

SeawaxLog User Manual

Classes and applications to add pro active error monitoring and event logging to your webapp based applications and web services.

1

Version 1.1.0, March 29, 2011

¹© 2011 SeawaxLog is an Antwise Solutions product

Index

Introduction.....	3
Background reasons for writing this product.....	4
SeawaxLog example Workspace.....	5
Installing the SeawaxLog example workspace.....	5
Examples	6
Automatically adding page visits to the access log.....	6
Adding login event to the action log.....	6
Adding application specific events to the action log.....	7
Adding User ID and login status to the logs.....	8
SeawaxLog support applications.....	10
WxLogs.....	10
Error Log.....	10
Action Log.....	14
Page Event.....	17
Application Actions.....	19
System Settings.....	21
WxEmailErrors.....	23
WxTestEmailConfig.....	24
Adding the code to your workspace.....	25
Class cWxLog.....	26
Class cWxWebAppError.....	27
Class cWxWebApplicationLog.....	28
Class cWxWebEventLog.....	29

Introduction

The SeawaxLog product is a set of classes and applications to automatically log and report runtime errors, page access and application actions. All of these events are logged to a database and can be easily inspected using the supplied support applications. An application to automatically send email notifications when an error happens is included as well.

By using this product you have much better control to act upon problems in your product that are otherwise missed when users fail to report the problem they bumped into. When they do report problems you now have an audit trail of events with status of your application and things like a stack trace and HTTP POST headers to better pinpoint why this problem has happened.

The product has been designed in a modular way and consists of the following main components:

Windows based support applications:

- WxLogs application – With this application you can view the log details. It has a view for errors triggered, pages viewed and actions happened.
- WxEmailErrors application – This application can be setup to run as a scheduled task to automatically send any errors that happened by email.
- WxTestEmailConfig application – A simple application to test your current email settings.

The SeawaxLog example workspace:

An example workspace which is based on the standard order entry application from Data Access Worldwide which has some small changes made to it to highlight how-to use the techniques in question.

The SeawaxLogLib\Web Library:

This library contains all of the code needed for your webapp projects and webservises. It only has to be added to your project and can be included and activated with a few lines of code.

The SeawaxLogLib\Win Library:

This library contains all of the code needed for the windows based support applications. It only has to be added to your project after which you can compile the aforementioned support applications from above.

A set of DataFlex data files that you have to add to your application which are used to save the logging data. It is suggested to keep these files as DataFlex files as that way you can also log any database connectivity kit related errors.

Background reasons for writing this product

The reason this product is born is that it gives you a one stop place to check if problems occur. Without this tool whenever there's problems you'll have to check the logs from:

a.) WebApp administrator

By default the event log contains both connections and errors in the same log. Finding error reports is difficult on a busy web site. The event log view even becomes non responsive when you try to navigate it.

If you turn off the logging in the Web Application administrator tool (uncheck "Log all access to the Web Application"), it only logs the errors.

Problems with this event log are:

1. Unfortunately it is still difficult to navigate, when a user connects or a new error is triggered the cursor skips back to the top.
2. There's no way you can get that info except by logging into the host. In most cases the host is remote which complicates things a bit.
3. It just has the error and line, which quite often is not enough to troubleshoot the problem.
4. There's no way to extend it, no way to read the data outside of webapp administrator, the solution is completely proprietary.

b.) Windows event logs

The errors to look for here are spread over the application log (problems with your webapp processes) and the system log (problems with the service)

Of course any other windows error is also in here, so if you're not lucky you'll be searching like crazy.

The solution

By logging errors directly to a data file, you now have the ability to do anything you like with the error data.

Examples of what can be done are:

- You can create a web page to browse recent errors,
- Automatically email yesterday's errors in the morning,
- Email critical errors happening on your web site directly to the web site administrator. (this is part of the product, host license or better)

SeawaxLog example Workspace

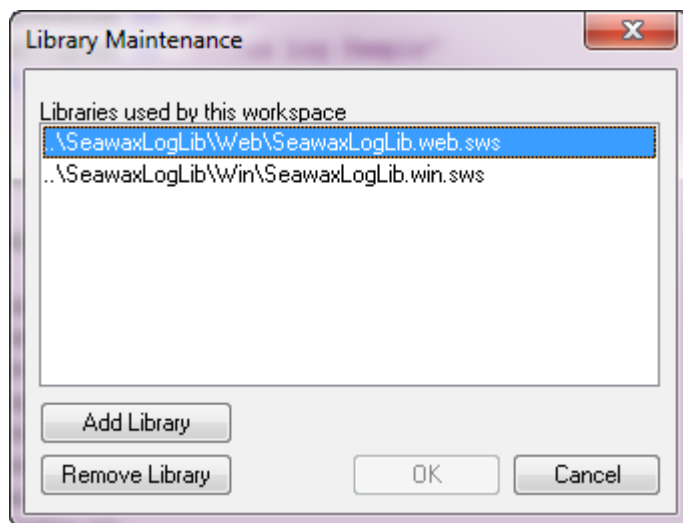
Installing the SeawaxLog example workspace

The example workspace “SeawaxLog” has to be registered in the Visual DataFlex Studio to properly use it. The SeawaxLogLib\Web library contains the code which takes care of the actual logging.

The SeawaxLogLib\Win library contains the code for the windows support applications.

You’ll need to register the SeawaxLog workspace in the Studio by going to the menu and choosing File-> Open Workspace and selecting the .sws file.

After you have done that, add the SeawaxLogLib\Web and SeawaxLogLib\Win libraries to the SeawaxLog workspace via menu option Tools->Maintain Libraries.



In order to run the webapp example, you must register the webapp using webapp administrator using the following steps:

- File > Configure New Web Application
- Select “Local Application Configuration”, Click Next
- Now browse to the webapp.exe under the SeawaxLog workspace, Click Next
- Select Process Pooling if you want or leave it at the default, Next
- Change Virtual Directory Name into “SeawaxLog” (Without the quotes), Next
- Application Name: SeawaxLog, Click Finish

You can now access the demo application from your browser at <http://localhost/SeawaxLog>

Examples

The standard Order Entry workspace has been renamed to SeawaxLog and has been extended with examples on how-to use the SeawaxLog libraries. Below we've highlighted some of those examples to show the most important features.

Automatically adding page visits to the access log

The method of automatically adding your page to the access log is very easy.

At the top of each .asp page add the call DoLogPage as follows, you can see this in each asp page in the example workspace.

If you already call a standard method from the top of your page, then it is suggested that you add the call in there as that would mean you wouldn't have to change all of your asp pages. The method will still be able to detect from which .asp page the call is invoked.

So your asp page should look like this:

```
<%  
    owxLog.call "msg_DoLogPage"  
%>  
  
<html>  
  
<head>
```

Or if your first call on each page is called "InitPage" as in:

```
<%  
    oSession.call "msg_InitPage"  
%>  
  
<html>  
  
<head>
```

Then augment your InitPage method in the oSession Web Object as:

```
Procedure InitPage  
    Send DoLogPage Of ghowxLog  
  
    // the rest of your method  
    ...  
End_Procedure // InitPage
```

Adding login event to the action log

In order to follow the status of what has happened while you are trying to unravel a problem, it can be very useful to log certain interesting state changes of your application. For example your application is likely to have a different behavior when someone is logged in, in comparison to not logged in. Knowing which user has logged in can be useful too.

In general the way to do that is to augment your login method. In the example workspace that is in the login.wo web object.

```
Function Login_User String sUserId String sPass Returns Integer
Integer bOk

// clear Login DD, Move values to file buffer and
// attempt a find EQ
Get ddClear "Users" To bOk
Send setDDValue "Users.loginName" (Uppercase(sUserId))
Send setDDValue "Users.password" (Uppercase(sPass))
Get ddFind "Users" 2 Eq To bOk
If (bOk) Send DoLogAction Of ghWxLog ACTION_USER_LOGIN (Uppercase(sUserId))
Else Send DoLogAction Of ghWxLog ACTION_LOGIN_FAILED (Uppercase(sUserId))
Function_Return bOk
End_Function
```

The last parameter in the call to DoLogAction, (Uppercase(sUserId)) is added as the description to your action log entry. This is a free field, you can put other info in there as well.

Adding application specific events to the action log

The action logging has been designed to make it easy to customize it to your application. In the Order Entry example we are maintaining a simple ordering system and it has a very basic inventory system where you can add or change products.

There are times when you would want to log these state changes, if only for auditing purposes, or to team up with the other logs when you are troubleshooting. For the purpose of this example we added the states “New Product saved” and “Product Details Updated”. You can easily add these states in the “Application Actions” screen in the WxLogs application. After regenerating the ActionCodes.h file, you can then log these states as follows (see invt.wo)

```
Object Invt_DD is a Invt_DataDictionary
Set DDO_Server to Vendor_DD
Send DefineAllExtendedFields

//
// Augmented to track when a user saves a new product or updates an
// existing one
Procedure Request_Save
Boolean bHasRecord
Boolean bNewRecord
String sItemID

Move False To bNewRecord
Get HasRecord To bHasRecord
If (bHasRecord=false) Begin
    Move True To bNewRecord
End
```

```

    Get Field_Current_Value Field Invt.Item_Id To sItemID
    Forward Send Request_Save
    If (Err=false) Begin
        If (bNewRecord) Send DoLogAction Of ghoWxLog ACTION_NEW_PRODUCT sItemID
        Else Send DoLogAction Of ghoWxLog ACTION_PRODUCT_UPDATED sItemID
    End
End_Procedure // Request_Save
End_Object    // Invt_DD

```

Adding User ID and login status to the logs

As mentioned before you can add application specific data to your logs using a callback mechanism. Standard information would be a user ID and logged in status. In addition to that you have 3 more custom data fields you can fill and set.

Down here we illustrate how-to set User ID and logged in status only, but the method for setting custom data is exactly the same.

In our order entry workspace example, the user id and logged in status of the current user is stored in a cookie. So down here we get this information from the cookie, in your system it could come from a datafile or a property somewhere. The method is still pretty much the same.

The file to look at is WxLog.wo. This webobject is in its default form completely empty (see the SeawaxLogLib/Web library for this version) In the order entry example we have copied that file into the workspace and added the following code.

Use cWxLog.pkg

Object oWxLog is a cWxLog

```

Function UserNameFromCookie Returns String
    String sUserName

    Move "" To sUserName
    Get Cookie "User" "Name" To sUserName
    Function_Return sUserName
End_Function // UserNameFromCookie

Function LoggedInStatusFromCookie Returns Boolean
    Boolean bIsLoggedIn
    String sValue

    Move False To bIsLoggedIn
    Get Cookie "Rights" "" To sValue
    If (sValue="1") Move True To bIsLoggedIn
    Function_Return bIsLoggedIn
End_Function // LoggedInStatusFromCookie

```

```

//
// You can use this method in your object instantiation of this class to have

```



```

// it populate any of the following properties:
//   psCurrentUserID
//   pbLoggedIn
//   psCustomData1
//   psCustomData2
//   psCustomData3
//
// If you fill these in then the data that is used is saved in any of the log
// files.
Procedure onUpdateLog
  String sCurrentUserId
  Boolean bLoggedIn

  Get UserNameFromCookie To sCurrentUserId
  Get LoggedInStatusFromCookie To bLoggedIn

  Set psCurrentUserID To sCurrentUserId
  Set pbLoggedIn      To bLoggedIn
End_Procedure // onUpdateLog

End_Object // oWxLog

```

The onUpdateLog method is a so called callback method and is called from within each write action to the log. It allows you to set the five properties, psCurrentUserID, pbLoggedIn, psCustomData1, psCustomData2 and psCustomData3.

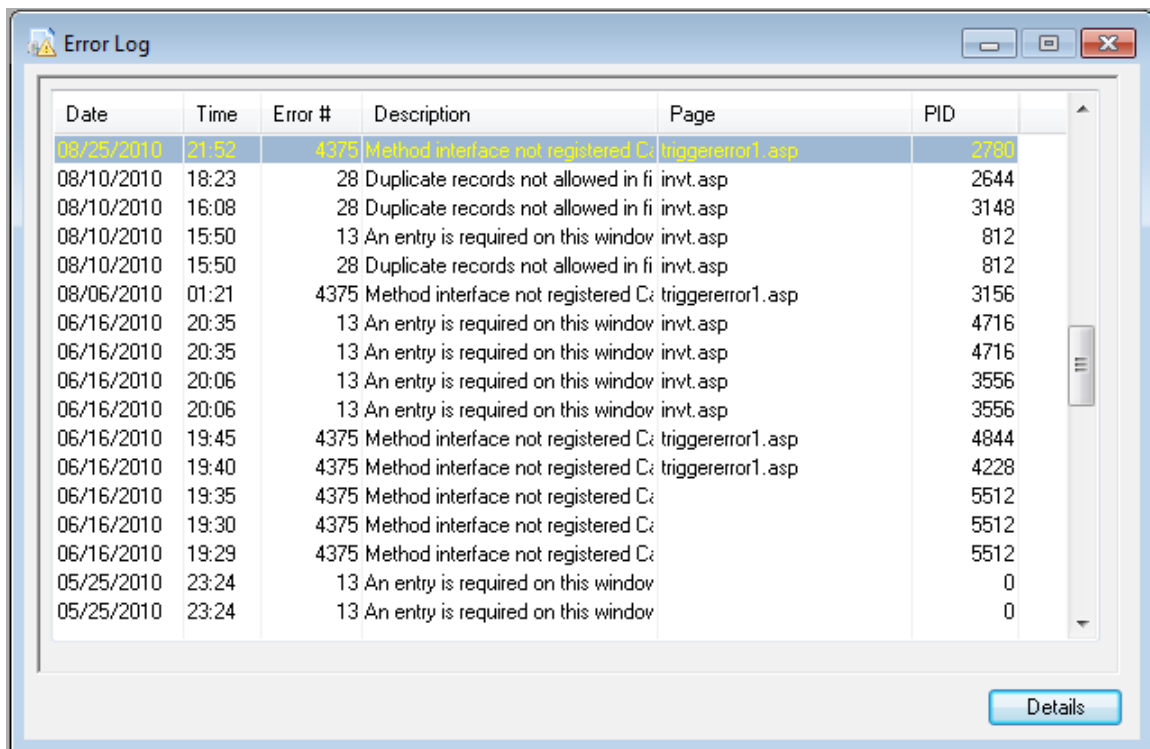
SeawaxLog support applications

WxLogs

An easy way to inspect the SeawaxLog logfiles is to start the WxLogs application. It has a variety of views that give you an easy insight into the collected data.

Error Log

Every time an error is triggered by the webapp or webservice, it gets written out into the Error Log.



The screenshot shows the 'Error Log' application window. It contains a table with the following columns: Date, Time, Error #, Description, Page, and PID. The table lists several error entries, with the first row highlighted in blue. A 'Details' button is located at the bottom right of the window.

Date	Time	Error #	Description	Page	PID
08/25/2010	21:52	4375	Method interface not registered C:\triggererror1.asp		2788
08/10/2010	18:23	28	Duplicate records not allowed in file invt.asp		2644
08/10/2010	16:08	28	Duplicate records not allowed in file invt.asp		3148
08/10/2010	15:50	13	An entry is required on this window invt.asp		812
08/10/2010	15:50	28	Duplicate records not allowed in file invt.asp		812
08/06/2010	01:21	4375	Method interface not registered C:\triggererror1.asp		3156
06/16/2010	20:35	13	An entry is required on this window invt.asp		4716
06/16/2010	20:35	13	An entry is required on this window invt.asp		4716
06/16/2010	20:06	13	An entry is required on this window invt.asp		3556
06/16/2010	20:06	13	An entry is required on this window invt.asp		3556
06/16/2010	19:45	4375	Method interface not registered C:\triggererror1.asp		4844
06/16/2010	19:40	4375	Method interface not registered C:\triggererror1.asp		4228
06/16/2010	19:35	4375	Method interface not registered C:\triggererror1.asp		5512
06/16/2010	19:30	4375	Method interface not registered C:\triggererror1.asp		5512
06/16/2010	19:29	4375	Method interface not registered C:\triggererror1.asp		5512
05/25/2010	23:24	13	An entry is required on this window invt.asp		0
05/25/2010	23:24	13	An entry is required on this window invt.asp		0

You get a quick overview of the errors that have happened by opening the Error Log View from the View menu.

This list shows you the date/time, DataFlex error number triggered, the webapp asp page from where the error was triggered and the windows Process ID (PID) of the webapp process triggering the error. Knowing the PID helps you tracking the error when process pooling is enabled..

After selecting an error you can see more details by either double clicking on the row or by clicking on the details button below.

The screenshot shows a window titled "Error Log Details" with a close button (X) in the top right corner. The window contains several input fields and checkboxes for error details:

- Date: 08/25/2010
- Time: 21:52
- Error Number: 4375
- Line Number: 8660
- Process ID: 2780
- User ID: (empty)
- Script Page: triggererror1.asp
- Origin Object: 0
- Primary Key: (empty)
- Operation: W
- ☐ Processed
- ☐ Logged In
- Description: Method interface not registered Call Error: Message not registered msg_CallingAnUndefinedMessa

Below the input fields is a tabbed interface with four tabs: "Details" (selected), "Data", "Stack Trace", and "Custom". The "Details" tab shows the following text:

```
Method interface not registered Call Error: Message not registered msg_CallingAnUndefinedMessageHere
VDF Error#: 4375 in line: 8660.
```

At the bottom right of the window are three buttons: "Previous", "Next", and "Close".

The details screen gives you more information about the error and the origination of the error. Besides the fields also in the list view, you can see here.

Error Line Number, which is the normal error line number as in the default error object.

User ID: If your webapp identifies user by ID/number or name then you can set this field from within a Callback method accordingly (See WxLogs.pkg)

Origin Object, if the error was triggered from within a datadictionary, then the origin object will show the name of the datadictionary that triggered the error.

Primary Key, if a record is in memory then the PK form will contain the unique record ID field data of the main datadictionary.

Operation: is the datadictionary operation when the error occurred.

Processed is a field that indicates if the error has been read by the process that generates the email. Basically it tells you if an email has been send to the administrator.

Logged in this value can be set from within a Callback method in WxLogs

The Previous and Next buttons allow you to select previous or next lines in the underlying list view. In other words, you can use them to see the next or previous error.

You can also use the shortcut keys Ctrl+Up Arrow and Ctrl+Down Arrow to access these buttons.

The screenshot shows a window titled "Error Log Details" with a close button (X) in the top right corner. The window contains several input fields for error details:

Date:	08/25/2010	Script Page:	triggererror1.asp
Time:	21:52	Origin Object:	0
Error Number:	4375	Primary Key:	
Line Number:	8660	Operation:	W
Process ID:	2780	<input type="checkbox"/> Processed	
User ID:		<input type="checkbox"/> Logged In	
Description:	Method interface not registered Call Error: Message not registered msg_CallingAnUndefinedMessa		

Below the input fields are four tabs: "Details", "Data", "Stack Trace", and "Custom". The "Stack Trace" tab is selected, showing the following text:

```
MSG_ERROR_REPORT (433) - 0wXWEBAPPERROR (18) - at address 14317  
GET_DDCALL (1284) - OSALESP (174) - at address 8660  
MSG_ONDFFUNC (4880) - DVDFINETSESSION (33) - at address 7213  
MSG_STARTWEBAPP (5240) - OWEBAPP (31) - at address 11395  
[start] - at address 17153
```

At the bottom of the window are three buttons: "Previous", "Next", and "Close".

Having the stack trace of when an error occurs can sometimes provide you with a direct clue of why the error happened. In this example it tells you that the undefined method in the script page triggererror1.asp is not published in object oSalesp.

Error Log Details

X

Date:08/10/2010

Script Page:invnt.asp

Time:18:23

Origin Object:0

Error Number:28

Primary Key:

Line Number:3935

Operation:W

Process ID:2644

☐ Processed

User ID:

☐ Logged In

Description:Duplicate records not allowed in file ..\SeawaxLog\Data\invnt.k1

Details

Data

Stack Trace

Custom

HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pipe, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml, */*

HTTP_ACCEPT_LANGUAGE:en-us

HTTP_CONNECTION:Keep-Alive

HTTP_HOST:localhost

HTTP_REFERER:http://localhost/SeawaxLog/Invnt.asp

HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 1.1.4322)

HTTP_COOKIE:ASPSESSIONIDAARRCQRC=LFBFHKAMBKPFBMMLHKIDALB; Rights=1; User=FullName=John+J%2E+Smith&Pass=john&Name=john

HTTP_UA_CPU:x86

HTTP_CONTENT_LENGTH:176

HTTP_CONTENT_TYPE:application/x-www-form-urlencoded

HTTP_ACCEPT_ENCODING:gzip, deflate

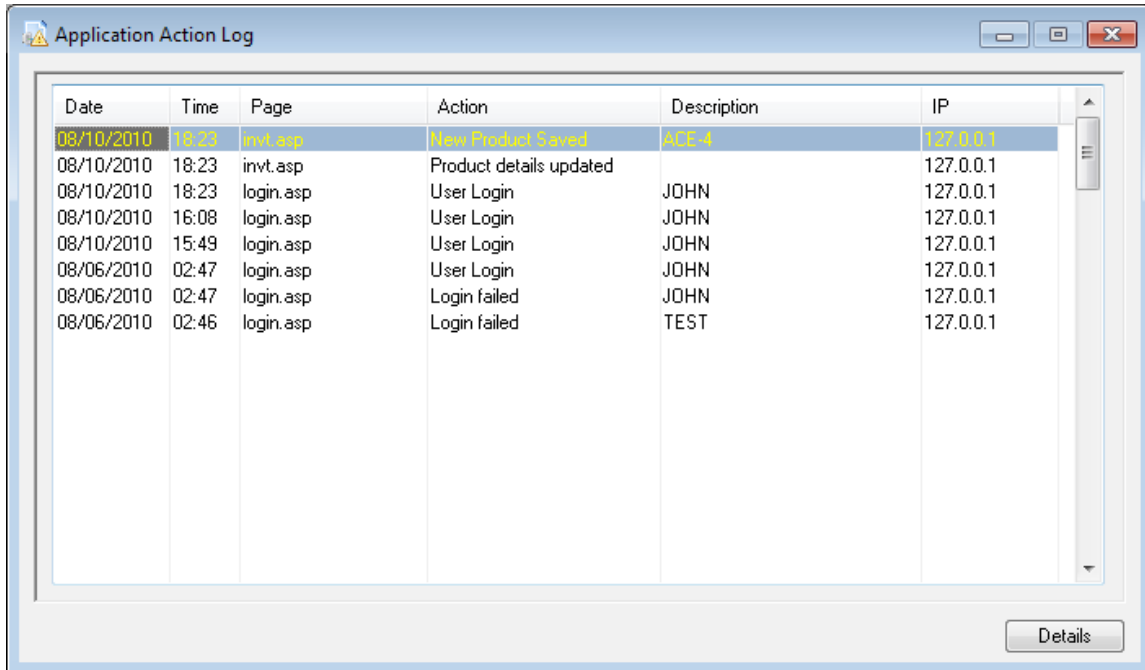
Previous

Next

Close

For a GET operation it means that the parameters passed on the URL are saved. For a POST operation it means that all of the HTTP headers send during POST are saved,

Action Log

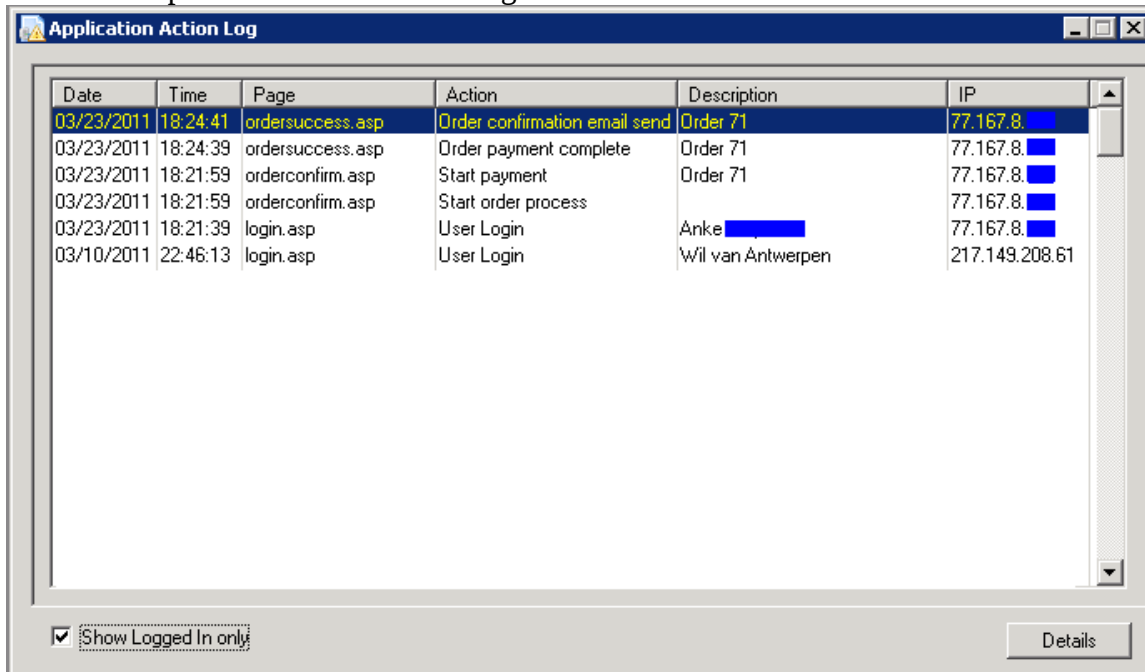


The screenshot shows a window titled "Application Action Log" with a table of application events. The table has six columns: Date, Time, Page, Action, Description, and IP. The first row is highlighted in blue. A "Details" button is located at the bottom right of the window.

Date	Time	Page	Action	Description	IP
08/10/2010	18:23	inv.t.asp	New Product Saved	ACE 4	127.0.0.1
08/10/2010	18:23	inv.t.asp	Product details updated		127.0.0.1
08/10/2010	18:23	login.asp	User Login	JOHN	127.0.0.1
08/10/2010	16:08	login.asp	User Login	JOHN	127.0.0.1
08/10/2010	15:49	login.asp	User Login	JOHN	127.0.0.1
08/06/2010	02:47	login.asp	User Login	JOHN	127.0.0.1
08/06/2010	02:47	login.asp	Login failed	JOHN	127.0.0.1
08/06/2010	02:46	login.asp	Login failed	TEST	127.0.0.1

The action log can be used to log specific state changes – also called an action - in your application. This is very useful as it only logs the state changes that you are interested in. The Actions can be defined by you.

A nice example for that is the following screenshot:



The screenshot shows a window titled "Application Action Log" with a table of application events. The table has six columns: Date, Time, Page, Action, Description, and IP. The first row is highlighted in blue. A checkbox labeled "Show Logged In only" is checked at the bottom left. A "Details" button is located at the bottom right of the window.

Date	Time	Page	Action	Description	IP
03/23/2011	18:24:41	ordersuccess.asp	Order confirmation email send	Order 71	77.167.8. [redacted]
03/23/2011	18:24:39	ordersuccess.asp	Order payment complete	Order 71	77.167.8. [redacted]
03/23/2011	18:21:59	orderconfirm.asp	Start payment	Order 71	77.167.8. [redacted]
03/23/2011	18:21:59	orderconfirm.asp	Start order process		77.167.8. [redacted]
03/23/2011	18:21:39	login.asp	User Login	Anke [redacted]	77.167.8. [redacted]
03/10/2011	22:46:13	login.asp	User Login	Wil van Antwerpen	217.149.208.61

In the above screenshot you see that it is fairly easy to follow an ordering process in the action logs. Please also note that the checkbox to only show the actions for logged in users has been checked.

The screenshot shows a window titled "Action Log Details" with a close button (X) in the top right corner. The window contains several input fields and a checkbox:

- Date: 08/10/2010
- Time: 18:23
- Script Page: invt.asp
- IP Address: 127.0.0.1
- Process ID: 2644
- Action: 0 New Product Saved
- User ID: (empty)
- Logged In: ☒
- Description: ACE-4

Below these fields is a tabbed interface with "Data" and "Custom" tabs. The "Data" tab is selected, and it contains a large empty text area. At the bottom of the window are three buttons: "Previous", "Next", and "Close".

The detail view of the action log.

The Previous and Next buttons allow you to select previous or next lines in the underlying list view. In other words, you can use them to see the next or previous error. You can also use the shortcut keys Ctrl+Up Arrow and Ctrl+Down Arrow to access these buttons.

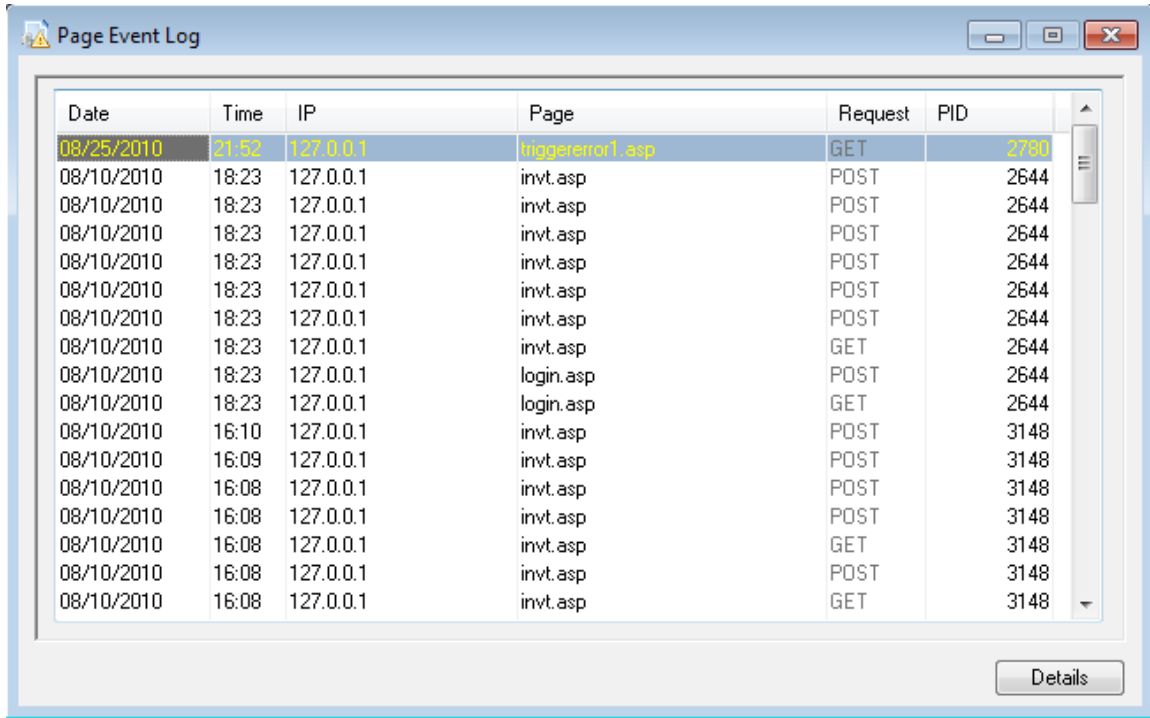
The screenshot shows a window titled "Action Log Details" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a form with the following fields:

Date:	08/10/2010	Script Page:	login.asp
Time:	18:23	IP Address:	127.0.0.1
Process ID:	2644	Action:	9 User Login
User ID:		<input type="checkbox"/> Logged In	
Description:	JOHN		

Below these fields is a tabbed interface with two tabs: "Data" and "Custom". The "Custom" tab is selected, revealing three text input fields labeled "CustomData1:", "CustomData2:", and "CustomData3:". At the bottom right of the window are three buttons: "Previous" (highlighted in blue), "Next", and "Close".

With the CustomData fields you can log up to three customizable data states to your log entry. The description on how-to do this is briefly discussed at the “[Adding User ID and login status to the logs](#)” example.

Page Event



The screenshot shows a window titled "Page Event Log" with a table of log entries. The table has six columns: Date, Time, IP, Page, Request, and PID. The first row is highlighted in blue. The table contains 20 rows of data, showing various page requests and their corresponding timestamps and IP addresses. A "Details" button is located at the bottom right of the window.

Date	Time	IP	Page	Request	PID
08/25/2010	21:52	127.0.0.1	triggerer1.asp	GET	2780
08/10/2010	18:23	127.0.0.1	inv.t.asp	POST	2644
08/10/2010	18:23	127.0.0.1	inv.t.asp	POST	2644
08/10/2010	18:23	127.0.0.1	inv.t.asp	POST	2644
08/10/2010	18:23	127.0.0.1	inv.t.asp	POST	2644
08/10/2010	18:23	127.0.0.1	inv.t.asp	POST	2644
08/10/2010	18:23	127.0.0.1	inv.t.asp	POST	2644
08/10/2010	18:23	127.0.0.1	login.asp	POST	2644
08/10/2010	18:23	127.0.0.1	login.asp	GET	2644
08/10/2010	16:10	127.0.0.1	inv.t.asp	POST	3148
08/10/2010	16:09	127.0.0.1	inv.t.asp	POST	3148
08/10/2010	16:08	127.0.0.1	inv.t.asp	POST	3148
08/10/2010	16:08	127.0.0.1	inv.t.asp	POST	3148
08/10/2010	16:08	127.0.0.1	inv.t.asp	GET	3148
08/10/2010	16:08	127.0.0.1	inv.t.asp	POST	3148
08/10/2010	16:08	127.0.0.1	inv.t.asp	GET	3148

Logs access of end user visits to your pages.

This logs every page that you have added the DoLogPage logic to.

If you are tracing an error, you might want to use this to backtrack what steps the user has exactly taken to trigger the problem. You can do that by matching up date/time/process ID and –if saved- things like User ID/logged in status.

The Previous and Next buttons allow you to select previous or next lines in the underlying list view. In other words, you can use them to see the next or previous error. You can also use the shortcut keys Ctrl+Up Arrow and Ctrl+Down Arrow to access these buttons.

Page Log Details

Date: 08/25/2010 Script Page: triggererror1.asp

Time: 21:52 IP Address: 127.0.0.1

Process ID: 2780 Request Method: G

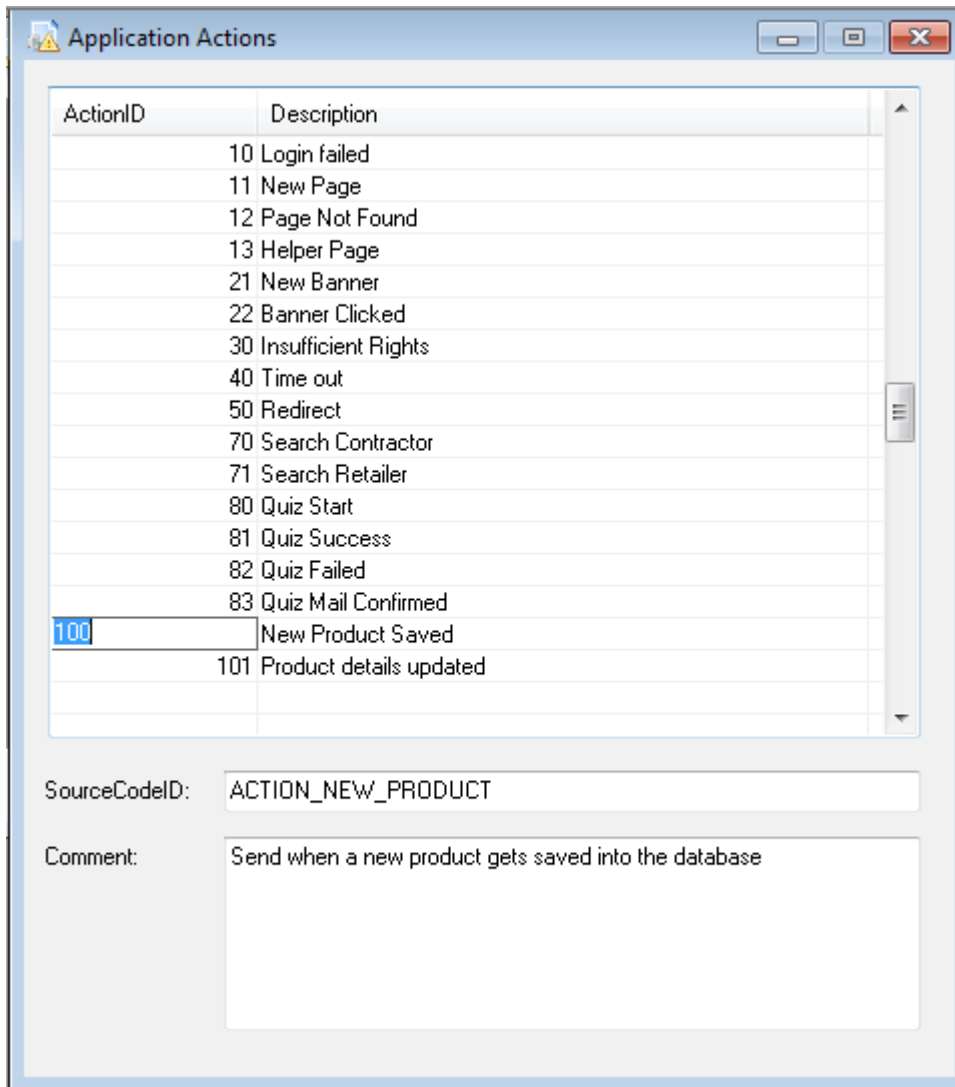
User ID: ☐ Logged In

Description:

Data Custom

Previous Next Close

Application Actions



The screenshot shows a window titled "Application Actions" with a list of actions and a form for adding a new one. The list has two columns: "ActionID" and "Description". The "ActionID" column has a text input field with "100" entered. The "Description" column has a text input field with "New Product Saved" entered. Below the list, there are two text input fields: "SourceCodeID:" with "ACTION_NEW_PRODUCT" and "Comment:" with "Send when a new product gets saved into the database".

ActionID	Description
10	Login failed
11	New Page
12	Page Not Found
13	Helper Page
21	New Banner
22	Banner Clicked
30	Insufficient Rights
40	Time out
50	Redirect
70	Search Contractor
71	Search Retailer
80	Quiz Start
81	Quiz Success
82	Quiz Failed
83	Quiz Mail Confirmed
100	New Product Saved
101	Product details updated

SourceCodeID: ACTION_NEW_PRODUCT

Comment: Send when a new product gets saved into the database

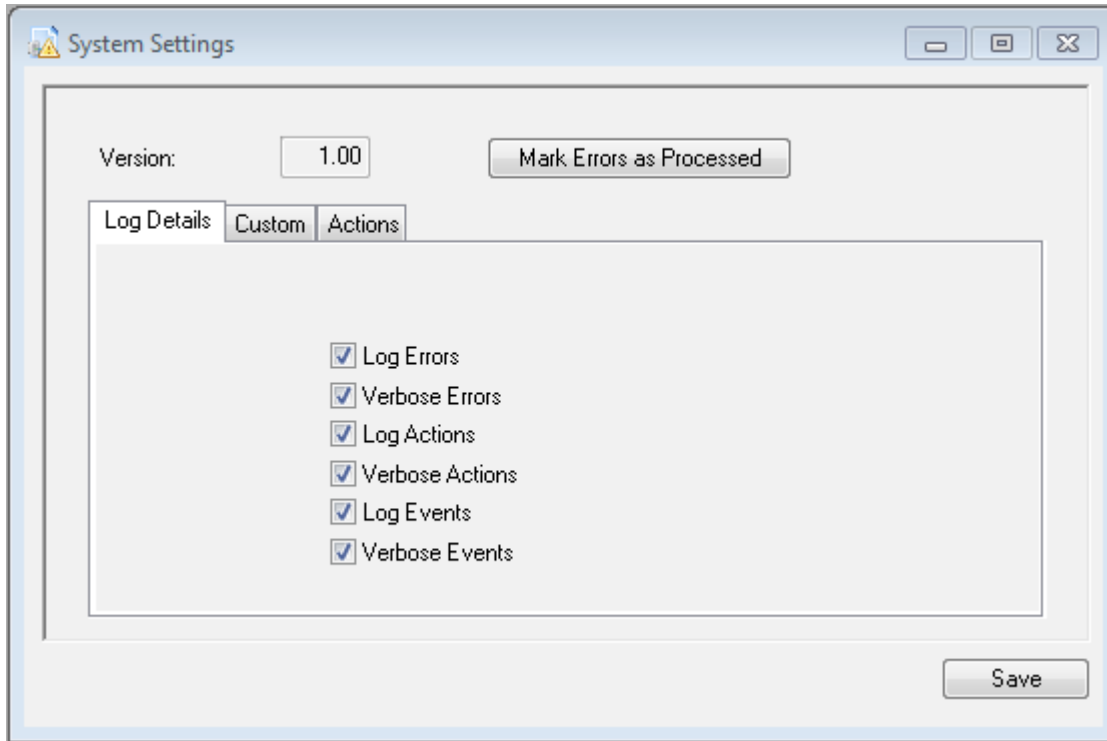
Application Actions are a dynamic feature to allow you to track if a certain event happened in your web application. This comes with several actions preset, but you can add as many actions as you want.

As you need to be able to identify the action in your source code in a clear and concise way, we have come up with the following solution. You can freely define your actions here in this file (Use Action ID 1001 and up to leave room for adding standard actions)

Then define SourceCodeID as a string prefixed with "ACTION_", just as in the example above. We want to log whenever someone creates a new product and have defined a source code constant "ACTION_NEW_PRODUCT". That's the code you will use in your source.

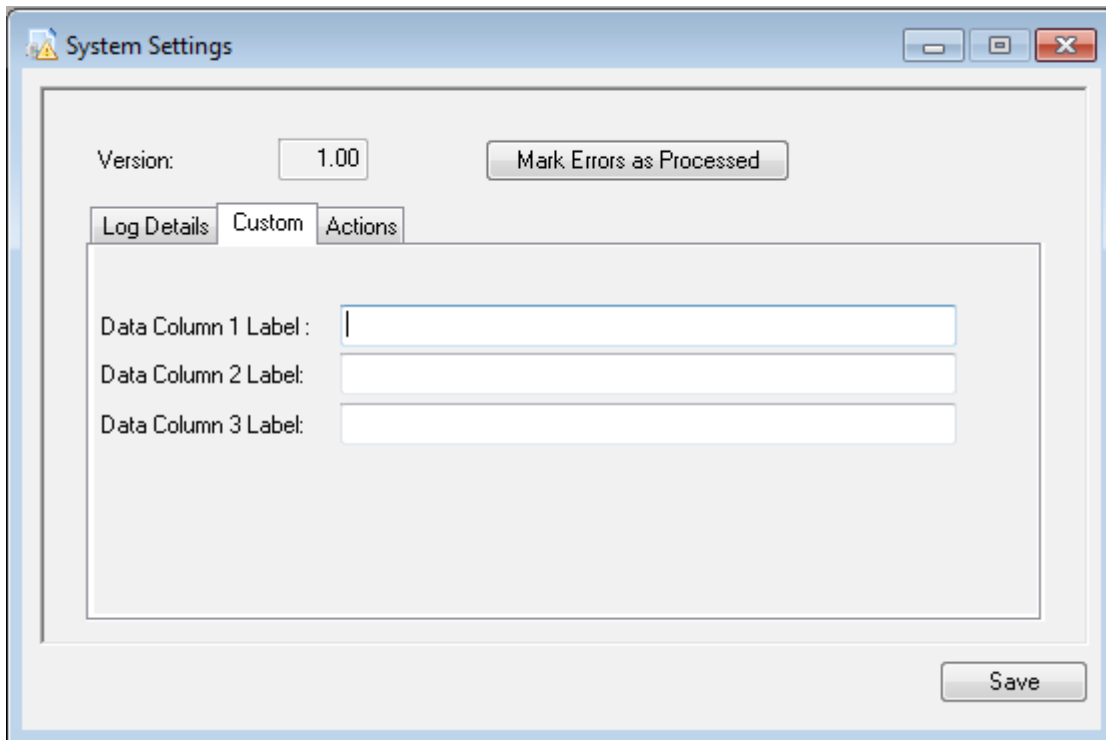
The way that works is that you can generate a new `ActionCodes.h` header file from the system settings Actions tab (see system settings section in the manual)

System Settings

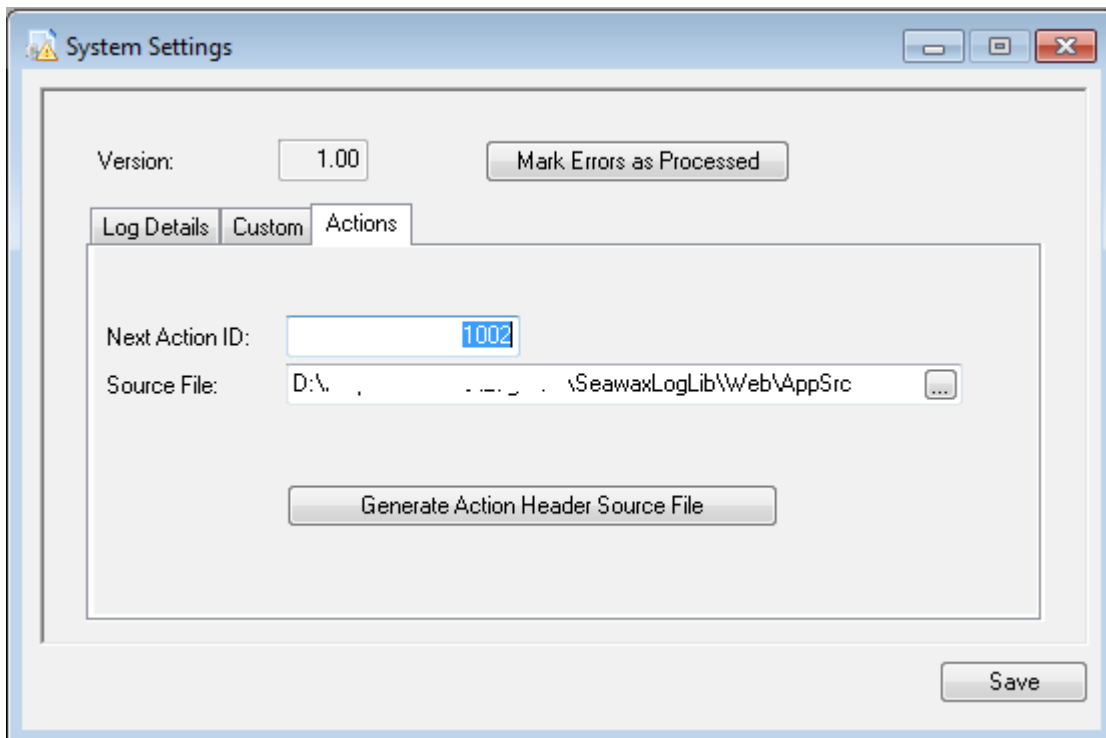


With the system settings dialog you can change certain behavior of the logging in your webapp. For starters you can mark all errors as read, so that if you just have been testing a bit and have collected a lot of errors in your log file that you don't get spammed by a lot of email if you activate your error email notifications.

Another thing you can tweak here is to actually enable/disable any of the logs. Setting the verbosity level to true (checked) has the effect that the HTTP headers are saved during a GET or POST action.



You can set the labels that are shown in the WxLogs application for custom data that you want to be stored with your logs. By setting the labels here, you don't have to remember what CustomData1 actually stands for.

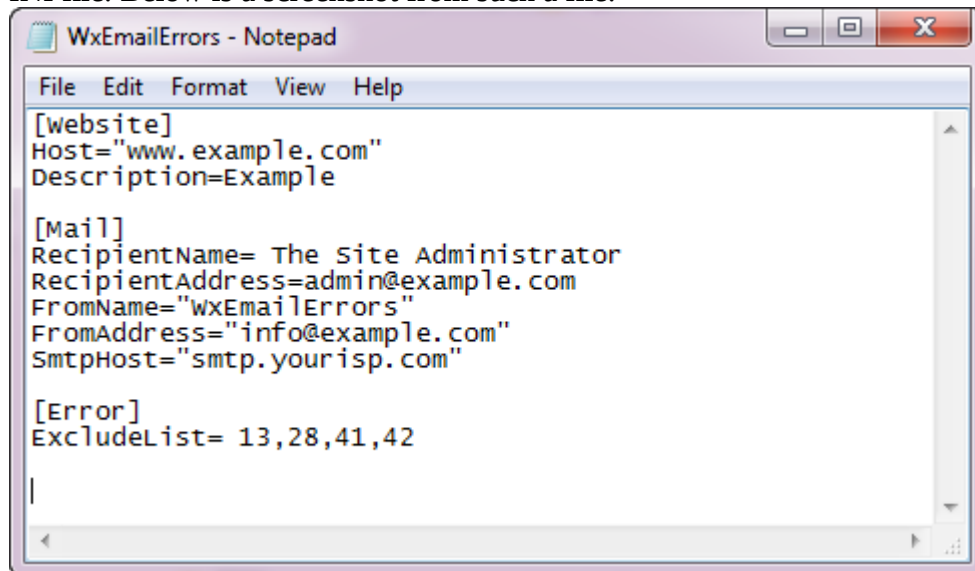


WxEmailErrors

The WxEmailErrors application can be used to automatically send the web site administrator an email with all the technical details whenever an error has been triggered. In order to do this the application has its own embedded smtp send control, which allows you to send the email directly via smtp.

The idea is to run this application on your web host automatically every 10 minutes or so using the standard windows tasks feature. This way if a user triggers an error in your service or webapp, you will get this information automatically by email and take direct action if needed.

As this application has no user interface at all, it is configured via a standard windows INI file. Below is a screenshot from such a file.



Website section

The host field can be used to identify the website for which this is run.
The description field is used as a subject line along with the error message.

Mail section

This section speaks a bit for itself, you set the sender as FromName, FromAddress and the recipient of the automatic email as RecipientName, RecipientAddress.

Error section

The excludelist is a comma separated list that allows you to filter which errors you do not want to see an email from. Some errors trapped by the error handler are not critical, take for example error 13 “An entry is required.. ” error. In pretty much all cases that is an error which is expected and doesn’t need any action..

WxTestEmailConfig

This little program can be run to test the email setup configuration file. When you run this it will read the WxEmailErrors.ini file and gives you an easy means to test if the automated email “service” WxEmailErrors should work.

Adding the code to your workspace

You'll have to add the Wx... data files to your datafolder. If you are using an SQL database backend, my suggestion is to NOT migrate it to SQL as that way you'll be able to also log errors related to connectivity problems.

Add the SeawaxLib Web library to your workspace

Your application needs just two lines of code to get the basics working.

```
Use AllWebAppClasses.pkg
```

```
Use cWxAllWebLogs.pkg
```

```
Object oApplication is a cApplication
```

```
...
```

```
Object oWebApp is a cWebApp
```

```
    Use WxLog.wo
```

```
...
```

By putting the package before the cApplication object, it ensures that the error handler is activated before any data files are opened.

You'll also need to add the following 5 files to your data folder, just unzip the WxLogDatafiles.zip datafiles into your data folder.

You do not need to add the files to the filelist.cfg

Filenumber	FileName	Function	
3062	WxErrLog.dat	Error Log	
3065	WxEvtLog.dat	Visiting pages Log	
3066	WxAppAct.dat	Application Actions	
3067	WxAppLog.dat	Application Log	
3069	WxLogSys.dat	Log System table	

After that the basics work and your errors will automatically be logged in the WxErrLog data file.

```
WebApp.src* x
Use AllWebAppClasses.pkg
Use cWxAllWebLogs.pkg

Object oApplication is a cApplication
  Set psCompany to "Data Access Worldwide"
  Set psProduct to "Visual DataFlex Examples"
  Set psVersion to "15.1"
  Set psProgram to "Seawax Log Sample"
End_Object

Object oWebApp is a cWebApp

  Use WxLog.wo
  Use Customer.wo
  Use CustomerReport.wo
  Use Invt.wo
  Use Login.wo
  Use OrderDtlReport.wo
  Use OrderReport.wo
  Use SalesP.wo
  Use Vendor.wo

  Procedure OnAttachProcess
    Forward Send OnAttachProcess
    // Will only log on session attach if session pooling is enabled.
    Send DoLogPage Of oWxLog "Attach session"
  End_Procedure // DoAttach

  Procedure OnDetachProcess
    Forward Send OnDetachProcess
    // Will only log on session detach if session pooling is enabled.
    Send DoLogPage Of oWxLog "Detach session"
  End_Procedure // DoAttach
```

Class cWxLog

This class allows you to have your webapp log a variety of details to a set of DataFlex data file. This enables you to create auditing, troubleshooting and store visiting log data.

The web object contains handles to call the following methods.

- DoLogPage
- DoLogAction

- DoLogError

There's a global variable called ghoWxLog which holds the object ID of the oWxLog instance and which can be used without having to refer to the oWxLog web object.

DoLogPage can be used to track what is happening. The webobject method is supposed to be called from the top of your page. In its most bare form call it from asp. If you already have an initialize method that you call then you can just call the method from within webapp directly.

You can pass it a description, but that's optional. It will automatically log things such as

- date
- time
- IP address visitor
- asp script page name
- post or get method
- webapp process id (useful when process pooling is enabled)
- description (if passed)

Everything is saved in the WxEvtLog data file.

DoLogAction can be called from anywhere in your application to log a specific action or status in your application. Examples of this are:

- Confirmation Order Email is send
- User Logged in
- Banner is clicked
- ...

The description is again optional the action is a predefined action and will have to be added to the define enumeration list.

The following items will automatically be logged as well:

- date
- time
- IP address visitor
- asp script page name
- webapp process id (useful when process pooling is enabled)
- description (if passed)

Actions are saved in the WxAppLog data file

DoLogError is to be used to add custom messages to your error log. The normal way of doing this would be to call the Error command. The downside of using the Error command is of course that it can display this error in the page. You can call this to add extra info in the WxErrLog data file.

Class cWxWebAppError

Private: You should use the cWxLog class to communicate with the error handler.

This class allows you to have your webapp related errors to be redirected to a DataFlex error logging data file. Doing so enables you to improve troubleshooting (as you can log more related info) and to automate handling the error.

This code is tested and written for Visual DataFlex 15.1

The current version of the webapp errorhandler errorlog data file has the following fields:

- 1) standard line and extended error string
- 2) if it is a datadictionary error:
 - for a recnum table the value in the record identity (RECNUM) and fields and data that make up index.1
 - for a non recnum table the fields that make up primary index and the data in the primary index
- 3) the name of the originating object where the error was triggered
- 4) the datadictionary operation when the error was triggered (save/delete/clear/find)
- 5) Of course date/time when the error happened

What has been added is:

- 1) The ability to filter out (don't report) datadictionary required errors
- 2) Add the .asp page name called when the error happened (not for web services)
- 3) Make vdf15.1 compatible which includes adding stack traces to the log for errors
- 4) Some of the content of the response header object (querystring/referer page/UserAgent/Remote Address/Method type like GET or POST)
- 5) If one identifies users by having them log in to your webapp? Then there's a callback method so you can add the user name into the logs.

Class cWxWebApplicationLog

Private

Used to log events of your web application.

An event is for example a user logging in successfully, a failed login, starting a particular process.

In addition it also logs details such as page accessed, IP number, date time.

Class cWxWebEventLog

Private

Used to log activity of your web application.
It also logs details such as page accessed, IP number, date time.