combit GmbH
Untere Laube 30
78462 Konstanz

Whitepaper

Product: List & Label

Self Help

Table of Contents

Introduction

This whitepaper shall be a brief guide how to find and fix problems. It gives you some hints on how to approach a possibly occurring misbehavior, where to start searching and how to move on further in troubleshooting. However – if you're still in the need of the assistance from our support team, it helps you in making prearrangements to give them as much information as possible.

The major focus lies on debugging. In our case this actually means examining your application during execution. For this purpose our Tool "Debwin3" will assist you. You'll find it in your List & Label installation path in the "Tools" subdirectory. It traces your application and generates a log file containing every action performed by List & Label. As you keep on reading these lines, you will get an overview about the structure and the syntax of the log file. You will also learn how to find problems more quickly by using it.

Procedure for Troubleshooting

If a problem is occurring, it's recommended to follow a certain troubleshooting procedure, here are the steps to take (we're discussing a further description for each later):

1. Check for an up-to-date Service Pack
2. Read Service Pack readme
3. Search online knowledgebase
4. Create a log file
5. Analyze a log file
6. Creating a Dump File in case of crashes
7. Contacting support

Check for an Up-to-date Service Pack

Please first check if you already have installed the latest service pack of List & Label. Most problems reported by users/developers are fixed in service packs. Therefore, it is recommended to always keep List & Label up to date.
There are two ways to find out which version you're using. One is using the designer dialog accessible via a shortcut (the other one is analyzing the log file - which we're getting on later in this article).
Open the List & Label Designer and press <CTRL + SHIFT + F12>. The dialog that opens contains some information about List & Label and your system. The most important information is the version number. You find the current List & Label version and subversion number under "Version". Additionally, the log file shows which modules are loaded.

The following link provides a summary of current service packs available for your registered products:
http://www.combit.net/en/servicepacks.aspx
For each service pack listed on our webpage you will find a 'readme' in form of a PDF file. All changes between the versions are listed there sorted by modules.

Read Service Pack ReadMe

In the service pack readme new features, changes and fixed problems since the previous version are documented.

So please have a closer look at the service pack readme first. Maybe a problem is fixed very quickly and easily by downloading and installing the latest service pack.

Search Online Knowledgebase

If you're already guessing or even knowing what's causing your problem, you can also have a look at our free online knowledgebase. Many articles contain hints and solutions for known and frequently occurring problems. Articles like "Where Do I Find the List & Label Serial Number" or "Barcode objects – limiting bar width" are merely a fraction of what the knowledgebase contains. The search form will take you directly and quickly to these articles. Follow this link to the online knowledgebase:

http://www.combit.net/en/knowledgebase

Debugging of .NET Applications

Problems occurring on the developer PC can be easily found in most cases. The usual features of the development environment can be used to spot a problem relatively quickly. The first step is to catch any occurring exceptions and to find their cause (see section "Error Handling With Exceptions" in the Programmer's Manual).
As a development component List & Label is naturally run under a variety of different constellations on the end user side. To find problems there as easily as possible a dedicated debug tool is available which provides a logging function for problems occurring rarely or only on certain systems so problems can also be examined under systems without a debugger.
Of course the logging function can also be used on the developer PC and provides the possibility to check all calls and return values at a glance as well.

Create Log File

If a problem only occurs on a customer system, the first thing to do is to create a log file. The tool Debwin can be used for this purpose. It can be found in the "Tools" directory of the List & Label installation.
Debwin has to be started before the application. If the application is started afterwards, all calls of the component with their return values as well as additional information about module versions, operation system, etc. will be logged.
Every exception thrown under .NET represents a negative value of a function in the log. There is usually more helpful information in the log.
If the application is supposed to create debug logs without the help of Debwin, this can be done in the configuration file of the application. Logging can be forced as follows:

```
<configuration>
 <appSettings>
  <add key="ListLabel DebugLogFilePath" value="C:\Users\Public\debug.log" />
  <add key="ListLabel EnableDebug" value="1" />
 </appSettings>
</configuration>
```

Custom Logging Mechanisms & Logs in Web Applications

Capturing log messages from List & Label using Debwin and the integrated log file writer are simple and convenient solutions for regular desktop applications.
However, for web applications, Windows services and multi-user systems this approach is of a very limited usefulness: Debwin and the integrated log file capture the log messages from all List & Label instances of a process and there is no separation between the outputs of concurrently running print jobs.
For those scenarios implementing a custom logging mechanism or using one of the popular logging frameworks like NLog or log4net becomes advisable. For that purpose, create a new class deriving from *LoggerBase* or implementing the *ILlLogger* interface directly. Passing such a logger object to the constructor of the ListLabel object will cause all log ouputs of this specific List & Label job to be sent to the specified logger. You may then filter the messages by different levels (Debug, Information, Warning and Error) and categories (e.g. data provider, .NET component, printer information, …).
The included sample project "C# Custom Logger Sample" contains examples of an own logger implementation as well as adapters which connect the prevalent logging frameworks NLog and log4net with the logging interface of List & Label.

Example: Redirecting log output to NLog:

```
ILogger   nlogLogger = NLog.LogManager.GetLogger("MyApp.Reporting");
ILlLogger llLogger = new ListLabel2NLogAdapter(nlogLogger);
ListLabel LL = new ListLabel(llLogger);
```

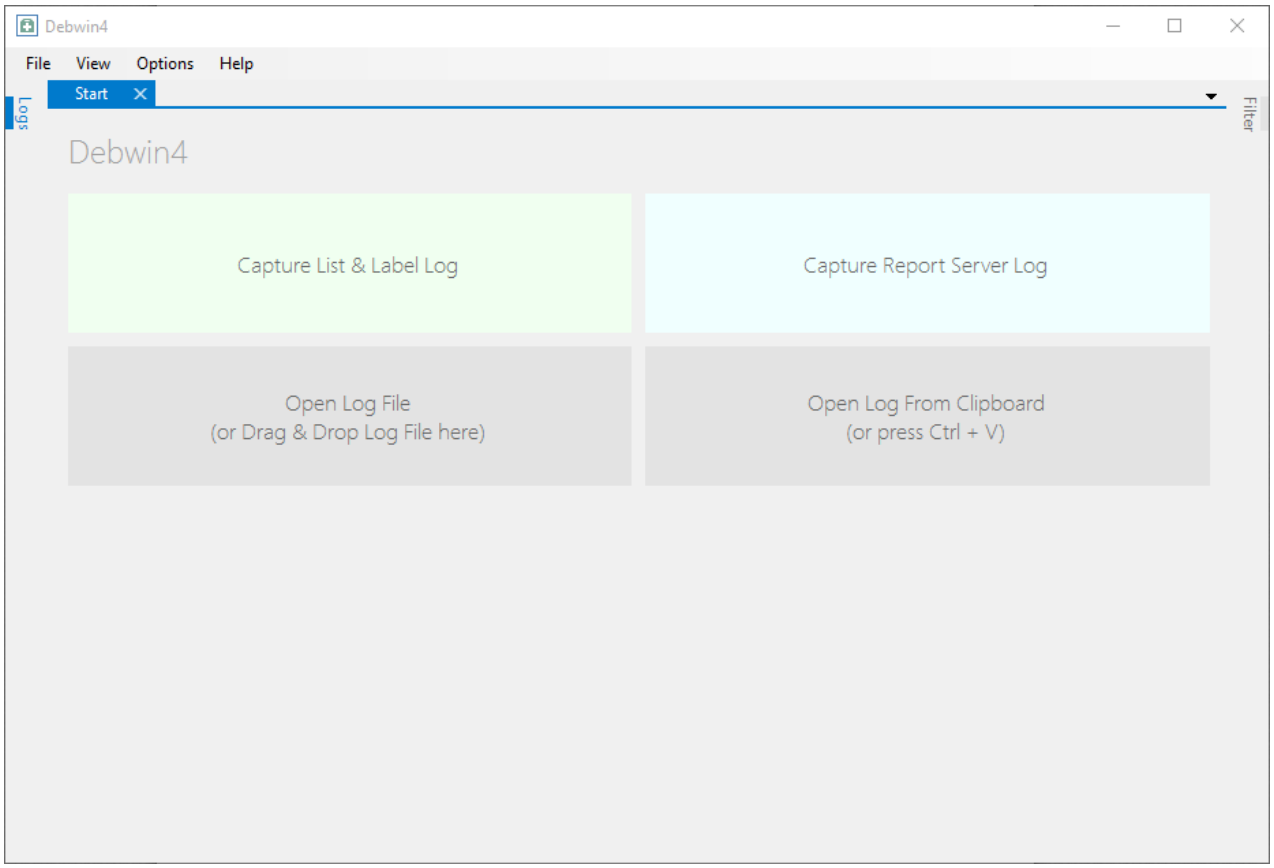
Please note:

- The properties "Debug" and "DebugLogFilePath" of the ListLabel class are ignored as soon as a custom logger is passed to the constructor.
- To avoid a strong decrease in performance, reduce the log outputs in your *ILlLogger* implementation to the minimum using the WantOutput() method.
- Most of the included data providers (optionally) also support an external logger object. Those data providers implement the *ISupportsLogger* interface and provide a SetLogger() method. If a data provider is not assigned a custom logger object, it inherits the logger of the ListLabel object.

Tip for NLog: Often log outputs appear as sudden bursts. Use the AsyncWrapper target of NLog for an async processing of the log messages so List & Label does not have to wait on it.

Creating a Log File With Debwin4

Debugging with Debwin has been completely overhauled and significantly simplified with version 22 of List & Label.



The debug mode must not be enabled via LlSetDebug() anymore. Start Debwin before the application you want to debug. As soon as your application is running, Debwin is processing and displaying the debug output.

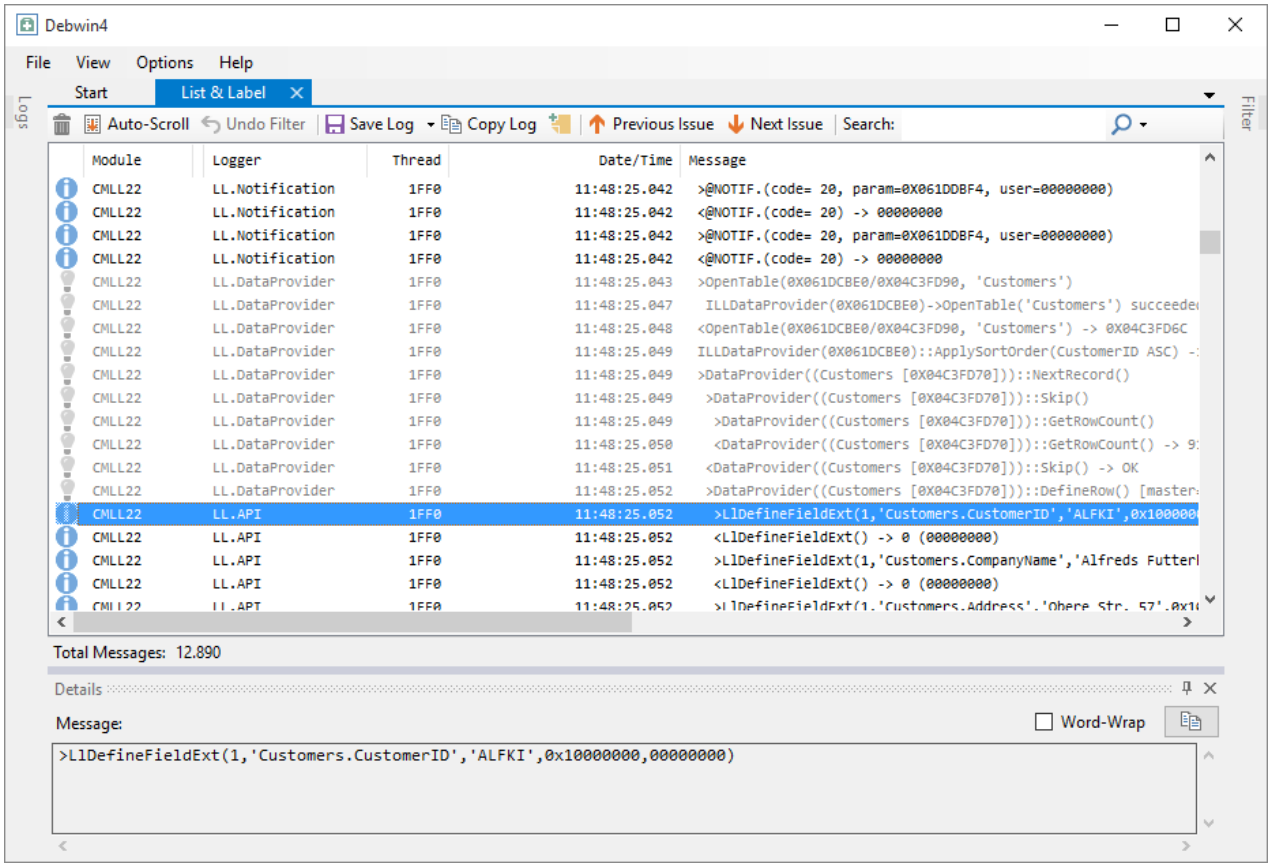
Following options are available:

Capture List & Label Log	Create log file for List & Label
Capture Report Server Log	Create log file for combit Report Server
Open Log File **(or Drag & Drop Log File here)**	Open existing log file
Open Log From Clipboard **(or press Ctrl + V)**	Open log output from the clipboard

When using Debwin4, the procedure familiar from Debwin3 also applies:

1. Start Debwin4.
2. Select Capture List & Label Log.
3. Start the application and try to reproduce the erroneous behavior.

Besides the error codes you get further information, so that in most cases the cause for the erroneous behavior can be found.



You see the called module (CMLL22), the logger, the Thread ID, timing information and the called function with parameters, as well as the return value of the function in the following line:

CMLL22 LL.API 1FF0 11:48:25.052 >LlDefineFieldExt(1, 'Customers.CustomerID', 'ALFKI', 0x10000000, 0000000)

During logging the following functions are available in Debwin4:

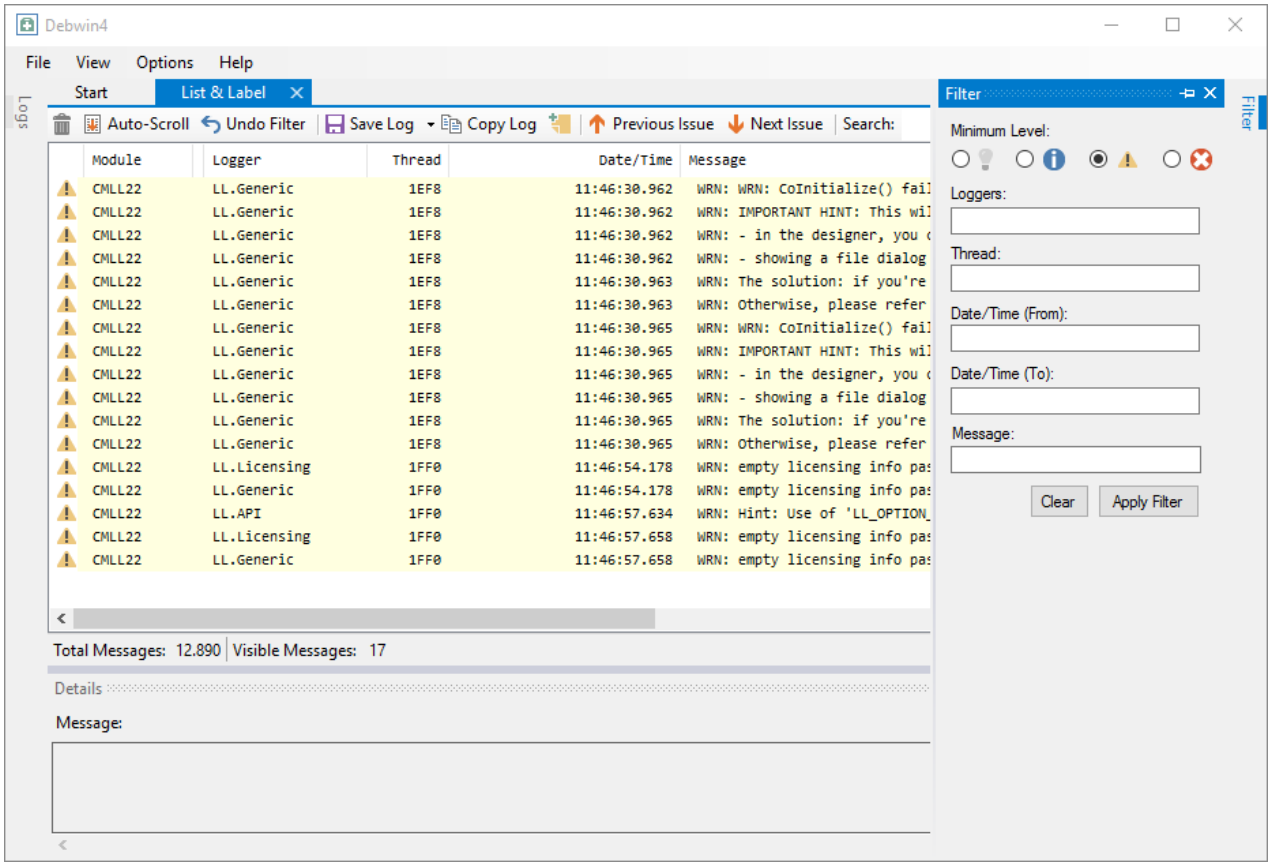
Clear All Messages	Clears all output in the current view.
Auto-Scroll	By activating this option, the focus of the output window is the last and therefore most recent debug output.
Undo Filter	Possible active filters will be reset.
Save Log	Saves the log file in the new log4 format.
Save and Open in Editor	Saves the log file in the log format and opens it in the editor.
Copy Log	Copies the debug output to the clipboard.
Add Custom Message	Adds a note to the log file.
Previous Issue	Jumps to the previous issue in the log file.
Next Issue	Jumps to the next issue in the log file.
Search	Search in the log file.

Filter Log File

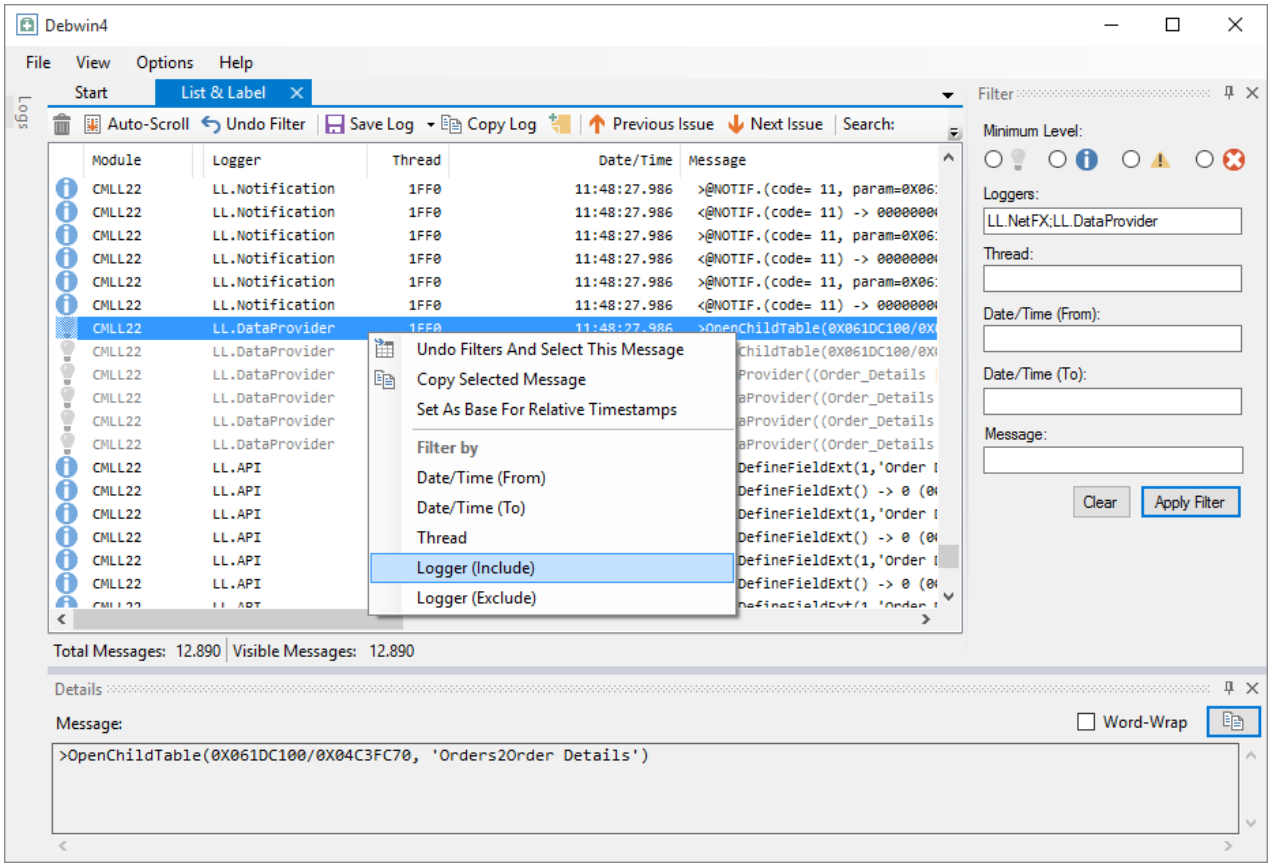
In the Filter window, you quickly can search for specific information or localize specific debug output.

Minimum Level	Defines the priority of the output (Debug output, information, warning or error).

Loggers	Limit the logger (LL.API, LL.DataProvider, LL.Export, LL.Generic, LL.Licensing, LL.NetFX, LL.Notification).
Thread	Limit to a specific thread.
Date/Time (From)	Debug output from a specific date/time.
Date/Time (To)	Debug output to a specific date/time.
Message	Specific debug output.



The filter allows using multiple semicolon-separated values. Additionally the filter criteria can be defined by e.g. a certain thread or different loggers – by using the context menu of the output window. Simply click on the specific line and select the logger to be included:



Hint: The search and the filter "Message" allow using regular expressions.

Creating a Log File for Server/Web Applications

To create log files for server/web applications it is necessary to run Debwin4 as an administrator (context menu "Run as administrator") and to select "Capture List & Label Log". The logging in services and web applications is automatically started.

Create a Log File With Debwin3 and Older

The next step in troubleshooting would be the creation of a log file with the aid of the tool "Debwin3". This tool logs all events from the start of your application on to the completion of a print job. If a problem appears while printing it will be logged by Debwin3. Debwin3 uses a line circular buffer, in which you can scroll down or up at will. This can be done with the common cursor buttons or the scrollbar.

Menu Description

LOGGING

Force Debug Mode: This option forces the debug mode of applications, meaning that you do not have to activate the debug mode in your code using "LlSetDebug()". All DLLs from version 15 on will support that. The default for this feature is set to "False" and the state of the option is not persistent. This behavior is intentional; you have to activate the feature explicitly if you want to create a log file. Please follow this procedure when using the "Force Debug Mode" option:

1. Start Debwin3.
2. Activate "Force Debug Mode".
3. Start the application and reproduce the misbehavior.

Logging active: Dis-/enables the output to the screen and all future output is directed to a log file. In previous versions of Debwin this option was separated as **Log to Window** and **Log to File**.

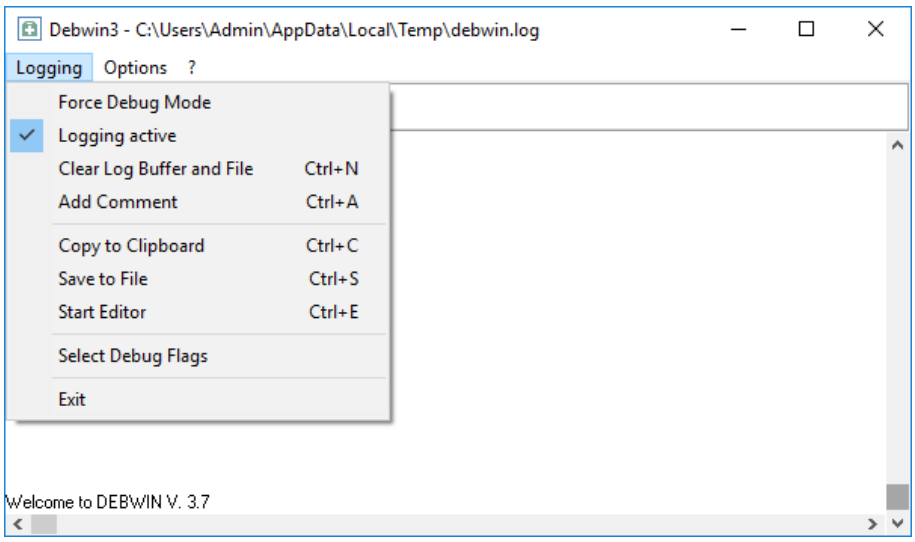
Clear Log Buffer and File: Deletes the log file and line ring buffer for the screen output.

Add Comment: Inserts a comment at the current position.

Copy to Clipboard: Copies the whole buffer into the clipboard.

Save to File: Opens a file selection dialog in which you can select a file name for saving.

Start Editor: Starts the configured editor.

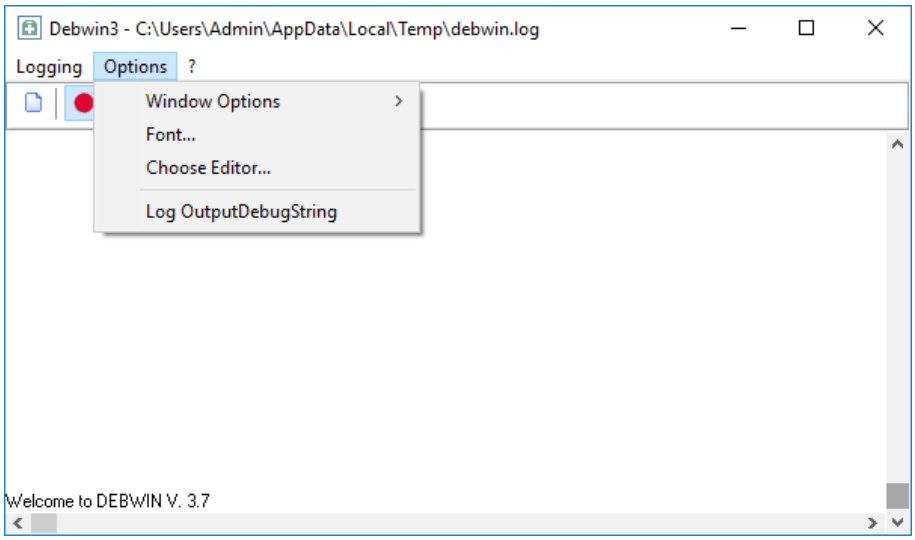


OPTIONS

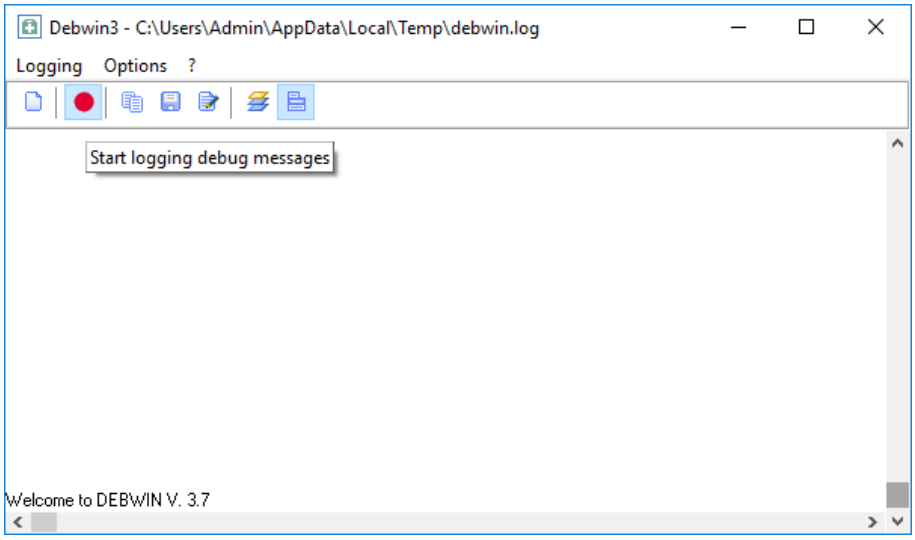
Window Options:

Debug Output Brings Debwin to Front: Debwin is always sent to foreground if new debug output occurs.

Stay on Top: The Debwin window stays always in foreground.
Font: Allows selecting another font.
Choose Editor: Allows selecting an alternative text editor.
Log OutputDebugString: Logs output made via the Windows API "OutputDebugString".



TOOLBAR and TITLEBAR
For an explanation of the toolbar buttons tooltips are shown. The window titlebar displays the name of the log file and the path where it is created.



Creating a Log File in a Local Development Environment

- Start the debug mode **at program start** using the function "LlSetDebug()" (or use the option "Force Debug Mode").

  Using .NET: "LL.Debug = LlDebug.Enabled;"
  Using VCL: "LL.Debug = 1;"

- Start Debwin3 and make sure that logging is activated (Options "Log to Window" and "Log to File". Please follow the instructions in the programmer's reference).
- Start your application and reproduce the steps that led to the problem
- The debug outputs are now evaluated and displayed by Debwin3
- Save the outputs using "Save to File"
- The creation of the log file is now completed

Creating a Log File for a Server-/Web Application

Debug outputs are not possible if you are using List & Label as a web service, because this service is running in another user context than the indicated workstation.

In order to still get debug information, please call "LlSetDebug" as follows:

LlSetDebug(LL_DEBUG_CMBTLL | LL_DEBUG_CMBTLL_LOGTOFILE)

All debug outputs are rerouted to the file "combit.log" in the %APPDATA% directory. You can open this directory easily by typing % APPDATA % into the address line of your explorer. Please note that the usage of List & Label as a service requires corresponding server licenses.
The pipe character (|) denotes the OR-operator. Please use the OR-operator of your programming language.
When using the .NET assembly please take notice of the hints in your programmer's reference.

Analyze a Log File

General Layout of a Log File

All log files created by Debwin3 have the same structure. First, the so called "Header" is created. The header is followed by the actual debug outputs, which contain function calls, return values and possible error codes.

Header

SysInfo of	: 24.07.2011 13:48:39
Application	: ???\COMBIT\LL??\LLDEMO32.EXE [??.000]
List & Label	: ???COMBIT\LL??\REDIST...\CMLL??.DLL [??,0,0,0 (11-08-06 08:13)]
Serial number	: *********
LL flags	: USE_IDP,,DD(0),MCBS(disabled)
User and system name	: ***** on *****
Open jobs	: 1 (in this task)
ACP, OEMCP, LIBCP	: 1252, 850, 1252
Keyboard	: 00000407
Max. RTF version on system	: 0401
OS version	: Windows NT 5.1, build 2600, Service Pack 3, Uniprocessor Free
	:Emulated OS: Windows XP (Professional), (Uniprocessor Free), (x86-32 Processor) v5.1 Build:2600 Service Pack:3
	: Underlying OS: Windows XP (Professional), (Terminal Services in Remote Admin Mode), (Uniprocessor Free), (x86-32 Processor) v5.1 Build:2600 Service Pack:3
Temporaries	:
GetTempPath()	: C:\Documents and Settings\????\Local Settings\Temp\
	(hard disk, 5970456 KB free), R/W check OK
env(TEMP)	: C:\DOCUME~1\???\LOKALE~1\Temp (hard disk, 5970456 KB free)
env(TMP)	: C:\DOCUME~1\???\LOKALE~1\Temp (hard disk, 5970456 KB free)
env(PATH)	: C:\WINDOWS\system32; C:\WINDOWS; C:\WINDOWS\System32\Wbem; C:\Program Files\Microsoft SQL Server\90\Tools\binn\
Printers	: PDFCreator: PDFCreator on PDFCreator:, Status 00000000
	: Microsoft XPS Document Writer: Microsoft XPS Document Writer on XPSPort:, Status 00000000
	: Send To OneNote 2007: Send To Microsoft OneNote Driver on Send To Microsoft OneNote Port:, Status 00000000
Default printer	: PDFCreator

System Information

Paths temporary files

Paths temporary files

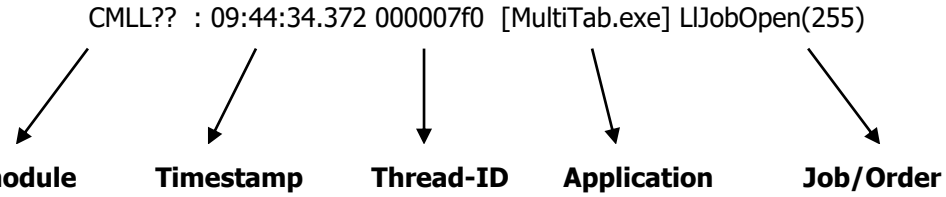
Existing printers

➔ Default printer of the system

System information:	Information about your system Information about your version of List & Label, sub version, service pack
Paths temporary files:	Storage location for temporary files while printing Free space at storage location Read-/write check (R/W check OK)
Existing printers:	Listing of all printers that are available on your system
Default printer:	Currently used default printer

Based on this header information you can get to know your current version of List & Label. **This header should always be contained in log files for the combit support.**

Debug Outputs
As mentioned the header is followed by the actual debug outputs. A line of these debug outputs could look like:



LL module
- Identifies the responsible List & Label module

Timestamp
- Shows the time of execution of the print job
- Is important if e.g. while printing some indefinable pause of approx. 30 seconds is occurring wherein nothing's happening
- Soothe time span may help to find out what's causing this or you may search the log for noticeable long time spans of 30 seconds

Thread-ID
- Identifies the thread and helps to distinguish the different threads

Application
- Names the running application, in this case it is the List & Label sample "MultiTab"

Job/Order
- Identifies the current job, in this case it is to open a project

Error Detection
It is rather easy to locate an error in the log file. Errors correspond to negative return values. So the only thing you have to do is to look for a "=-" in the log file.
It is important to find the first occurrence – if several are listed – of the error, because later entries may just be follow ups and therefore irrelevant in troubleshooting.
You will find a summary of all error codes and their causes in the appendix of your programmer's reference.
Such an entry in the log file could look like:

CMLL?? : 15:33:32.343 000009e8 =-12 (An error occurred during print. Most frequent cause: print spooler is full)

Most important is the output of the value "=-12". This error code is reported by "cmll??.dll". In addition you see the exact time when the problem occurs and the corresponding thread ID. Inside the brackets you can see a description of the error.
The error codes (-996), (-997) and (-998) are not necessarily to be regarded as an error. These few codes are just hints.

CMLL?? : 09:44:38.370 000007f0 =-996 (The table name has changed)
This message indicates that the name of the table has changed.

CULL?? : 11:11:47.263 00000340 =-998 (Present data record did not fit onto the page)
This message indicates that the current data record did not fit onto the page. This return code is needed to report that a new page has to be initialized and to send the current data record one more time.

CMLL?? : 09:48:23.389 00001c21 [MultiTab.exe] =-17 (No preview files found, File is damaged or empty.)
Parameter error, the description is self-explanatory. It is an error while opening the preview.

Checking Return Codes
After calling a function it is advisable to check the return code. So you are guaranteed a correct implementation.
Proper function calls return a value of "0". All return values smaller than zero "-??" are pointing to an error. In .NET so called Exceptions are thrown instead of negative return values. So you are able to catch all possible errors in a catch block.

Creating a Dump File

A dump file can be created with two different tools:

1) Procdump
Please load that utility from http://technet.microsoft.com/en-us/sysinternals/dd996900.aspx

Please start your application and attach ProcDump to the application process. Run the following command line:
procdump -accepteula –e –ma <application>.exe –o %TEMP%\dump.dmp

After that the tool will show some information about the process, the thread etc. Now please reproduce the problem. ProcDump will create the dump file automatically when the exception appears.

2) WinDbg
The following link takes you to an installation guide on how to install the tool on a developer's computer.
http://msdn.microsoft.com/en-us/windows/hardware/gg463009

If you are using Windows XP or Windows Vista, please click on "Get the standalone debugging tools for Windows XP as part of Windows 7 SDK".
If you are using Windows 7 or a newer version, please click on "Get Debugging Tools for Windows (WinDbg) (from the SDK)".

<<
Install Debugging Tools for Windows as a Standalone Component
If you do not want an entire kit WDK or SDK, you can install the Debugging Tools as a standalone component from the Windows SDK. In the installation wizard, select Debugging Tools, and clear other components that you don’t want.
<<

Please use the 32-bit version of WinDbg when debugging a 32-bit process and the 64-bit version for a 64-bit process.

Install the tool (when using Windows 7 or higher: please only tick the box "Debugging Tools for Windows" while installing).
Now start WinDbg via the newly created start menu entry.
Start the debugger and run your application with the debugger with "File > Open executable" (or "File > Attach to Process" if your application is still running). To start the debug process just type "F5" (or "Debug > Go"). "*busy*" should be shown in the command line. Now please run the actions to reproduce the problem respectively the crash of your application. Please type ".dump /ma C:\temp\combit.dmp" in the debugger command line if the problem occurs.

Contacting Support

If you are not able to detect the error, then please contact the combit support team. To get the problem solved quickly it is advisable to give as much information as possible to the support team. Normally, the following information (files) is needed:

1. An exact *description of the error* (Where did the error occur? At which step?). The online form assists you to provide the most important facts. Additionally you may send the *project file* (*.lst, *.lsr etc.) and a preview file (*.ll), if the problem is visible in there.

and

2. Maybe you can reproduce the behavior using our "S*ample Application*" *(*DemoApplication22.exe)? Please send the used printing template (project file), so that the support team can use it to reproduce the behavior. Otherwise a *minimized and executable source code sample* is necessary. You can use one of our examples that come with your List & Label installation or you can minimize your application as long as it can be compiled and executed by the support easily.

If this should not be possible a log file could help in some cases:

3. Complete *log file* (with a **complete header** that shows the print process up to the actual problem)

You get some crashes or exceptions? Maybe this one helps:

4. Dump