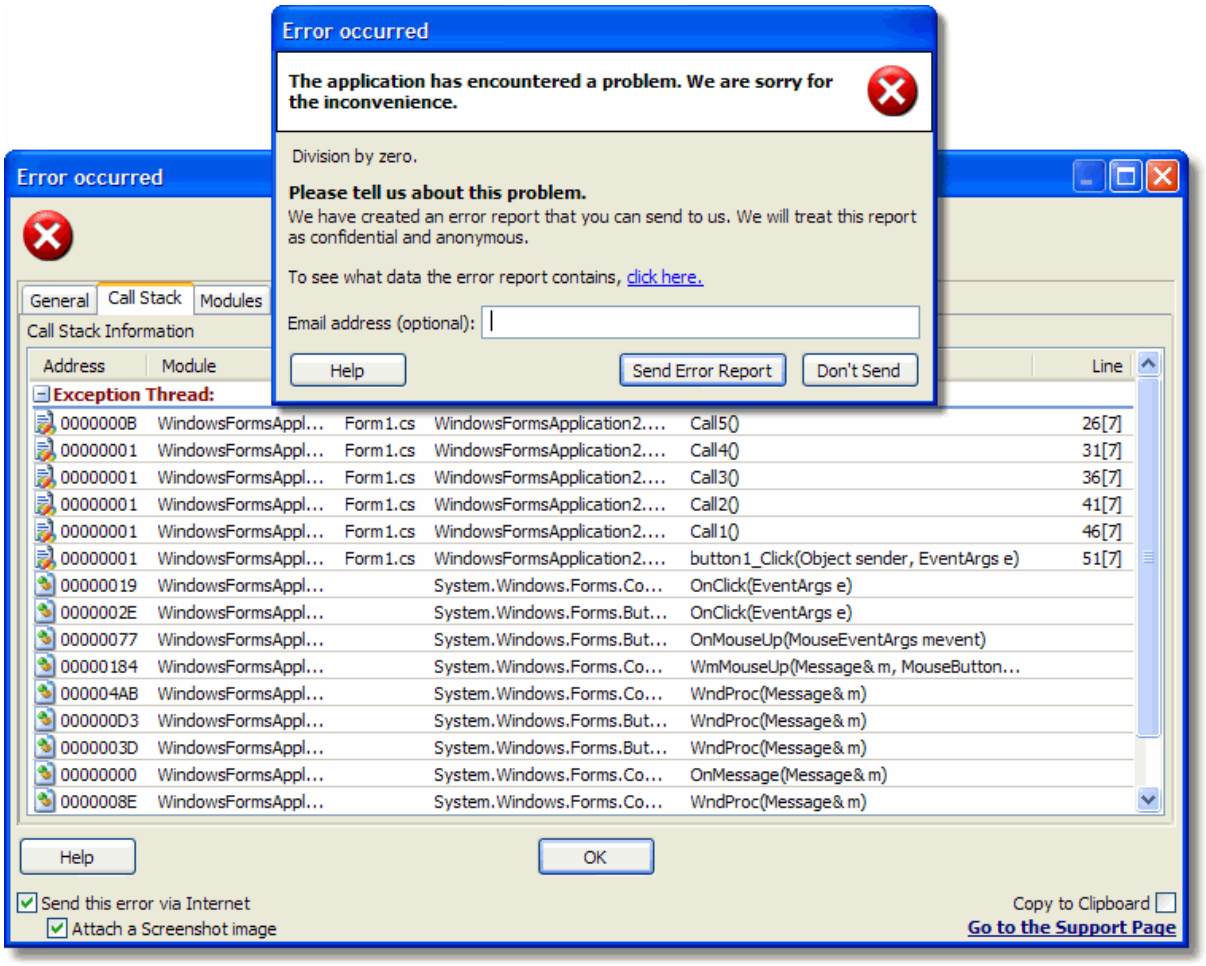
EurekaLog.NET does not affect the performance of your application, as it only executes when an exception is raised. To works EurekaLog.NET needs only of the compiled file and optionally the relative .PDB (to obtain a stack trace with the source line numbers).

EurekaLog.NET is compatible with Visual Studio 2005 and 2010.

It comes with full money back guarantee, is royalty free, and freely updatable!

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

See "features" 5 topic for further details.
See "How to use EurekaLog.NET 3" topic for a quick start guide



A typical example of EurekaLog.NET dialog.

1.2 Installation

Before installing EurekaLog.NET, you must close all Visual Studio instances.

Run the EurekaLog.NET installation program and select one or more Visual Studio versions you wish to install.

Start the installation.

When installation ends, open the Visual Studio IDE and you are ready to work with EurekaLog.NET.

Note: EurekaLog.NET install the new EurekaLog.NET 41 component into the EurekaLog.NET component palette.

1.3 How to use EurekaLog.NET

You must recompile your Visual Studio projects in order to use EurekaLog.NET with them.

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

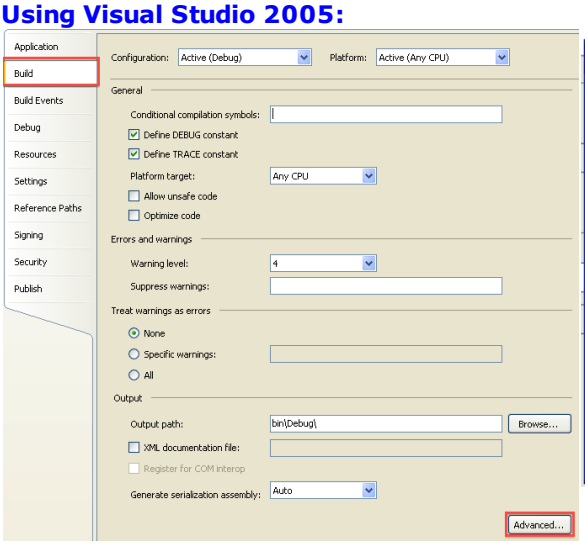
Full debug info

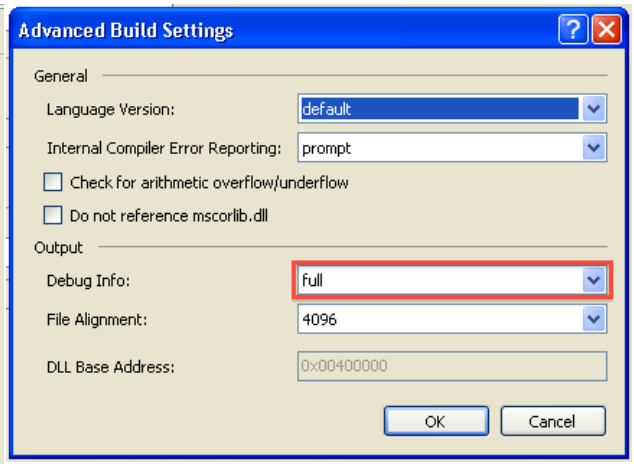
There is one exception to this claim. Namely, in order to be able to display line numbers in the exceptions' call stacks, EurekaLog.NET needs complete debug information.

In order to enable this you need to do two things:

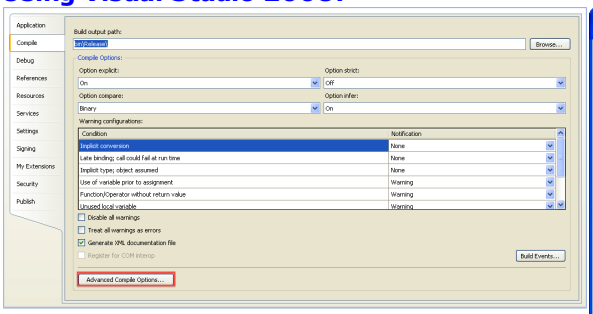
- Enable full debug info for the project. You can accomplish this in the build settings for the project. Click on the "Advanced" button, and set "Debug Info" to "full".

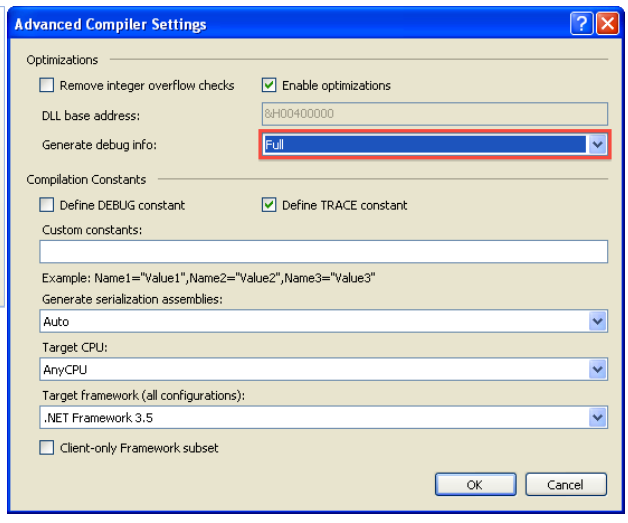
Using Visual Studio 2005:



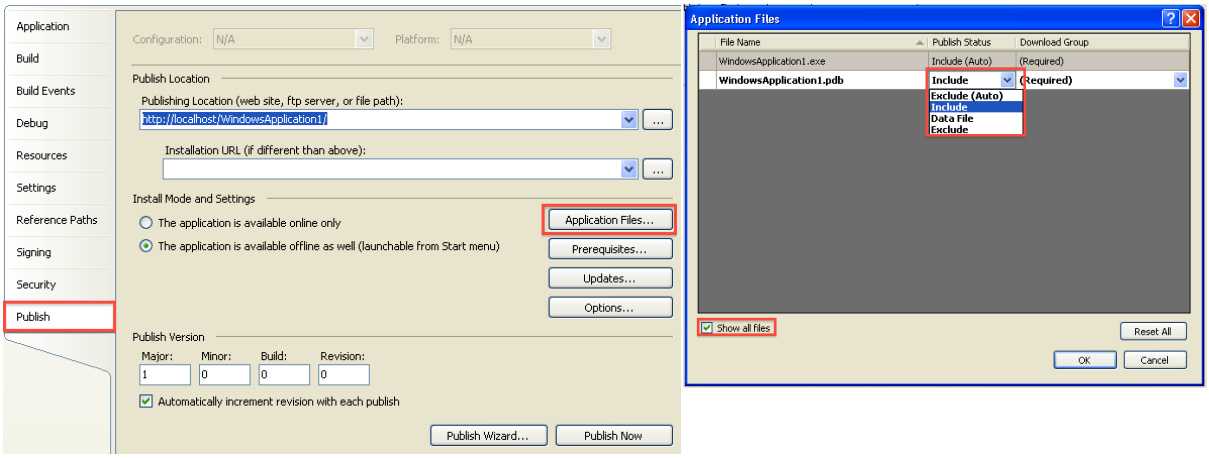


Using Visual Studio 2008:





- Include the project's pdb file with your deployment. Easiest way to do this is by using the "Publish" tab in the project's properties. Click on the "Application Files" button, then check "Show all files" and set the pdb files' publish status to "Include" and download group to "Required". You do not need EurekaLog.NET's debug info (EurekaLog.pdb). Of course, this is only one way to do this. You can, also, add those files manually, or however your distributing system allows you to.



Visual Studio 2010 Client Profile

Starting with Visual Studio 2010, Microsoft introduced .NET Client Profiles for .NET 3.5 and 4.0. The client profiles represent a subset of the full .NET framework with what Microsoft considers most commonly used libraries.
EurekaLog.NET needs some of the libraries that are excluded in the Client Profile subset. Hence, EurekaLog.NET does **not** support the usage of the Client Profiles and requires the full .NET framework, or the .NET Compact Framework for Windows Mobile projects..

To access to your project EurekaLog.NET options, use the new "EurekaLog/Options..." menu item (see here 9).

1.4 Features

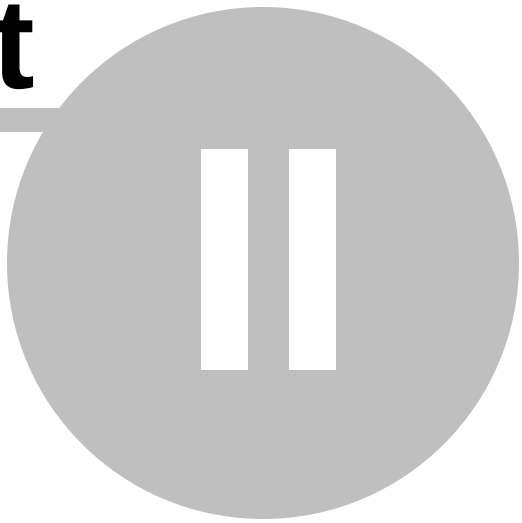
EurekaLog.NET contains all the features that you need in a bug resolution system:

Common features	
Supported IDE & languages	
Visual Studio versions	2005-2010
Visual C#	✔
Visual Basic .NET	✔
Delphi Prism	✔
Compiled file & Debug data	
Decrease in Application's performance	none
IDE & Compiler	
Full integration with Visual Studio IDE	✔
EurekaLog Viewer full integrated with the Visual Studio IDE	✔
The new EurekaLog Viewer is more similar to a stand alone BUG tracking tool	✔

Full revisited EurekaLog.NET Options form	✔
Documentation	
CHM Help file	✔
Printable PDF manual	✔
On-line HTML manual	✔
New application capabilities	
Catches of every **EXCEPTION!!!**	✔
Display the Processes/Modules list sections	✔
Display of more Hardware and Software info *(DPI, printer, VGA, privileges, ...)*	✔
Full customizables Exceptions Filters *(can choose style, behavior, messages ...)*	✔
Environment variables (%EnvironmentVariable%) support	✔
Creation of a detailed Log for every exception *(module, unit, class, method, line ...)*	✔
Termination/restarting after a customizable number of exceptions	✔
Fully customizable exception management with new EurekaLog.NET events	✔
Debug of third-party DLLs	✔
Exception dialogs & HTML error page	
Full customizable Exception-Dialog *(with more new styles - as MS style)*	✔
Add a customizable HELP button *(call an event)*	✔
Full UNICODE logs handling	✔
Fully customizable message texts collections *(for multi-language applications)*	✔
Adding customizable "Support Link" in the Exception-Dialog	✔
Customizable HTML error page via HTML template	✔
Automatic Exception-Dialog closing after a customizable time	✔
Jump from Exception Dialog line to source code line *(with a simple double-click)*	✔
Customizable log-file section view *(show/hide Modules-Processes)*	✔
Send messages & Files	
Delivery of every new BUG to the most used Web BUG-Tracking tools	✔
Compress and encrypt all files to send in in ZIP format	✔
Delivery of a customizable email 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	✔
Upload of log-file and attached files via HTTP/HTTPS and FTP protocol	✔
Send a log-file copy in XML format	✔
Send the email/"upload files" in a separated thread	✔
Attach customizable files to the send message	✔
Add customizables data to the log-file	✔

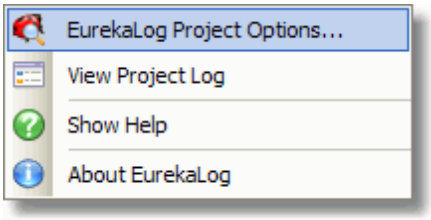
Add customizables fields to the uploading HTML page	✔
Type of supported applications	
ASP.NET	✔
WinForms	✔
Console	✔
WPF Applications	✔
WPF Browser Applications	✔
Other	
New EurekaLog Viewer full integrated with the Windows Shell	✔

Part



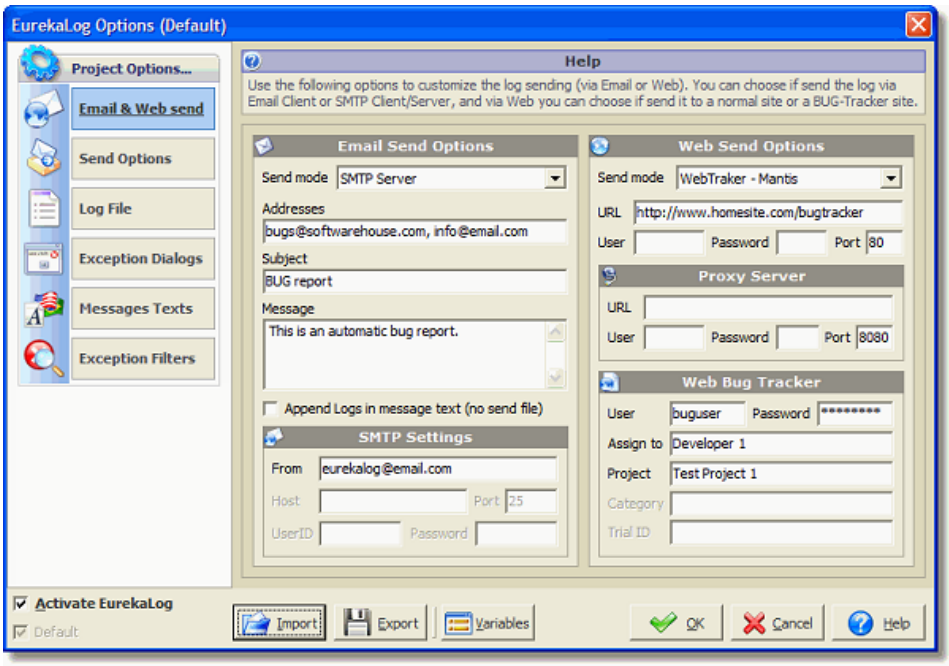
2 New options

2.1 New menu options



New options in IDE's EurekaLog.NET menu, added by EurekaLog.NET: click on "EurekaLog Project Options..." to access EurekaLog.NET options.

2.2 Email & Web Send Tab



Here you may tell the program to automatically send an email (multiple addresses are allowed) a Web message or a bug report to a Web BUG-Tracking tool (like BugZilla, Mantis or FogBugz) whenever an exception is raised.

Email Send Options:

You can choose to send the email via *default email client*, via *SMTP client* (with or without SSL support) or via *internal SMTP server*.

If you select the *SMTP client/SMTP-SSL 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 (to a generic web site):

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 you can use the "Proxy Server" box.

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...
    }
  }
?>
```

Note: Make sure, the assigned URL points to the script directly, not the web page's URL. For example, http://bugs.com/upload.php would produce the desired effect, while, only http://bugs.com won't work.

Web Send Options (to a BUG-Tracking tool):

Activating this options you may tell to the program to send the Log-File with all the other attached files to a Web BUG-Tracking tool (like BugZilla, Mantis or FogBugz).

You can send the bug report via *HTTP or HTTPS* (secure HTTP).

To set all the Web BUG-Tracking parameter use the "Web BUG Tracker" box.

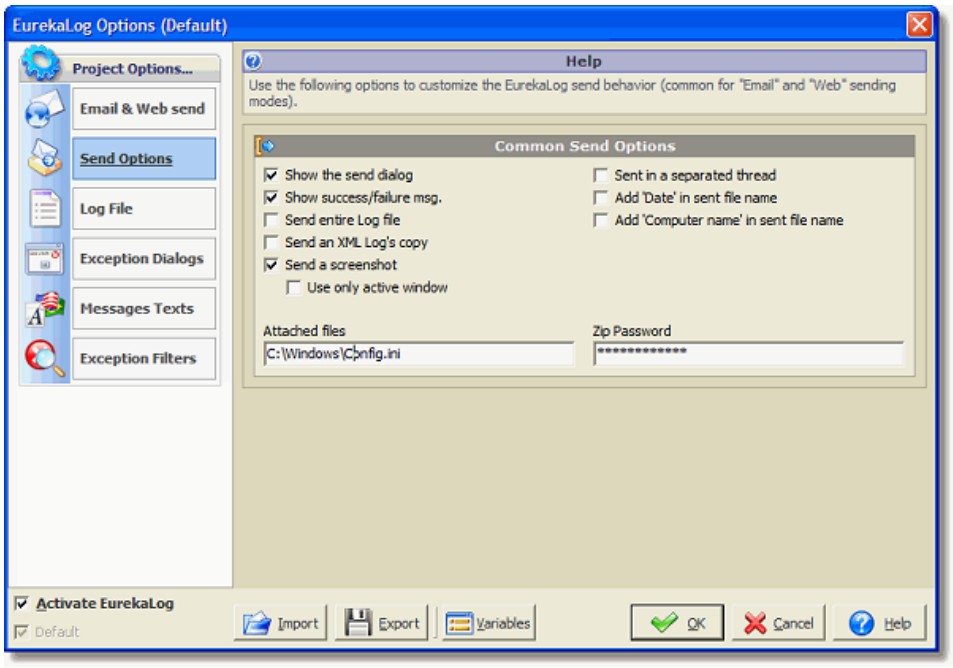
New Events:

To send to a Web HTML page a specified fields list, it's available the new CustomFieldsRequest 33 event.

To add some custom data to the Log text it's available the new CustomDataRequest 32 event.

To send custom attached files with the message (Email and Web) it's available the new AttachedFilesRequest 32 event.

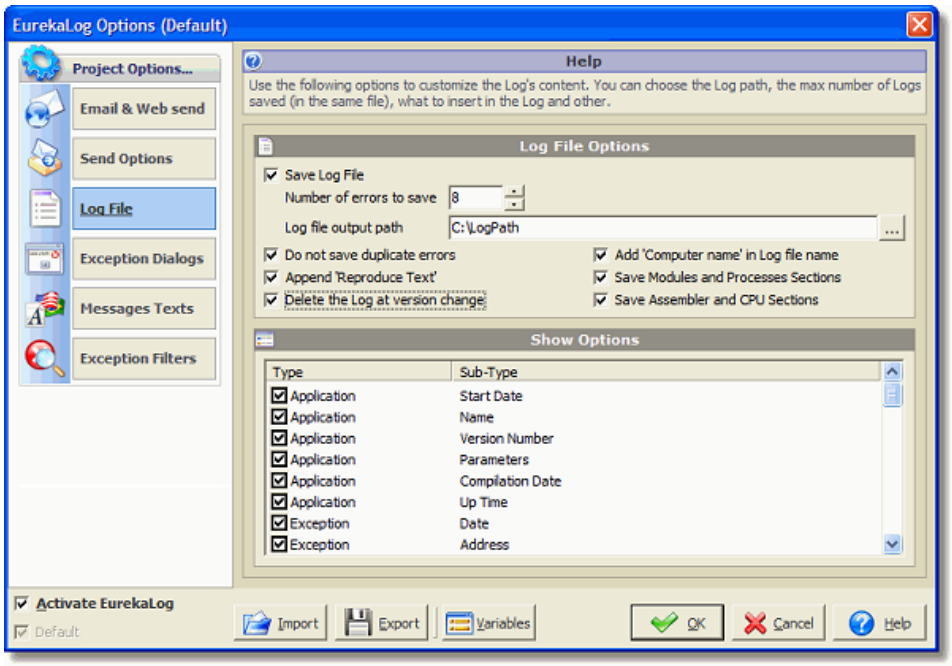
2.3 Send Options Tab



Common Send Options:

"Show the send dialog"	Show the send dialog
"Show success/failure msg."	Show the success 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
"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 sent file name"	Add the current date-time *(in ISO format)* to the sent file name
"Add 'Computer name' in sent file name"	Add the Computer name to the sent file name
"Attached files"	A list of files path to attach to the sending message
"Zip Password"	An optionally password used to encrypt the sent ZIP file

2.4 Log File Tab



In this section you may customize the log file.

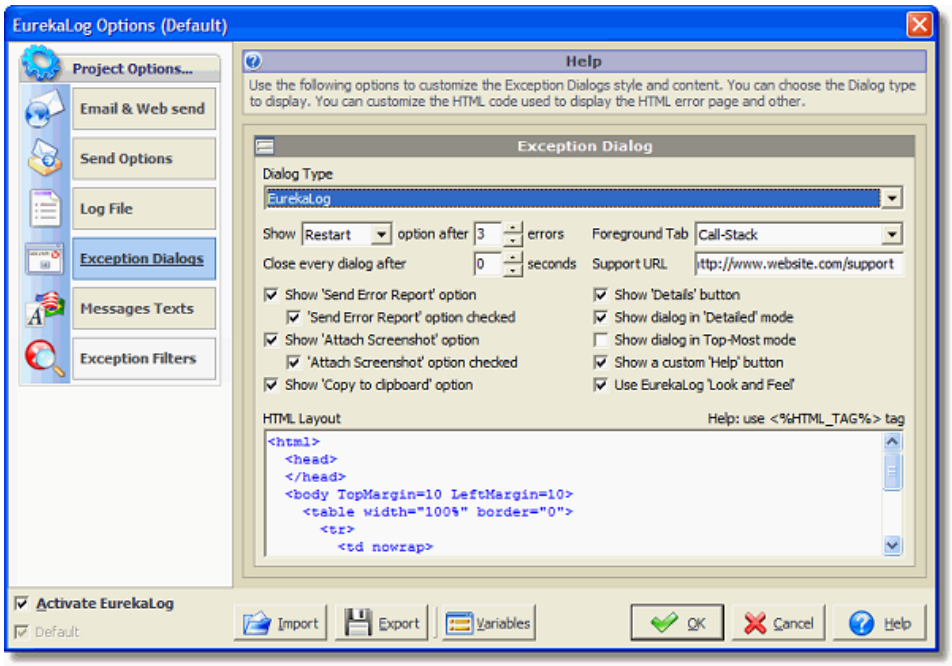
Saving Options:

"Save Log File"	Save the log file into the "Output path" folder
"Number of errors to save"	Max number of logged exceptions to save (min=1)
"Output Path"	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 "% AppData%/EurekaLog for Visual Studio / ProjectName" folder)
"Do not save duplicate errors"	Inhibit logging exceptions that have already been logged to the log file *(Inhibiting it's send too)*
"Append 'Reproduce Text'"	Append to the Log File a text, entered by the user of your program, describing How to reproduce the error
"Delete the Log at version change"	Delete the Log file when the program version it's changed
"Add 'Computer name' in Log file..."	Add the Computer-Name to the Log file name
"Save Modules and Processes Sections"	Save the Modules and Processes sections in the Log File *(displaying it in the Exception Dialog)*
"Save Assembler and CPU Sections"	Save the Assembler and CPU sections 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 log-file.

2.5 Exception Dialogs Tab



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

Dialog Type:

You can choose the Exception Dialog from:

- none
- MessageBox
- MS Classis
- EurekaLog Dialog

Terminate/Restart option:

You can choose to Terminate or Restart the application after a specified number of exceptions.

Foreground option:

You can indicate which Tab, General, Call-Stack, Modules, Processes, Assembler or CPU, should be presented in the foreground when an exception is presented on screen.

Close Dialog option:

You can choose to close every Exception Dialog after a specified number of seconds that the dialog is unused.

Support URL option:

You can set this option to display in the Exception Dialog a specified Internet link (URL).

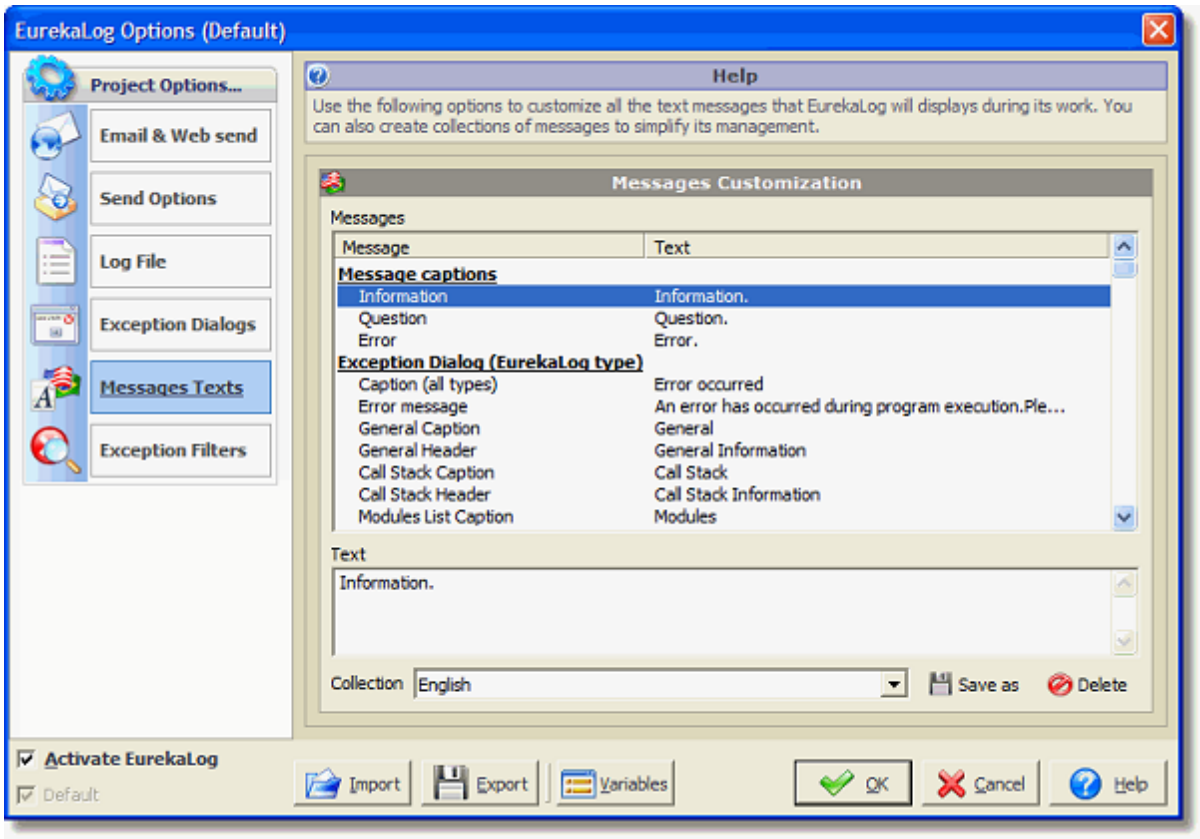
Common Options:

"Show 'Send Error Report' option"	Show the 'Send Error Report' option
"'Send Error Report' option checked"	Check the 'Send Error Report' option
"Show 'Attach Screenshot' option"	Show the 'Attach Screenshot' option
"'Attach Screenshot' option checked"	Check the 'Attach Screenshot' option
"Show 'Copy to clipboard' option"	Show the 'Copy to clipboard' option
"Show 'Details' button"	Show the 'Detail' button *(used to diaplay all the detailed exception data)*
"Show dialog in 'Detailed' mode"	Show the Exception-Dialog as the customer has pressed the 'Detail' button
"Show dialog in Top-Most mode"	Show the Exception-Dialog in Top-Most mode *(in Top to all displayed windows)*
"Show a custom 'Help' button	Show a customizable 'Help' button in the main Exception Dialog (you can customize its text via "Messages text Tab 15" and its behavior via the "CustomButtonClickNotify 34" event)
"Use EurekaLog 'Look and Feel'"	Display the Exception-Dialog using it's personal 'Look and Feel' *(ignoring the standard Windows 'Look and Feel')*

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.

2.6 Messages Texts Tab



Here you can customize the texts of EurekaLog.NET so you can use EurekaLog.NET in your favorite 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.

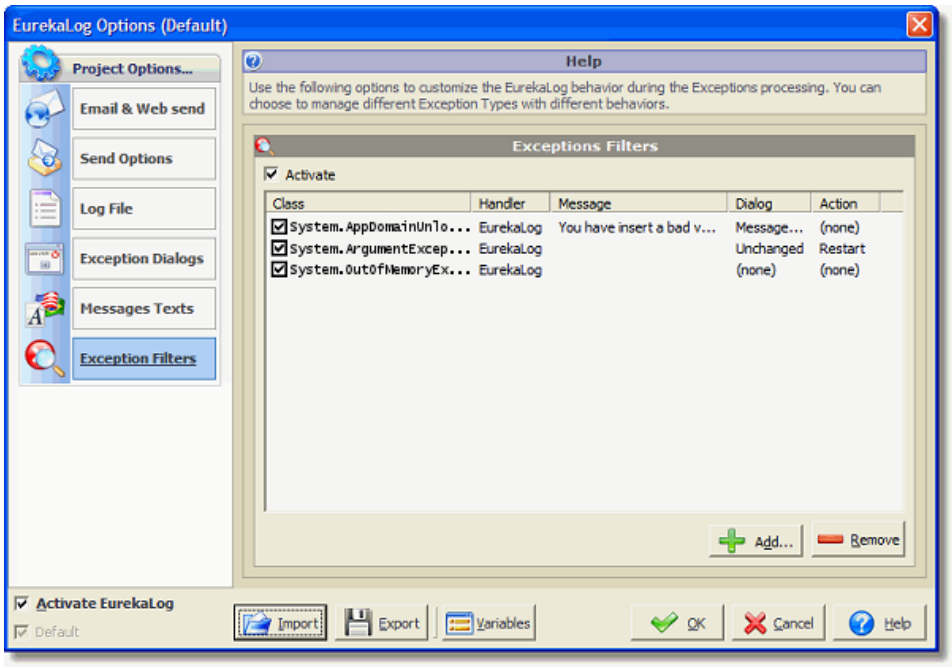
You can save every your "Messages Text Collection" using the "Save as" button.

You can delete a "Messages Text Collection" using the "Delete" button.

You can load a saved "Message Text Collection" simply selecting it from the drop-down list.

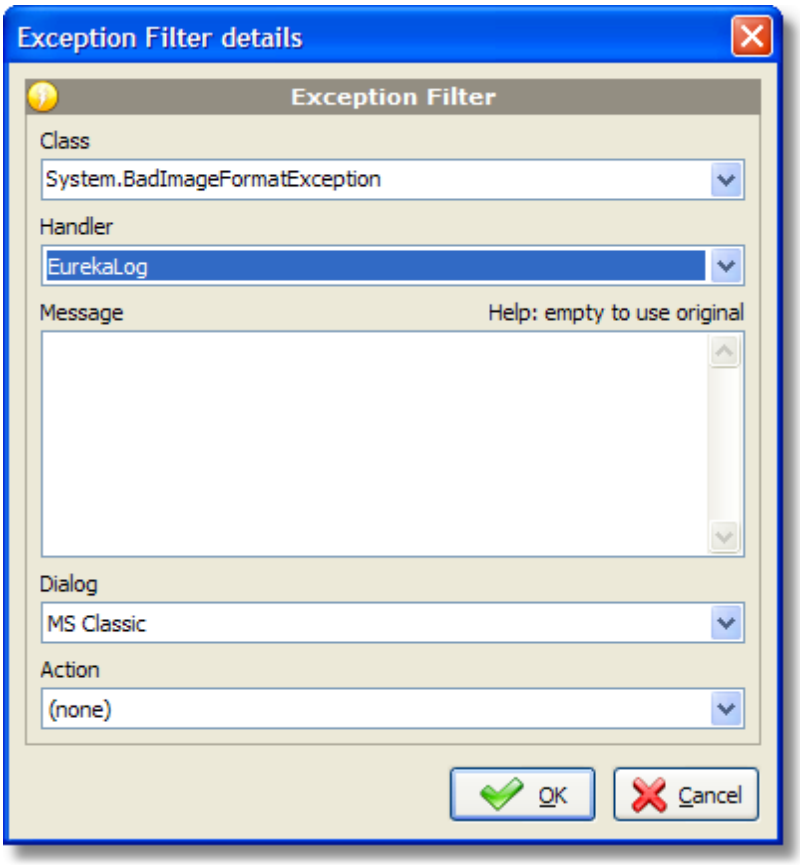
Using the previous options you can also create and save more different EurekaLog.NET translations.

2.7 Exception Filters Tab



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

When an exception on this list is intercepted, it will not be processed in a normal way but following the specified Exception Filter setting.



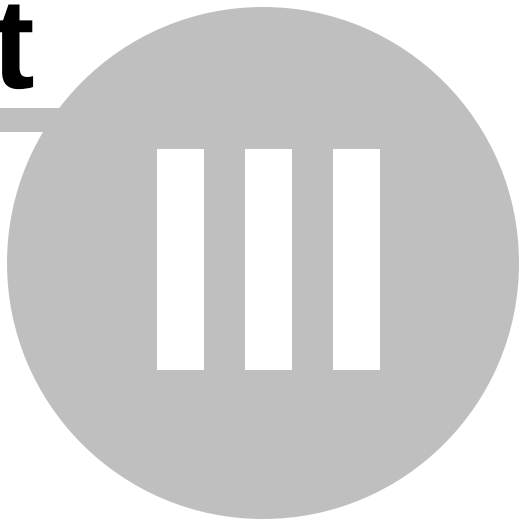
Is possible choose:

- the Filter's Exception Type
- the Exception Handler (none = the exception will be ignored, EurekaLog = the exception will be processed by EurekaLog.NET)
- The Message to display (empty to use the original Exception message)
- the Exception Dialog to use (none, Unchanged, MessageBox, MS Classic, EurekaLog)
- the Action to execute (none, Terminate, Restart).

Use the "Add" key to add a new Exception Filter. Use the "Remove" key to remove the selected Exception Filter.

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

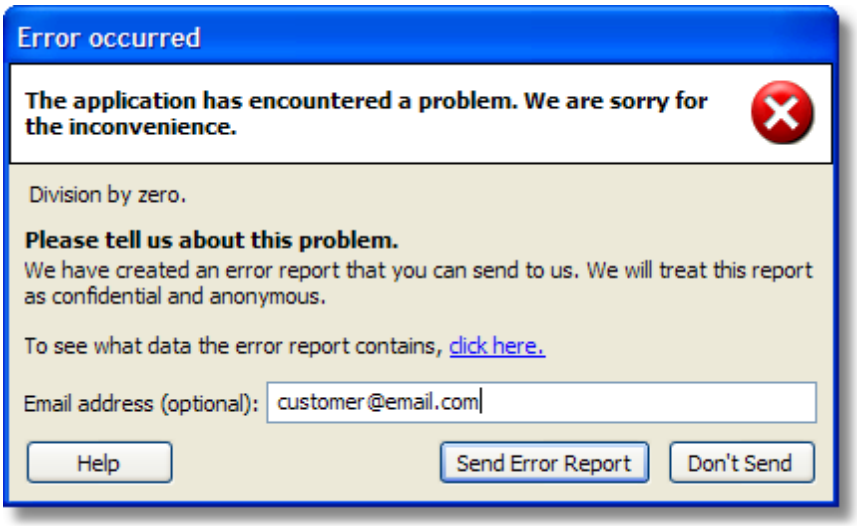
Part



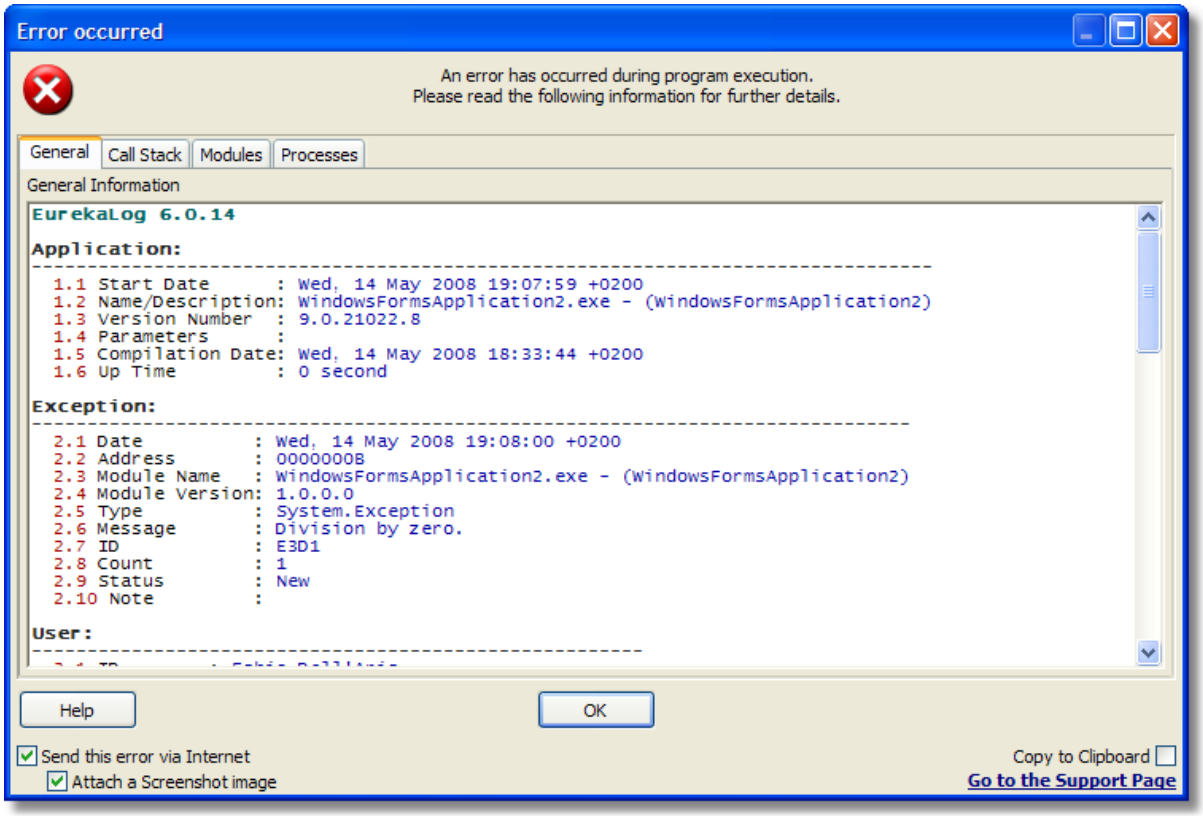
3 Types of Application

3.1 GUI

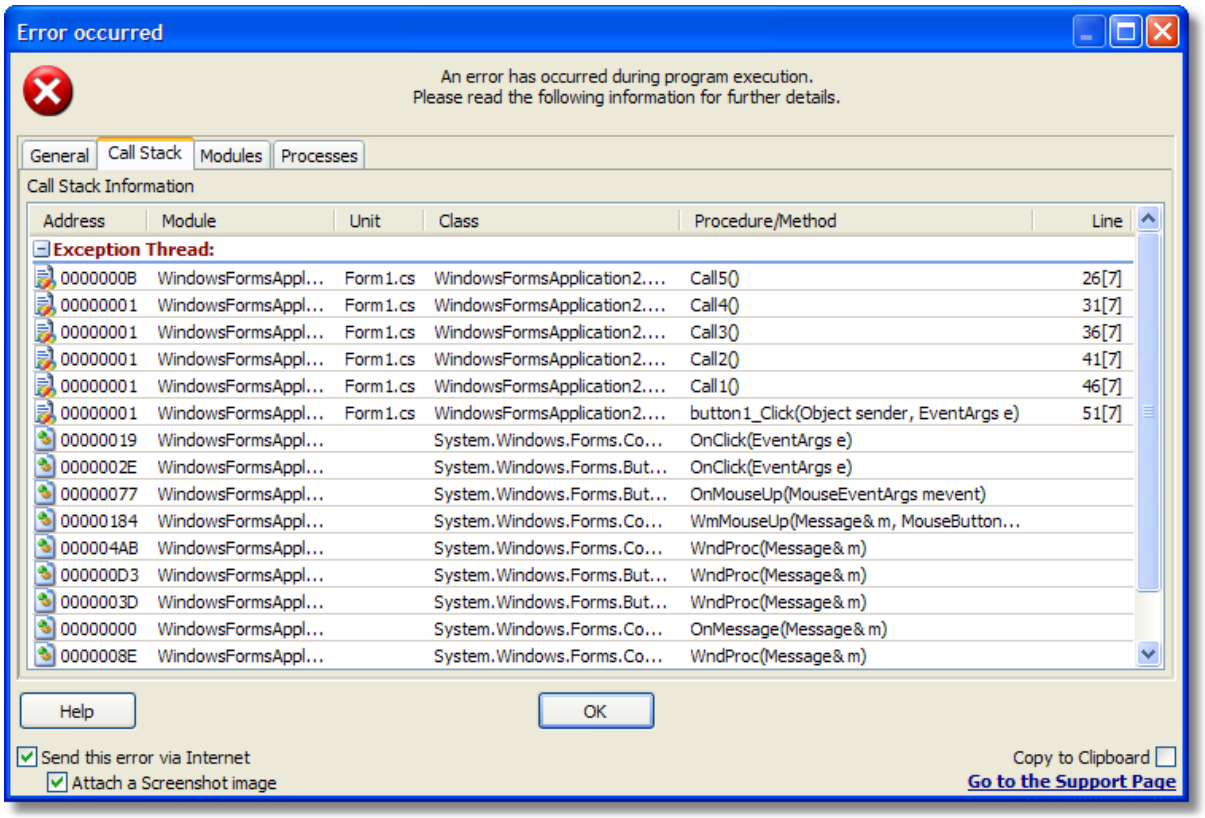
Information displayed when EurekaLog.NET runs in a GUI (standard graphical) application:



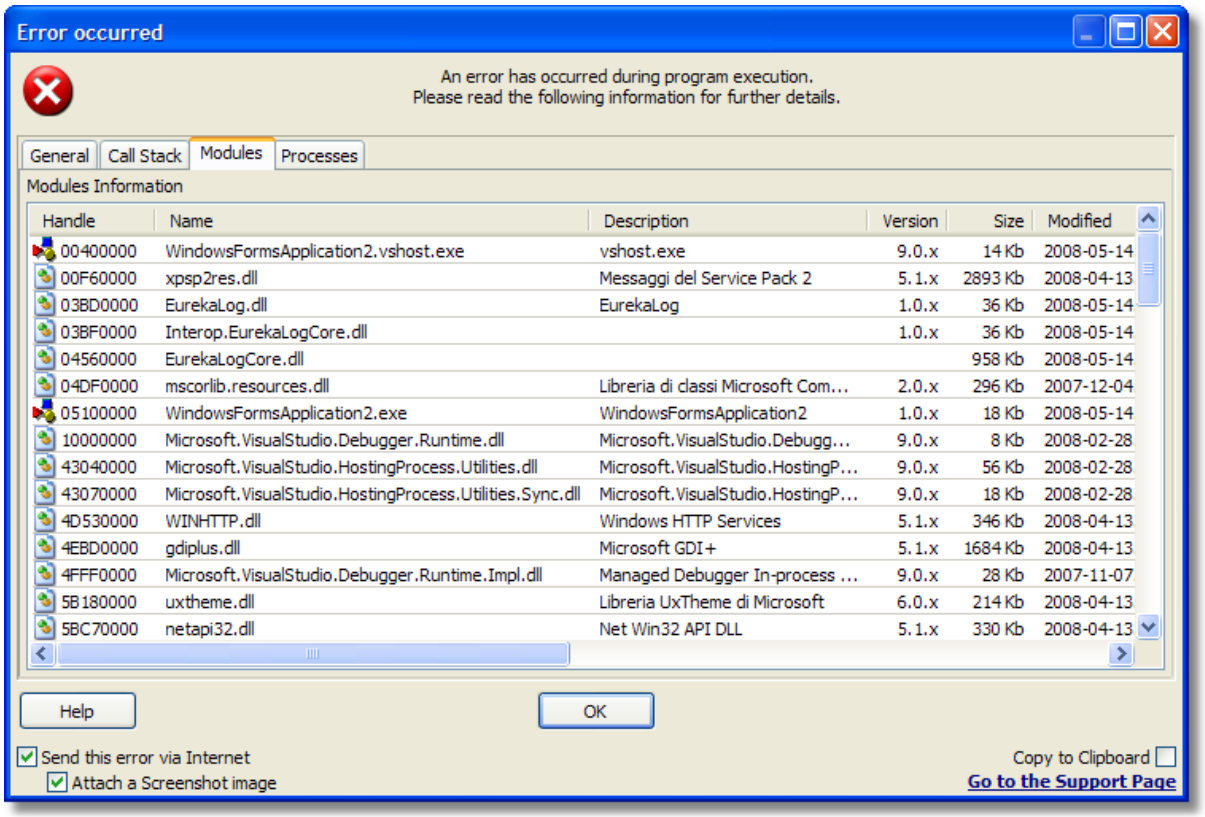
Pressing the "click here" link you can show detailed information about the exception.



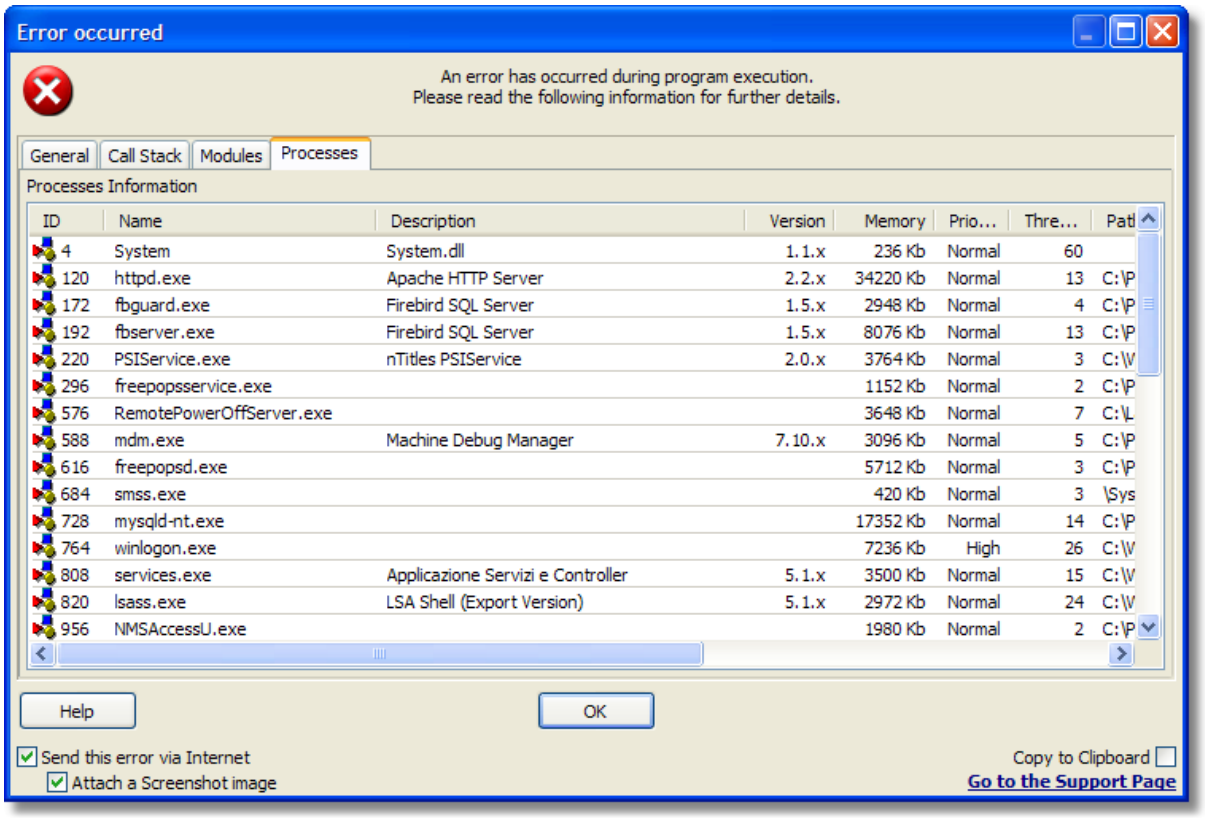
General information.



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



The Modules list.



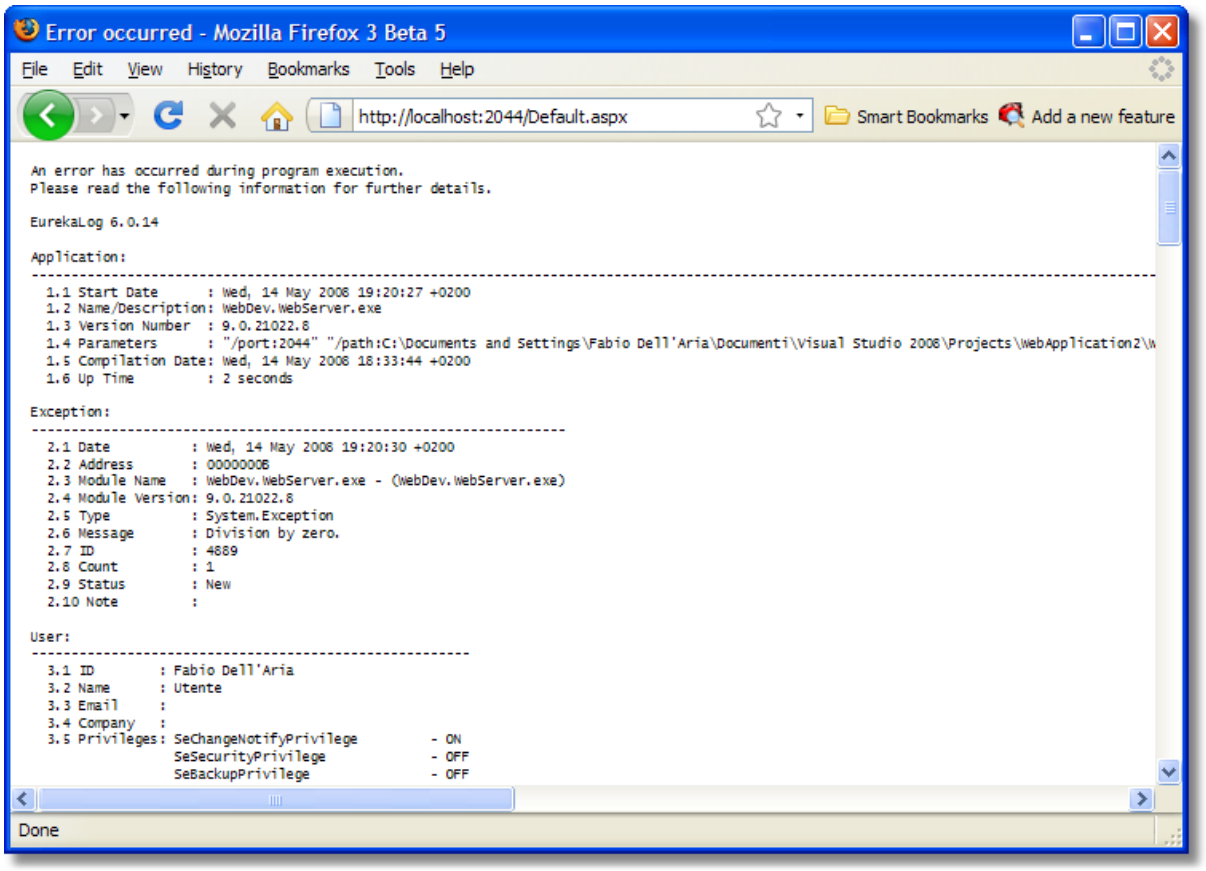
The Processes list.

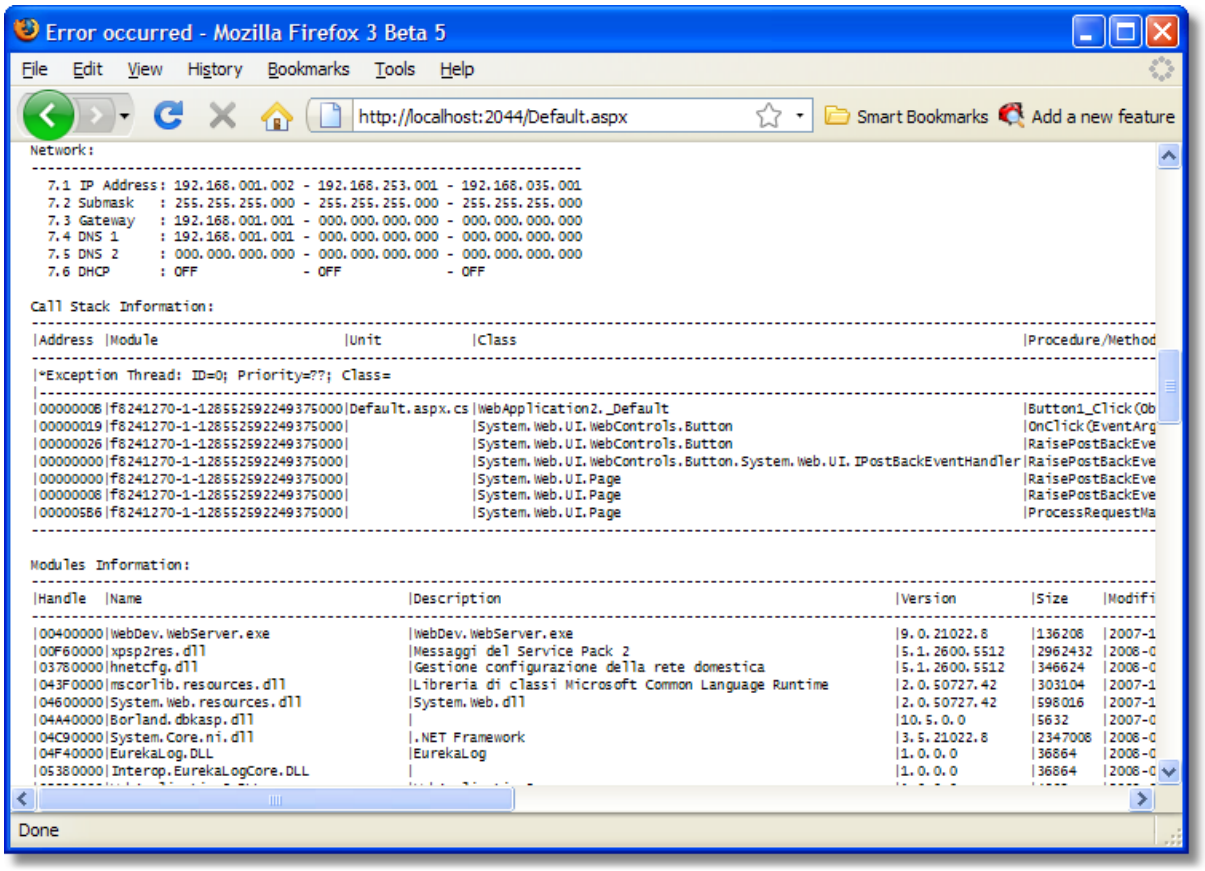
3.2 Console

EurekaLog.NET displays this information when running in a Console application:

3.3 Web

Information shown by EurekaLog.NET when it runs in a Web application:





Part

4 Events

4.1 ExceptionNotify event

4.1.1 How to use ExceptionNotify

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

The syntax of this event is :

```
private void EurekaLog_ExceptionNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, ref bool Handled)
```

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog.NET option at run-time.

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

4.1.2 Examples

In this example, we're going to intercept the exceptions and test, if it is a null object exception, we're going to display a message, and we'll stop the exception from reaching the reporting service.
If it is an unexpected exception, we let EurekaLog.NET handle it.

C# example:

```
private void eurekaLog1_ExceptionNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, ref bool Handled)
{
    if (Error is NullReferenceException)
    {
        MessageBox.Show("Ooopps, someone forgot to create a new instance of that
object.");
        //handled == false means that EurekaLog shouldn't handle this exception. True
is default.
        Handled = false;
    }
    else
    //here we don't change the handled ref, because we want EurekaLog to display the
error dialog.
        MessageBox.Show("I really don't know what's the problem. Honest");
}
```

Visual Basic example:

```
Private Sub EurekaLog1_ExceptionNotify(ByVal [Error] As System.Exception, ByVal
Options As EurekaLogSystem.EurekaLogOptions, ByRef Handled As System.Boolean) Handles
EurekaLog1.ExceptionNotify
    If TypeOf [Error] Is NullReferenceException Then
        MessageBox.Show("Ooopps, someone forgot to create a new instance of that
object.")
        Handled = False 'handled == false means that EurekaLog shouldn't handle this
exception. True is default.
    Else
        MessageBox.Show("I really don't know what's the problem. Honest") 'here we
don't change the handled ref, because we want EurekaLog to display the error dialog.
    End If
End Sub
```

4.2 ExceptionActionNotify event

4.2.1 How to use ExceptionActionNotify

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

The syntax of this event is:

```
private void EurekaLog_ExceptionActionNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.EurekaActionType Action, ref bool Execute)
```

This event is called at each of following action points within EurekaLog.NET:

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

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog.NET option at run-time.

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

4.2.2 Examples

In this example, we're going to make EurekaLog.NET ask for confirmation for every major step, and report when the step's finished.

C# example:

```
private void eurekaLog1_ExceptionActionNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.EurekaActionType Action, ref bool Execute)
{
    switch (Action)
    {
        case EurekaLogSystem.EurekaActionType.atShowingExceptionInfo:
            if (MessageBox.Show("Do you want me to show the exception info?",
"EurekaLog",
                MessageBoxButtons.YesNo) == DialogResult.No) Execute = false;
            break;
        case EurekaLogSystem.EurekaActionType.atShowedExceptionInfo: MessageBox.Show(
"Done, I showed the exception info"); break;
        case EurekaLogSystem.EurekaActionType.atSavingLogFile:
            if (MessageBox.Show("Do you want me to save the log file?", "EurekaLog",
                MessageBoxButtons.YesNo) == DialogResult.No) Execute = false;
            break;
        case EurekaLogSystem.EurekaActionType.atSavedLogFile: MessageBox.Show("Done");
 break;
        case EurekaLogSystem.EurekaActionType.atSendingEmail:
            if (MessageBox.Show("About to send an e-mail. Ok?", "EurekaLog",
                MessageBoxButtons.YesNo) == DialogResult.No) Execute = false;
            break;
        case EurekaLogSystem.EurekaActionType.atSentEmail: MessageBox.Show("Sent");
break;
        case EurekaLogSystem.EurekaActionType.atSendingWebMessage:
            if (MessageBox.Show("Avout to send a web message. Ok?", "EurekaLog",
                MessageBoxButtons.YesNo) == DialogResult.No) Execute = false;
            break;
        case EurekaLogSystem.EurekaActionType.atSentWebMessage: MessageBox.Show("Web
message sent"); break;
        case EurekaLogSystem.EurekaActionType.atTerminating: MessageBox.Show("That's
all folks... application is terminating"); break;
    }
}
```

Visual Basic example:

```
Private Sub EurekaLog1_ExceptionActionNotify(ByVal [Error] As System.Exception, ByVal
Options As EurekaLogSystem.EurekaLogOptions, ByVal Action As EurekaLogSystem.
EurekaActionType, ByRef Execute As System.Boolean) Handles EurekaLog1.
ExceptionActionNotify
    Select Case Action
        Case EurekaLogSystem.EurekaActionType.atShowingExceptionInfo
            If MessageBox.Show("Do you want me to show the exception info?",
"EurekaLog", MessageBoxButtons.YesNo) = DialogResult.No Then
                Execute = False
            End If
        Case EurekaLogSystem.EurekaActionType.atShowedExceptionInfo
            MessageBox.Show("Done, I showed the exception info")
        Case EurekaLogSystem.EurekaActionType.atSavingLogFile
            If MessageBox.Show("Do you want me to save the log file?", "EurekaLog",
MessageBoxButtons.YesNo) = DialogResult.No Then
                Execute = False
            End If
        Case EurekaLogSystem.EurekaActionType.atSavedLogFile
            MessageBox.Show("Done")
        Case EurekaLogSystem.EurekaActionType.atSendingEmail
            If MessageBox.Show("About to send an e-mail. Ok?", "EurekaLog",
MessageBoxButtons.YesNo) = DialogResult.No Then
                Execute = False
            End If
        Case EurekaLogSystem.EurekaActionType.atSentEmail
            MessageBox.Show("Sent")
        Case EurekaLogSystem.EurekaActionType.atSendingWebMessage
            If MessageBox.Show("Avout to send a web message. Ok?", "EurekaLog",
MessageBoxButtons.YesNo) = DialogResult.No Then
                Execute = False
            End If
        Case EurekaLogSystem.EurekaActionType.atSentWebMessage
            MessageBox.Show("Web message sent")
        Case EurekaLogSystem.EurekaActionType.atTerminating
            MessageBox.Show("That's all folks... application is terminating")
    End Select
End Sub
```

4.3 ExceptionErrorNotify event

4.3.1 How to use ExceptionErrorNotify

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

The syntax of this event is:

```
private void EurekaLog_ExceptionErrorNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.EurekaActionType Action, ref bool Retry)
```

This event 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
```

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog.NET option at run-time.

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

4.3.2 Examples

In case an error occurs when trying to save or send an error report, EurekaLog.NET allows the developer to write custom behavior, and ask for a retry. In this simple example, when error occurs, the application asks the user if he wants to retry the operation.

C# example:

```
private void eurekaLog1_ExceptionErrorNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.EurekaActionType Action, ref bool Retry)
{
    switch (Action)
    {
        case EurekaLogSystem.EurekaActionType.atSavingLogFile:
            Retry = MessageBox.Show("Failed saving the log file. Retry?", "EurekaLog",
                MessageBoxButtons.RetryCancel) == DialogResult.Retry;
            break;
        case EurekaLogSystem.EurekaActionType.atSendingEmail:
            Retry = MessageBox.Show("Failed sending an e-mail. Retry?", "EurekaLog",
                MessageBoxButtons.RetryCancel) == DialogResult.Retry;
            break;
        case EurekaLogSystem.EurekaActionType.atSendingWebMessage:
            Retry = MessageBox.Show("Failed sending a web message. Retry?",
"EurekaLog",
                MessageBoxButtons.RetryCancel) == DialogResult.Retry;
            break;
    }
}
```

Visual Basic example:

```
Private Sub EurekaLog1_ExceptionErrorNotify(ByVal [Error] As System.Exception, ByVal
Options As EurekaLogSystem.EurekaLogOptions, ByVal Action As EurekaLogSystem.
EurekaActionType, ByRef Retry As System.Boolean) Handles EurekaLog1.
ExceptionErrorNotify
    Select Case Action
        Case EurekaLogSystem.EurekaActionType.atSavingLogFile
            Retry = MessageBox.Show("Failed saving the log file. Retry?", "EurekaLog",
MessageBoxButtons.RetryCancel) = DialogResult.Retry
        Case EurekaLogSystem.EurekaActionType.atSendingEmail
            Retry = MessageBox.Show("Failed sending an e-mail. Retry?", "EurekaLog",
MessageBoxButtons.RetryCancel) = DialogResult.Retry
        Case EurekaLogSystem.EurekaActionType.atSendingWebMessage
            Retry = MessageBox.Show("Failed sending a web message. Retry?",
"EurekaLog", MessageBoxButtons.RetryCancel) = DialogResult.Retry
    End Select
End Sub
```

4.4 AttachedFilesRequest event

4.4.1 How to use AttachedFilesRequest

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

The syntax of this event is:

```
private void EurekaLog_AttachedFilesRequest(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.StringList AttachedFiles)
```

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog option at run-time.

Use the *"AttachedFiles"* list to add all the needed files using the form: *AttachedataFields.Add ("MyFileFullPath")*

4.4.2 Examples

In this example, the scenario is: The license information for out product is stored in the file % LICENSE_URL%/EurekaLog.license, and the current user's license has expired. So, in case a LicenseExpiredException has been thrown, it would be good to have access to the EurekaLog. license file, right? No problem, we attach that file to the bug report, too.

C# example:

```
private void eurekaLog1_AttachedFilesRequest(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.StringList AttachedFiles)
{
    if (Error is LicenseExpiredException)
    {
        AttachedFiles.Add(String.Format(@"{0}\EurekaLog.license", Environment.
ExpandEnvironmentVariables("%LICENSE_URL%")));
    }
}
```

Visual Basic example:

```
Private Sub EurekaLog1_AttachedFilesRequest(ByVal [Error] As System.Exception, ByVal
Options As EurekaLogSystem.EurekaLogOptions, ByVal AttachedFiles As EurekaLogSystem.
StringList) Handles EurekaLog1.AttachedFilesRequest
    If TypeOf [Error] Is LicenseExpiredException Then
        AttachedFiles.Add(String.Format("{0}\EurekaLog.license", Environment.
ExpandEnvironmentVariables("%LICENSE_URL%")))
    End If
End Sub
```

4.5 CustomDataRequest event

4.5.1 How to use CustomDataRequest

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

The syntax of this event is:

```
private void EurekaLog_CustomDataRequest(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.FieldList DataFields)
```

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog.NET option at run-time.

Use the *"DataFields"* list to add all the needed values using the form: *DataFields.Add ("FieldName", "FieldValue")*

4.5.2 Examples

In this example, we're going to add an additional line in the log file.

C# example:

```
private void eurekaLog1_CustomDataRequest(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.FieldList DataFields)
{
    DataFields.Add("Example", "Just to show where the custom data appears in the log
file");
}
```

Visual Basic example:

```
Private Sub EurekaLog1_CustomDataRequest(ByVal [Error] As System.Exception, ByVal
Options As EurekaLogSystem.EurekaLogOptions, ByVal DataFields As EurekaLogSystem.
FieldList) Handles EurekaLog1.CustomDataRequest
    DataFields.Add("Example", "Just to show where the custom data appears in the log
file")
End Sub
```

Now, if you open the log file, there will be one additional category of data. It should look something like this:

```
Custom Information:
-------------------------------------------------------------------------------
  8.1 Example: Just to show where the custom data appears in the log file
```

4.6 CustomWebFieldsRequest event

4.6.1 How to use CustomWebFieldsRequest

EurekaLog.NET includes a "CustomWebFieldsRequest" event for interaction with the user program when a raised exception is handled by EurekaLog.NET 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:

```
private void EurekaLog_CustomWebFieldsRequest(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.FieldList WebFields)
```

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog.NET option at run-time.

Use the *"WebFields"* list to add all the needed Web fields using the form: *WebFields.Add ("FieldName", "FieldValue")*

4.6.2 Examples

In this example, we're adding two custom web fields, that are required by our web tracker, in order to be able to upload the bug report. Let's say that out bug tracker requires user name and password on upload.
It's simple to add the credential information to the request.

C# example:

```
private void eurekaLog1_CustomWebFieldsRequest(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, EurekaLogSystem.FieldList WebFields)
{
    WebFields.Add("UserName", Options.TrakerUserID);
    WebFields.Add("Password", Options.TrakerPassword);
}
```

Visual Basic example:

```
Private Sub EurekaLog1_CustomWebFieldsRequest(ByVal [Error] As System.Exception, ByVal
Options As EurekaLogSystem.EurekaLogOptions, ByVal WebFields As EurekaLogSystem.
FieldList) Handles EurekaLog1.CustomWebFieldsRequest
    WebFields.Add("UserName", Options.TrakerUserID)
    WebFields.Add("Password", Options.TrakerPassword)
End Sub
```

4.7 CustomButtonClickNotify event

4.7.1 How to use CustomButtonClickNotify

EurekaLog.NET includes an "CustomButtonClickNotify" event that will be executed every time that an user will click on the "Customizable Help Button 13".

The syntax of this event is:

```
private void EurekaLog_CustomButtonClickNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, ref bool CloseDialog)
```

Use the *"Error"* parameter to check the exception type.

Use the *"EurekaLogOptions"* parameter to get and set every EurekaLog option at run-time.

Use the *"CloseDialog"* parameter to choose if close or not the exception dialog.

4.7.2 Examples

EurekaLog.NET offers the option to have an additional button on the error dialog. If you want to take advantage of this feature, you would, certainly, want to define its behavior. You can achieve that using this event.
In this example, we're simply going to ask the user if he/she is certain that he/she wants to close the dialog. However, you may introduce a much more complex behavior, based on your application's needs.
Of course, you have to enable the custom help button in the options dialog first.

C# example:

```
private void eurekaLog1_CustomButtonClickNotify(Exception Error, EurekaLogSystem.
EurekaLogOptions Options, ref bool CloseDialog)
{
    CloseDialog = MessageBox.Show("Do you need help closing this dialog?", "EurekaLog"
,
        MessageBoxButtons.YesNo) == DialogResult.Yes;
}
```

Visual Basic example:

```
Private Sub EurekaLog1_CustomButtonClickNotify(ByVal [Error] As System.Exception,
ByVal Options As EurekaLogSystem.EurekaLogOptions, ByRef CloseDialog As System.Boolean
) Handles EurekaLog1.CustomButtonClickNotify
    CloseDialog = MessageBox.Show("Do you need help closing this dialog?", "EurekaLog"
, MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
End Sub
```

Part V

5 Common routines

5.1 Options access

EurekaLog.NET offers a property named *'CurrentEurekaLogOptions'* for access to the current EurekaLog.NET options.

Syntax of this property is as follows:

```
EurekaLogOptions EurekaLogSystem.ExceptionHandler.CurrentEurekaLogOptions
```

Here are some simple examples on these property's usage.

C# example:

```
// Changing the Email addresses and subject...
EurekaLogSystem.ExceptionHandler.CurrentEurekaLogOptions.EMailAddresses = "support@mai
EurekaLogSystem.ExceptionHandler.CurrentEurekaLogOptions.EMailSubject = "Bug report";
```

Visual Basic example:

```
// Changing the Email addresses and subject...
EurekaLogSystem.ExceptionHandler.CurrentEurekaLogOptions.EMailAddresses = "support@mai
EurekaLogSystem.ExceptionHandler.CurrentEurekaLogOptions.EMailSubject = "Bug report"
```

5.2 Options management

EurekaLog.NET offers two main ways for loading and saving the options file: to a file, or to a stream. Hence, two groups of load/save methods: SaveToStrem / LoadFromStream and SaveToFile / LoadFromFile.

These are the method's prototypes:

```
public void SaveToStream(Stream stream)
public void LoadFromStream(Stream stream)

public void SaveToFile(string FileName)
public void LoadFromFile(string FileName)
```

Here are some simple examples on these method's usage.

C# example:

```
private void eurekaLog1_ExceptionNotify(Exception Error, EurekaLogSystem.EurekaLogOpti
{
    //we want different options when ArgumentNullException is thrown
    //since they change often, we keep those options in a file in the same dir as the
    if(Error is ArgumentNullException)
    {
        Options.LoadFromFile(Directory.GetCurrentDirectory() + "/NullExceptionOptions.
    }
    //we want different options when ArgumentOutOfRangeException is thrown
    //but those options don't change, so we keep those as resources embedded in the ex
    else if (Error is ArgumentOutOfRangeException)
    {
        if (Assembly.GetEntryAssembly().GetManifestResourceNames().Contains("EurekaLog
```

```
        {
            Options.LoadFromStream(Assembly.GetEntryAssembly().GetManifestResourceStre
        }
    }
}

private void eurekaLog1_ExceptionErrorNotify(Exception Error, EurekaLogSystem.EurekaLo
{
    //if sending e-mails fails, don't try to anymore
    if(Action == EurekaLogSystem.EurekaActionType.atSendingEmail)
    {
        Options.EMailSendMode = EmailSendMode.esmNoSend;
        Options.SaveToFile(Directory.GetCurrentDirectory() + "/EmailSendFails.eof");
    }
}
```

Visual Basic example:

```
    Private Sub EurekaLog1_ExceptionNotify(ByVal [Error] As System.Exception, ByVal Op
        'we want different options when ArgumentNullException is thrown
        'since they change often, we keep those options in a file in the same dir as t
        If TypeOf [Error] Is NullReferenceException Then
            Options.LoadFromFile(Directory.GetCurrentDirectory() + "/NullExceptionOpti
        ElseIf TypeOf [Error] Is IndexOutOfRangeException Then
            ''we want different options when ArgumentOutOfRangeException is thrown
            ''but those options don't change, so we keep those as resources embedded i
            If Assembly.GetEntryAssembly().GetManifestResourceNames().Contains("Eureka
                Options.LoadFromStream(Assembly.GetEntryAssembly().GetManifestResource
            End If
        End If
    End Sub

    Private Sub EurekaLog1_ExceptionErrorNotify(ByVal [Error] As System.Exception, ByV
        'if sending e-mails fails, don't try to anymore
        If Action = EurekaLogSystem.EurekaActionType.atSendingEmail Then
            Options.EMailSendMode = EurekaLogSystem.EmailSendMode.esmNoSend
            Options.SaveToFile(Directory.GetCurrentDirectory()+"/EmailSendFails.eof"
        End If
    End Sub
```

5.3 Customized messages

Let's think of a scenario where, for different groups of users you'd like your application to show different custom messages, which the administrator can modify.
Best solution would be to provide editable translation files. So, we need a way for EurekaLog.NET to save and load these files. For that purpose, EurekaLog.NET offers these two methods:

```
public void SaveCustomizedTextsToFile(string FileName)
public void LoadCustomizedTextsFromFile(string FileName)
```

Here's a simple sample on how to use these two methods:

C# example:

```
public void LoadTranslations(EurekaLogSystem.EurekaLogOptions Options, UserRole User)
{
    Options.LoadCustomizedTextsFromFile(Directory.GetCurrentDirectory() + "/Translatio
```

```
        Enum.GetName(UserRole, User) + ".elt");
}

public void SaveTranslations(EurekaLogSystem.EurekaLogOptions Options, UserRole ForUse
{
    Options.SaveCustomizedTextsToFile(Directory.GetCurrentDirectory() + "/Translations
        Enum.GetName(UserRole, ForUser) + ".elt");
}
```

Visual Basic example:

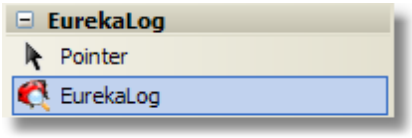
```
Public Sub LoadTranslations(ByVal Options As EurekaLogSystem.EurekaLogOptions, ByVal U
    Options.LoadCustomizedTextsFromFile(Directory.GetCurrentDirectory() + "/Translatio
End Sub

Public Sub SaveTranslations(ByVal Options As EurekaLogSystem.EurekaLogOptions, ByVal F
    Options.SaveCustomizedTextsToFile(Directory.GetCurrentDirectory() + "/Translations
End Sub
```

Part

6 EurekaLog component

6.1 How to use this component

EurekaLog.NET install the "EurekaLog" component into the EurekaLog component palette.



This component has seven events:

- ExceptionNotify 27
- ExceptionActionNotify 28
- ExceptionErrorNotify 30
- AttachedFilesRequest 32
- CustomDataRequest 32
- CustomWebFieldsRequest 33
- CustomButtonClickNotify 34

Part VII

7 Types

7.1 EmailSendMode

This one is the structure of EmailSendMode type:

Namespace EurekaLogSystem.

```
public enum EmailSendMode
{
  esmNoSend = 0,
  esmEmailClient = 1,
  esmSMTPClient = 2,
  esmSMTPServer = 3,
}
```

7.2 WebSendMode

This is the WebSendMode type:

Namespace EurekaLogSystem.

```
public enum WebSendMode
{
  wsmNoSend = 0,
  wsmHTTP = 1,
  wsmHTTPS = 2,
  wsmFTP = 3,
  wsmBugZilla = 4,
  wsmFogBugz = 5,
  wsmMantis = 6,
}
```

7.3 EurekaActionType

This is the EurekaActionType type:

Namespace EurekaLogSystem.

```
public enum EurekaActionType
{
  atShowingExceptionInfo = 0,
  atShowedExceptionInfo = 1,
  atSavingLogFile = 2,
  atSavedLogFile = 3,
  atSendingEmail = 4,
  atSentEmail = 5,
  atSendingWebMessage = 6,
  atSentWebMessage = 7,
  atTerminating = 8,
}
```

7.4 ExceptionDialogType

This is the ExceptionDialogType type:

Namespace EurekaLogSystem.

```
public enum ExceptionDialogType
{
  edtNone = 0,
  edtMessageBox = 1,
  edtMSClassic = 2,
  edtEurekaLog = 3,
}
```

7.5 TFilterHandlerType

This is the TFilterHandlerType type:

Namespace EurekaLogSystem.

```
TFilterHandlerType = (fhtNone, fhtRTL, fhtEurekaLog);
```

7.6 ForegroundType

This is the ForegroundType type:

Namespace EurekaLogSystem.

```
public enum ForegroundType
{
  ftGeneral = 0,
  ftCallStack = 1,
  ftModules = 2,
  ftProcesses = 3,
}
```

7.7 TerminateBtnOperation

This is the TerminateBtnOperation type:

Namespace EurekaLogSystem.

```
public enum TerminateBtnOperation
{
  tbNone = 0,
  tbTerminate = 1,
  tbRestart = 2,
}
```

7.8 CommonSendOptions

This is the CommonSendOptions type:

Namespace EurekaLogSystem.

```
[FlagsAttribute]
public enum CommonSendOptions
{
  sndShowSendDialog = 1,
  sndShowSuccessFailureMsg = 2,
  sndSendEntireLog = 4,
  sndSendXMLLogCopy = 8,
```

```
  sndSendScreenshot = 16,
  sndUseOnlyActiveWindow = 32,
  sndSendLastHTMLPage = 64,
  sndSendInSeparatedThread = 128,
  sndAddDateInFileName = 256,
  sndAddComputerNameInFileName = 512,
}
```

7.9 ExceptionDialogOptions

This is the ExceptionDialogOptions type:

Namespace EurekaLogSystem.

```
[FlagsAttribute]
public enum ExceptionDialogOptions
{
  edoSendErrorReportChecked = 1,
  edoAttachScreenshotChecked = 2,
  edoShowCopyToClipOption = 4,
  edoShowDetailsButton = 8,
  edoShowInDetailedMode = 16,
  edoShowInTopMostMode = 32,
  edoUseEurekaLogLookAndFeel = 64,
  edoShowSendErrorReportOption = 128,
  edoShowAttachScreenshotOption = 256,
  edoShowCustomButton = 512,
}
```

7.10 LogOptions

This is the LogOptions type:

Namespace EurekaLogSystem.

```
[FlagsAttribute]
public enum LogOptions
{
  loNoDuplicateErrors = 1,
  loAppendReproduceText = 2,
  loDeleteLogAtVersionChange = 4,
  loAddComputerNameInLogFileName = 8,
  loSaveModulesAndProcessesSections = 16,
}
```

7.11 EurekaLogOptions

This is the EurekaLogOptions type:

Namespace EurekaLogSystem.

```
// Options Class...
public class EurekaLogOptions
{
  // Properties...
  public virtual bool AppendLogs ...
```

```
    public virtual string AttachedFiles ...
    public virtual int AutoCloseDialogSecs ...
    public virtual int AutoCrashMinutes ...
    public virtual int AutoCrashNumber ...
    public virtual TerminateBtnOperation AutoCrashOperation ...
    public virtual CommonSendOptions CommonSendOptions ...
    public virtual string EMailAddresses ...
    public virtual string EMailMessage ...
    public virtual EmailSendMode EMailSendMode ...
    public virtual string EMailSubject ...
    public virtual int ErrorsNumberToSave ...
    public virtual int ErrorsNumberToShowTerminateBtn ...
    public virtual ExceptionDialogOptions ExceptionDialogOptions ...
    public virtual ExceptionDialogType ExceptionDialogType ...
    public virtual ForegroundType ForegroundTab ...
    public virtual LogOptions LogOptions ...
    public virtual string OutputPath ...
    public virtual string ProxyPassword ...
    public virtual ushort ProxyPort ...
    public virtual string ProxyURL ...
    public virtual string ProxyUserID ...
    public virtual bool SaveLogFile ...
    public virtual string SMTPFrom ...
    public virtual string SMTPHost ...
    public virtual string SMTPPassword ...
    public virtual ushort SMTPPort ...
    public virtual string SMTPUserID ...
    public virtual string SupportURL ...
    public virtual TerminateBtnOperation TerminateBtnOperation ...
    public virtual string TrakerAssignTo ...
    public virtual string TrakerCategory ...
    public virtual string TrakerPassword ...
    public virtual string TrakerProject ...
    public virtual string TrakerTrialID ...
    public virtual string TrakerUserID ...
    public virtual string WebPassword ...
    public virtual ushort WebPort ...
    public virtual WebSendMode WebSendMode ...
    public virtual string WebURL ...
    public virtual string WebUserID ...
    public virtual string ZipPassword ...
    public string[] CustomizedTexts ...
    public string TextsCollection ...
}
```

Part VII

8 EurekaLog Viewer

8.1 How to use the Viewer

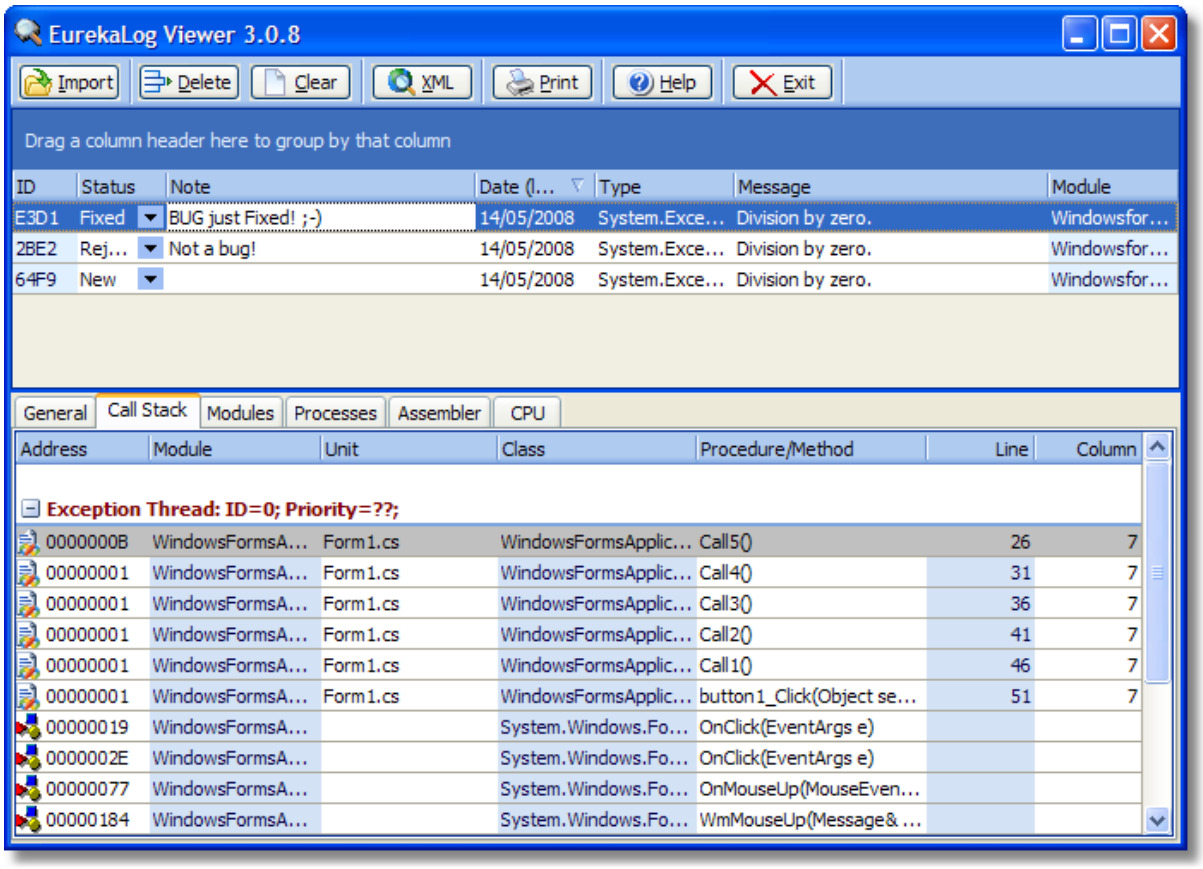
Use this Viewer to analyze and process all the bug reports received from EurekaLog.NET (.el files).

You can open this program with a double-click on the Log file (.el) or using the "View Project Log..." 9 new item in the EurekaLog IDE menu.

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

Use the Delete button to remove the currently selected bug report.

use the Print button to print the current selected bug report.



Part

9 Support

9.1 On-line support

Feel free to visit http://www.eurekalog.com/support.php or send an e-mail to support@eurekalog.com

Index