Reporting unhandled exceptions in your Windows Phone apps

While many bugs are caught during development, nothing tests an app as much as real-world testing on a deployed application. At this point it is important to be able to get debug information back from the end-user. This article collects links to other public topics on how to catch unhandled exceptions in your app, log them, and send them to the developer for analysis.

Basic methods

There are two basic methods for getting information about exception conditions that occur in an app:

- Detect when an exception occurs and save the extended error information.

  There is a method named `Application_UnhandledException()` in **app.xaml.cs** file which gets called whenever an unhandled exception gets fired. Extended exception information like the stack trace can be written to a crash log here:

```
private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
{
 string errorString = DateTime.Now.ToLocalTime().ToString() + " | " + "Message: " +
e.ExceptionObject.Message + "Stack Trace: " + e.ExceptionObject.StackTrace;
 WriteCrashLog(errorString);
}
```

- Log data at checkpoints in code. This allows you to verify "last good" points in your code execution and get some information about the state of the running app.
- If you're running the app yourself you can display message boxes at checkpoints: this can be more comfortable than logging and having to read the log later. Normally you won't do this in an app deployed to the customer.

Once a crash log has been created, the user should be informed that a problem has occurred and asked to confirm whether the log can be uploaded to a bug tracking server or sent to a specified email address (during debugging the log may also be displayed in a special page in the app).

Ideally the user should be informed of the crash at the point it occurs - this is what is done in the example code. Unfortunately, following a crash there is no guarantee that the app will be in a suitable state for sending the log. As a result it is common practice in commercial apps to inform the user when the app is next restarted.

Article list

Various approaches based on the above methods are well documented on the Internet. The following articles are recommended reading:

- Handling Exceptions on Windows Phone 7 (Jimmy's Blog)
- Exception Class (MSDN)
- Handling and Throwing Exceptions (MSDN)
- Managing Errors in a Windows Phone Silverlight Application (zorn consulting blog)
- Error Reporting on Windows Phone 7 (Andy Pennell's Blog)
- Windows Phone 7 Error Handling, Reporting (geoffhudik blog)
- Search on Exception handling (google)

Example code

The attached example is a modification of Listbox handling in Windows Phone 7 in which `listBoxFlickerSearch_SelectionChanged()` creates an unhandled exception when a user clicks on the list item. The exception is then displayed in its own view. This approach is very similar to that described in Jimmy's Blog above.

The full source code of the example is available here: File:ErrorHandler.zip