

The Microsoft® .NET common language runtime provides a means for notifying programs of exceptions in a uniform way, even if the module generating the exception is written in a different language than the one handling the exception.

Exceptions represent a breach of an implicit assumption made within code. For example, if your code tries to access a file that is assumed to exist, but the file is missing, an exception would be thrown. However, if your code does not assume that the file exists and checks for its presence first, this scenario would not necessarily generate an exception.

Exceptions are not necessarily errors. Whether or not an exception represents an error is determined by the application in which the exception occurred. An exception that is thrown when a file is not found may be considered an error in one scenario, but may not represent an error in another. <http://msdn2.microsoft.com/en-us/library/ms954599.aspx>

The idea of creating a custom exception handler for unhandled exceptions is to create a uniform method of not only providing user feedback but also for a uniform way to record selected information to a log file. The first iteration of this exception handler will record information to a simple text file while the next release will move to an XML file format suitable for querying, filtering and reporting via a GUI interface specifically designed to work against the log file.

Keeping things simple for instrumentation all that is required is to

1. Add a DLL and configuration file to your project
  - a. Configure the above DLL configuration XML file which contains variable settings as listed below

**Sample**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<UHE>
  <PrimaryLogFile>\\dat2\IPD\DEV\Shared\Utilities\logExceptions.txt</PrimaryLogFile>
  <SendEmailTooAddress>GALLAGHE@exchange2.dor.state.or.us</SendEmailTooAddress>
  <EmailFromAddress>GALLAGHE@exchange2.dor.state.or.us</EmailFromAddress>
  <SmtpClientServer>exchange2.dor.local</SmtpClientServer>
</UHE>
```

2. Add the code shown in <figure 1> into ApplicationEvents.vb
3. All files reside in the executable/primary assembly folder.
4. Reference AppUnhandledExceptions to your project.

## Figure 1

```
Imports System.Windows.Forms
Imports AppUnhandledExceptions
Namespace My
    ' The following events are available for MyApplication:
    '
    ' Startup: Raised when the application starts, before the startup form is created.
    ' Shutdown: Raised after all application forms are closed. This event is not raised if the application terminates abnormally.
    ' UnhandledException: Raised if the application encounters an unhandled exception.
    ' StartupNextInstance: Raised when launching a single-instance application and the application is already active.
    ' NetworkAvailabilityChanged: Raised when the network connection is connected or disconnected.
    Partial Friend Class MyApplication
        Private Sub MyApplication_UnhandledException(ByVal sender As Object, ByVal eTheException As
Microsoft.VisualBasic.ApplicationServices.UnhandledExceptionEventArgs) Handles Me.UnhandledException
            Dim AppInfo As New LocalApplicationInformation
            Dim LogFileError As String = ""

            Try
                With New TheLogFile
                    .LogFileName = AppInfo.UnHandledExceptionLogFile
                    .WriteFile(LocalApplicationInformation.SysInfoToString(eTheException.Exception, True, True, True))
                End With
            Catch ex As Exception
                LogFileError = "Failed to write to application log file, please notify a developer"
            End Try

            Dim f As New ExceptionDialog
            Try
                f.Text = My.Application.Info.Title
                ' Toggles the state of a label which when visible alerts the viewer to
                ' contact a developer indicating the exception was not written to disk.
                If LogFileError.Length > 0 Then
                    f.ShowLogError = True
                End If
                f.ErrorBoxText = String.Format("An untrapped error occurred.{0}The error message was:{0}{1}", Environment.NewLine,
eTheException.Exception.Message)
                f.AppDetailsButton = True
                f.ErrorMoreText = Environment.NewLine & StackTracer.EnhancedStackTrace(eTheException.Exception, True)
                f.ShowDialog()
            Finally
                f.Dispose()
            End Try
        End Sub
    End Class
End Namespace
```

Behavior of the new unhandled exception handler

1. Will terminate after recording information to your log file
2. User is presented with a simple dialog which is capable of showing extended exception information.

## **IMPORTANT**

The log file will not be written too for the following reasons

- The path to the file is invalid
- The path does not exists
- If a network path the network may be unavailable.
- The user doesn't have permissions to this folder in regards to network and/or .NET policy permissions.
- The user does not have write permissions to the folder.
- You forgot to enter a path/filename in the DLL XML configuration file

## Flow of an unhandled exception

When an application instrumented with the unhandled exception handler encounters an exception

1. Control is transferred to `Class MyApplication` in `ApplicationEvents.vb` which calls methods within the custom unhandled exception handler DLL.
2. A local copy of the `Logfile` class is created using the class `LocalApplicationInformation` where the new constructor reads from an XML configuration file to get the path/filename of the logfile for this application.
3. “`LocalApplicationInformation.SysInfoToString`” is called within the write method of the logfile class which gets all the information needed to record the unhandled exception to the applications unhandled exception logfile.
4. A Windows dialog is invoked from the unhandled exceptions custom DLL which tells the operator of the application the application needs to terminate. The dialog describes in short details what caused the problem and to contact a developer responsible for this application. If the developers of the application elected to have an email sent to them they will receive an e-mail to a designated e-mail group configured specifically for this purpose. The e-mail switch is within the custom DLL XML configuration file. There are three fields for the e-mail, if all three are not filled in then the DLL will not send an e-mail.

## Configuration file for the unhandled exceptions DLL

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<UHE>
  <PrimaryLogFile>\\dat2\IPD\DEV\Shared\Utilities\logExceptions.txt</PrimaryLogFile>
  <SendEmailTooAddress>GALLAGHE@exchange2.dor.state.or.us</SendEmailTooAddress>
  <EmailFromAddress>GALLAGHE@exchange2.dor.state.or.us</EmailFromAddress>
  <SmtpClientServer>exchange2.dor.local</SmtpClientServer>
</UHE>
```

### PrimaryLogFile

Location of the logfile and actual name of the logfile to write entries into for unhandled exceptions only. If the logfile is not accessible for any reason the operator/user will have a message indicating this in the dialog discussed above in step four.