Whitepaper

Clarion templates for List & Label

Content

Introduction **5**

Getting Started **5**

Copying List & Label files to their correct locations 5

For all versions of List & Label 5

Registering your templates 6

Using the templates in your Application 6

Importing the txd File into Your Dictionary 6

Adding the List & Label Templates to Your Application (C5.5 and 6) 6

Multi-Dll applications 7

Single exe applications 7

Importing the txa files into your application 7

Add the conditional files to the Report Manager 7

Adding List & Label functionality to a Process or existing Report Template 8

Designing a report layout 8

Concept of List & Label with Clarion Applications **8**

LL_RepArea 8

LL_RepLay 9

Developer V End User **9**

Using the List & Label Templates **9**

The Global Extension Template 9

Setting your Defaults 9
Global Tab 9
Unicode Tab 10
Directories 10

Override 11
Default Options Tab 11
Preview Options Tab 11
User Interface 11
Designing Reports - The Layout Designer Button 12
Files tab 12
Define Manual Fields Tab 13
Optional condition (Condition Opt) 13

Adding List & Label to a Process, Report or Browse Template 13

Files to send tab 13
Select Report Layout tab 13
Manual Fields Tab 14
Binding Fields Tab 14
Page Breaks 14
Suppress Children Tab 15
Printer Selection Tab 15
Progress Options Tab 15
Error Message Tab 16
Options Tab 16
Preview Options Tab 17

Emailing Reports 17

Program Example 18

Using the Handcode Extension 18

Using Debwin - List & Label Debugger 19

Error Codes returned by List & Label 19

Selecting Report Layouts from within code 22

Changing the Default Language 22

Preventing Fields from being sent to List & Label 22

Changing the Name of a Field 23

Changing the Name of a File 24

Printing RTF text from your Application 24

Mailmerging using the Built in RTF editor 25

Using Multiple tables in List & Label's Designer 25

Formatting numbers in List & Label's Designer 26

Formatting Dates and Time in List & Label's Designer 26

Printing from a Queue 27

Introduction

This document is a modified version of the 'getting started' document from Solace. Solace were the original developers of this template which was sold to Radfusion. In 2010 Radfusion moved this template to the public domain under the BSD license.

This template is primarily for use with Clarion 6 ABC programs **only** and may work with Clarion 7 (but hasn't been tested).

The template (as currently modified) allows Clarion to access List & Label up to version 16. Comprehensive testing has **not** been undertaken and you choose to use it at your own risk.

Getting Started

This guide provides step by step instructions on integrating the Clarion ABC Templates into the Clarion environment. To use List & Label with a Clarion application you will need to copy some files into directories on your computer.

Please note versions 7 onwards of List & Label do NOT support 16-bit applications!

Copying List & Label files to their correct locations

There are a couple of files which must be made available to Clarion before an application containing the List & Label templates can be successfully compiled. These are **combit CLW** files that were supplied by combit when you installed List & Label.

Please note that these are not supplied by the Solace Templates.

They can normally be found in the directory C:\Program Files\combit\LL??\Programming Samples and Declarations\Clarion

Throughout this manual, any references to the Clarion version are interchangeable with the current version that you are running. For example, below the table refers to C55\bin. If you are using Clarion 6, then simply change this to Clarion6\bin etc. Please note however that this template has not been tested with Clarion 7 at all.

File Name	File Type	Copy to Location	ABC / Legacy
cmll??.clw	Standard definition file (Unicode from version 15 and later)	C55\libsrc	Both
cull??.clw	Only needed if using Unicode	C55\libsrc	Both

For all versions of List & Label

The files mentioned below will create procedures in your application which manage the List & Label report layout files.

While learning the use of the templates, using these files will bring you up to speed quickly. You don't have to use these procedures but can create your own. The provided procedures however do provide a reasonable starting point to base your own procedures on.

The windows in these procedures may be altered to suit each application's look and feel. As they are application (.txa) and dictionary (.txd) source files you may either copy them into your applications directory, or simply remember where you installed them (which will be where ever you installed your ListAndLabel template).

File Name	Description	ABC / Legacy
LLABC.TXA	>= C5.5 ABC version of procedures	ABC
LISTLABEL.TXD	>= C5.5 Dictionary source	Both

Registering your templates

To use the template it must be registered with your version of Clarion. The description below is for Clarion Versions 5.5 to 6. Registering templates for Clarion 7 requires a different process.

1. Load Clarion, then select the "Setup | Template Registry" pulldown menu option.
2. Click on the "Register" button.
3. Select "\Clarion??\TEMPLATE\ListLabel.TPL". (If all went well, you should now see: "- Class ListAndLabel - List & Label Templates" in your Template registry.)
4. Click on "Close".

Using the templates in your Application

To use the templates you must insert 2 tables into your dictionary and 5 procedures into your application (unless you choose to "Roll your own").

This can be done by importing a TXD file into you dictionary and a TXA file into your application (explained below) OR manually adding the tables into the dictionary and creating your own procedures (sorry no help with any of that at this stage!).

Importing the txd File into Your Dictionary

To import a file, open your dictionary, select "File|Import Text" and select the file ListLabel.txd (C4 upwards) or LLCW20.TXD (CW20 only). These files are located in the "Clarion\3rdparty\examples" directory.

You should then see the two files

LL_RepArea
LL_RepLay

in your dictionary. Close the dictionary and answer "Yes" when asked if you want to save changes.

Adding the List & Label Templates to Your Application (C5.5 and 6)

1. Open the application that you want to add List & Label functionality to.
2. On the procedure tree, click on the Global button.
3. On the Global Properties window click on the Extensions button.
4. Click Insert.
5. Either - Scroll down to Class ListAndLabel - List & Label Templates (ABC templates).
6. Doubleclick on LL_GlobalIncludes - List & Label Globals.
7. Select the version of List & Label that you wish to use. This will compile the correct List & Label libraries into your application.
8. You can ignore the rest of the settings for the moment UNLESS you are not using the default English language version, in which case you will need to change the language to which language you are using. See chapter "Changing the default language".
9. Click on the OK button to return to the Global Properties window.
10. Click on the OK button to return to the Procedures tree.

Multi-Dll applications

If you are compiling a multi-dll application, then you will need to add the extension template to the dll which declares your data files. On this dll, you do not need to switch on the checkbox marked ***L&L Data External (defined in another DLL).***

In all other .dlls, this needs to be switched on. This includes the .exe module. Please note that if a dll does not contain ANY list and Label functionality, then you do not need to add the global extension to that dll.

Single exe applications

For single exe applications, the checkbox marked *L&L Data External (defined in another DLL)* must be switched off.

Importing the txa files into your application

Your List & Label templates include a source files which contains pre-defined procedures which are the Report Layout Management (allows you to add, change, copy and delete report layouts) called **RptLayoutManager** and the Report Options (allows customers to select report layouts at runtime and set options) called **ReportOptions**.

Although these procedures are pre-defined, you may rename them (provided that you alter the necessary values accordingly) and may redesign the screen layouts to suit your own application's look and feel. The only thing that we would ask is that you do not change the parameters or prototypes as these are hard coded into the templates. Changing this will cause the compiler to fail.

To import these procedures, select File|Import Text and select the file from your Clarion\3rdparty\examples directory LLABC.TXA for the ABC templates

The following procedures will then be added:

CopyRptLayout
ReportOptions
RptLayoutManager
UpdateRepLay
SolaceLLPreview

You may now add an option on your menu (or where ever!) which calls the **RptLayoutManager** procedure to start designing your report layouts.

If your application is a multi-dll app, then the best place to import this to, is the .dll that will contain most of your reports.

Add the conditional files to the Report Manager

1. Open the procedure UpdateRepLay and click on the Extensions button.
2. Click on Extension 'Layout Manager button to call end user designer'.
3. On the first tab make sure that the checkbox 'Conditional Use of Files?' is switched on and click on the button 'Files to Conditionally send to List & Label'. Click on Insert and enter the condition: LLL_:AreaRef = 1
4. Then click on the button 'Files to Include' and add all the files which will be used when the Area Ref is 1 (for example if Area 1 is Invoices then add the files which are Invoice related)

5. You may add as many conditions and files as you like.

Adding List & Label functionality to a Process or existing Report Template

1. Add a process procedure that will generate, sort and filter the required data to be sent to List & Label. To this procedure add the List & Label Process Extension Template.
2. The process extension does not automatically send the files that you have declared under 'File(s) to process' section of the file schematic. This is because you may have files that are included in the report but are not declared in the file schematic. Therefore you must declare each file that will be required by List & Label in the extension template.
3. Click on the Button 'Select Files to send to List & Label' and add the files which will be used in this report.
4. On Tab 'Select Report Layout' enter ReportOptions into the Report Layout Procedure and enter 1 into the Report Area Value. This is telling the process to call the procedure ReportOptions before starting the process so that the end user can select the Report layout they are going to use, and also to only show Report layouts for Area 1
5. Compile and run your application

Designing a report layout

When your application is running, run up your Report Layout Manager and add a new layout or layouts. Give each layout a sensible description so that you will remember them later. Please also see the help file on details for using this procedure as a designer.

Run the process procedure that you made, select the report layout and you should (if all goes well), now have the report in the layout that you designed.

Concept of List & Label with Clarion Applications

The List & Label templates have been designed around the following concepts. There are two files which control the management of Report Layouts.

LL_RepArea

This file holds all the areas of Reports that your application will generate. For example, an accounts program may have Invoicing reports, Purchase reports, Customer reports, Supplier reports etc. For each of this areas of reports, you must add a record into LL_RepArea.

You may add records to LL_RepArea by using any file scanning utility that will read the file driver that you are using. This allows the Lists & Label templates to do two things.

1. Firstly, when you or your customer create a new Report Layout, they can be categorized under the particular report type. This in turn means that when that type of report is run, only the report layouts for that area can be selected.
2. Secondly, the templates have a section in the Report Designer button, to conditionally send files to List & Label depending on the file type. This means that you may define which files are relative to which report type and therefore not fill List & Label's variable list with your whole dictionary. You can override this is you wish, but it is not advisable. You may change the file's prefix from LLA_: to whatever you wish. It was named in this way so that importing from a TXD would be (unlikely) to clash with any existing files. HOWEVER, please do not change the name of the file as this is used in the templates to prevent the file from being sent to List & Label.

LL_RepLay

This file holds the actual description of the Layout Report that you will be defining. It also holds the Report 'Type' that List & Label requires. This type can either be '**Card**', '**List**' or '**Label**'. Please see your List & label documentation on definitions for these types.

You may change the file's prefix from LLL_: to whatever you wish. It was named in this way so that importing from a TXD would be (unlikely) to clash with any existing files. HOWEVER, please do not change the name of the file as this is used in the templates to prevent the file from being sent to List & Label.

At runtime, a procedures (normally this is the imported procedure ReportOptions) is called before data is passed to the List & label runtimes to determine which layout the customer wishes to use. The Templates then select the correct format, pass any additional options and file formats to List & label and then spool out the data as generated by the application.

Developer V End User

There is a file called Solace.ini (this will change in later iterations) which, should be placed in the Windows directory. The presence of this file allows the developer to use the procedures that will be given to the customer with enhancements that the customer should not usually have.

If the procedures RptLayoutManager and UpdateReplay detect this file, they will do a number of things:

1. RptLayoutManager: Changes the appearance of the both the browses to include the LLA_:Ref in the 'Areas' browse as well as update buttons and in the Layout browse, a factory default indicator will appear together with the List & Label filename which that record uses.
2. UpdateReplay: A checkbox for the field factory default appears. Allow factory default layouts to be edited or deleted. If the file Solace.ini is not found in the Windows directory, the above fields/browse changes will not appear.

NOTE: Unless you want your customer to have access to these settings, do not ship the file Solace.ini!

Using the List & Label Templates

The Global Extension Template

Setting your Defaults

Please note that these defaults only apply to new procedures only and will not be applied to existing procedures. (These defaults may be left blank if you wish)

Global Tab

Which version of List & Label are you using? The Drop List allows you to choose which version of List & Label is to be used. Please note that this will include the correct files from combit into your project depending on which version you choose.

If you select any version greater than 9, you will need to supply the personal license number supplied to you when you purchased List & Label. This number can be found in the file "C:\Program Files\combit\ LL??\PersonalLicense.txt". Copy and paste the license letters into the template.

Please note that this is not required for programs running on the machine onto which the List & Label development program has been installed.

However, if you try to run the programs on any other machine without this code, then they will not work.

You also need to indicate if you are using the Standard or Professional version. If you are using the Standard version, you will not be allowed to distribute the report layout designer to your end users.

The files that you send out are the same, but the Layout designer will not work on end user's machines.

Unicode Tab
If you wish to use Unicode reports, switch on this checkbox. Note that this will change the files that need to be distributed to your customers. See the app.shp file for a list of files to be shipped.

Directories

Directory for List & Label files: If you wish to store your List & Label project files (i.e. the files that contain the layout definitions etc.) in a directory other than the current directory, then you may enter that directory here.
If you wish to use a variable, then simply prefix it with a ! (i.e. !GLO:DirLL).

Directory for List & Label print preview files: If you wish to use a different directory for creating the print preview files in, then you may enter that directory here.
If you wish to use a variable, then simply prefix it with a ! (i.e. !GLO:DirLL).

It is recommended that you specify a local drive to create the preview files. This is for two reasons:

- On a large report, the preview files can be large (unless you specify that you want compressed storage - see below under the Preview Options Tab) so writing a large amount of data across a network work increase the network traffic and can therefore slow down the printing process.
- If two people try to run the same report at the same time, this can cause problems whereby the preview files may get over written by each other.

Create Directories if they do not exist. Set this option to true (default) to instruct the templates to automatically create a directory for you if one does not already exist.

Override
The override tab allows you to rename the fields or files when they are sent to List & Label or to prevent certain fields from being sent (such as reference variables, passwords etc.). This can also be done via the dictionary.

Default Report Layout Procedure: Selecting this report will automatically populate all new extension templates with the Procedure that can be called to select a report Layout.

When you have imported the .txa into your application, there is a default procedure called ReportOptions which we recommend that you use.

Default Type of Progress: Selecting this option will automatically set the type of progress bar on each new extension template when running reports.

Default Options Tab
These options coincide with the options available for the List & Label command LLSetOption().

Preview Options Tab
Although List & Label has a built in previewer, this can cause problems because Clarion does not always handle windows external to your application very well and can sometimes result in locking your application. For this reason, there is a procedure which can be imported from a txa which can be used instead of the built in List & Label previewer.

This previewer may then be modified to suit your application's look and feel. However, Clarion has now been modified to include better support of external Window handling and since C5, this is no longer an issue and you may use the List & Label internal previewer without worry that it may lock your program.
If you are exporting reports, you may select a variable to set to true, will then run the exported file in the registered program. This will internally generate a ShellExecute() api on the file that you have created. For example PDF files will automatically run ADOBE to view file after printing (provided that ADOBE is installed on the target system.

To use this feature, select a Global variable (defined as a BYTE). Unfortunately, the template language does not allow you to select files from the global extension, so you will need to type in the variable name. If no variable is selected here, then the file will not be previewed!

The ideal place to the set this variable to either True or False would be in the embed Before List and Label Export options.

User Interface
The user interface tab allows you to specify the language module that List & Label will use. See the combit website for a list of currently available languages.

You may also set the following options:

Dialog interface for the designer (i.e. if it has 95 Style buttons, Standard Window buttons or flat buttons), 3D text on buttons and popup tooltips.

Finally, you may specify a name that any manual variables that you define will appear under in the variable list in List & Label designer.

Designing Reports - The Layout Designer Button
The Layout Manager Button Control template controls the actual design phase of List and Label. Normally this would be added to an update procedure which is updating a record in the LL_RepLay file.

Two readymade procedures (RptLayoutManager & UpdateRepLay) are included with the List & Label templates and can be imported into your procedure from the files ListLabel.txa

The Layout Manager Button template is Control template and therefore can be added to you window from the Populate|Control Template section of the Clarion window formatter IDE. It is under:

Class ListAndLabel - List and Label Templates
LL_LayoutManager

(If you are using the default imported procedures, this is the 'Layout Designer' button on the Procedure UpdateRepLay)

After adding a button (or on existing button if importing from the txa file) click on the Actions Tab then on List & Label Options button.

Files tab
This tab allows you to specify which files are to be sent to List & Label with each report.

There are two options here:

1. Conditional Use of Files. By clicking this checkbox on, you may enter the conditions under which each file is sent to List & Label. Click on the button 'Files to Conditionally send to List & Label' and then click on Insert to add a new condition. Normally the condition that you enter will be to specify depending of the value of LLL_:AreaRef
   For example, if you have entered into LLRepArea (see Concepts) that Invoices have a reference of 1 then your condition will read: LLL_:AreaRef = 1
   You may then allocate which files would be used by an invoice report.
   Customers type reports may be LLL_:AreaRef number 5
   Therefore you would add a second condition of LLL_:AreaRef = 5
   And then allocate all the Customer type files to this condition.
   A file can belong to any number of conditions.

2. The second option is to Switch off the Conditional Use of Files checkbox. In this case ALL files in the OTHER FILES section of your file schematic will be send to List & Label for every report layout.

   Send Demo Data - Switch this option on (default) if you wish to be able to see demo data in the layout designer. The demo data will consist of repeated words in the format Filename.Fieldname

   Generate a STOP(Error()) is expression is invalid. - Switch this option on (default) is you want the templates to generate a Stop() if the condition that you have entered is invalid. We recommend that you leave this option on as it will help you to trace bugs.

Define Manual Fields Tab
The manual files allows you to define fields which are to be passed to List & Label but are not defined in the dictionary.

Field Name: Any valid name
Field Type: Pick from the drop down list. These are the types of field that List & Label supports.
Expression: Enter any valid Clarion expression. This will be assigned to List & Label at runtime.
Field Format: This is optional. All formatting can be done within List & Label
For example to send a formatted version of a person's full name:

Field Name: FullName
Field Type: String
Expression: Clip(CUS:SurName) & ', ' & Clip(CUS:FirstName)

Field Format:

Optional condition (Condition Opt)
You may enter a condition here which will limit the availability of the manual field. This is so that your users may only see certain fields under certain conditions. Leave blank if you wish the manual field to always be available. Normally, this will be similar to the file conditions (i.e. LLL_:AreaRef = 3 etc.)

Note: Any field that is added to the Designer and has been used on a report layout MUST be also added to each template that will produce the report. Failing to do this will result in List & Label returning an error -23

Adding List & Label to a Process, Report or Browse Template

The List & Label templates can be added to both Processes, Reports and Browses. Effectively List & Label replaces the Clarion Report Template.

The main difference between a Process Template and a Report Template, is that the Report Template sends each row of a view to the print engine whereas the process template does not.

The List & Label Process and Report Extension templates, 'mimic' the report template in that it intercepts the process and sends the results to List & Label for formatting

To add the List & Label template to a Process, Report or Browse simply click on the Extensions button on the procedure properties screen, and add the List & Label Process Extension.

Files to send tab
The process extension does not automatically send the files that you have declared under 'File(s) to process' section of the file schematic. This is because you may have files which are included in the report but are not declared in the file schematic. Therefore you must declare each file that will be required by List & Label in the extension template.

Click on the Button 'Select Files to send to List & Label' and add the files which will be used in this report.

Select Report Layout tab
Select the Procedure that you will be using to select the Report Layout.

It is advisable to use the Procedure that is supplied with your List & Label Templates as these have the correct parameters set up.

Finally, enter the Reference number for the area of report that you wish to use for this Browse/Report. The number entered here will be mapped to LLA_:Ref

Manual Fields Tab
The manual field section allows you to define fields which are to be passed to List & Label but are not defined in the dictionary.

Field Name: Any valid name
Field Type: Pick from the drop down list. These are the types of field that List & Label supports.
Expression: Enter any valid Clarion expression. This will be assigned to List & Label at runtime.
Field Format: This is optional. All formatting can be done within List & Label

For example to send a formatted version of a person's full name:
Field Name: FullName
Field Type: String
Expression: Clip(CUS:SurName) & ', ' & Clip(CUS:FirstName)
Field Format:

Binding Fields Tab
The List and Label templates use the Clarion EVALUATE() function to process the conditional usage of files that are sent to the List and Label Designer and Report Engines.
For this reason, you must BIND any fields or procedures that you are using in your conditions.

The List and Label templates can automatically BIND and UNBIND these fields or procedures for you but added them to the template.

Please note that if you prefer to BIND and UNBIND your fields and procedures by hand in the embedded code, then this facility is unnecessary.

Please also see the Clarion documentation on EVALUATE() in the Clarion Language reference for the rules on using BINDing fields and procedures.

Page Breaks
The List & Label templates make printing reports very easy. Essentially, an invoice report is a mixture of both Card and List reports.

At the top of the invoice you would normally have the Invoice header with you would print once per page (the Card part of the report) and underneath that the detail lines of the lines purchased (the List part of the report).

However, because it required a table to be present to hold the details, this must be selected as a List type of report.

The templates have a built in function which controls the supply of data to List & Label for this type of report.

On the extension for both Report and Process templates is a checkbox called "Is this an Invoice type of Report?"

By switching this option on, you may then supply the templates with the field in the details file which the templates need to watch to determine when to change to a new page or start a new invoice in the case of batch invoices, when this field value changes.
This field would be the field that you would normally use to relate the details file to the invoice header file. Underneath that you may add any other fields of functions which will generate a page break when they change.

Suppress Children Tab
The very nature of the design of the List & Label templates means that you can have any number of different layouts on each report/process/browse procedure.

However, this then causes a problem when you design a report with related files which work well with List reports where a parent record may have multiple children, but does not work as well on cards or labels where your user may only want to print out one record per parent. What then happens is that a label or card will be printed out for every parent and child record processed in the view.

To get around this problem, you may specify that Child Records are suppressed when printing either Labels or Cards. This is ignored for Lists which print out as expected. If you choose to suppress child records, you must also designate the field that will relates the child records to the parent so that a new Label or Card is only printed when this changes.

Printer Selection Tab
When you design a List & Label format, by default it will create a Printer Definition file (*.lbp - Labels, *.crp - Cards, and *.lsp - Lists) - See the developer documentation - page 13 for more details on these.

However, although you do not need to distribute these files as they will be recreated the first time a report is run or edited, it will be created for whatever the default printer on that user's machine is. This can then cause formatting problems when customer want to run the same report on different printers. For this reason, it is advisable to turn the checkbox option on for 'Create printer file for every report'. This will not slow down your report progress!

You may also wish to redirect to a different printer in code. In order to do this, simply select a variable which will hold the printer name to use. It is your responsibility to ensure that this variable is populated with the correct information. If a variable is selected, but not populated, then the default printer will be used unless the user opts to change it at runtime.

Progress Options Tab
List & label has 6 different types of progress window which you may use whilst the data is being processed. You may select one of these progress windows, or use the standard Clarion type progress window. Please see your List & Label documentation for the different types of progress windows that are available.

If you choose a List & Label type progress window, you may also add some text to be shown in the progress window.
This can be done either statically or dynamically.

- Static text does not change so will be shown whilst the reporting is being generated without changing.
- Dynamic text will be updated every time a record is processed. This can be useful when a large report is being generated and the progress bar is not updating very quickly. It can be made to show record as it is being processed and thereby show that the program is being busy!

If you select Dynamic text, an example would be:
'Now Printing ' & clip(STU:Firstname) & ' ' & Stu:LastName

If you want to display the current page number being printed, this can be done with the List & Label function LLPrintGetCurrentPage. For example

'Now Printing page number ' & LLPrintGetCurrentPage(LL_hJob)

If you choose a progress window that has moving graphics, the templates need to know the correct number of records to be printed. There are a number of options to do this.

1. Let the templates decide. This entails the templates running through the records to be printed prior to starting the print job. This is fine when you only have a 100 or so records to print but any more causes a pause before the printing starts. Leave the prompt blank for this option.
2. Supply your own figures. This can be done in two ways. Ether enter a number into the template (i.e. 50) or if you have another way to calculate the number then you may assign a variable to hold this value. If you wish to use a variable, remember to prefix it with a ! i.e. !Loc:NumberOfRecords

Error Message Tab
List & Label returns errors when opening layouts and these can be trapped and acted upon by the programmer, via the templates. On each template there is an Error Message tab. This tab gives the programmer the option to either display a message or call a procedure with parameters. Please note that any error codes returned by List & Label are held in a local variable called:
LL_Result Long

When using the message option, please remember to enclose any text in quotes. This allows strings to be built such as:
'An error ' & LL_Result & ' has arisen when opening the selected report layout.'
When using a procedure to handle the error, all parameters must be enclosed in brackets ()
You may switch off the error handling option if not required.
Please also note that there are embed points around the code that opens the List & Label Layouts
These are: (These are dependant on the type of Progress display selected)

- LL Before Opening With Box Start
- LL After Opening With Box Start - Preview
- LL After Opening With Box Start - Print
- LL After Opening - Preview Clarion Progress
- LL After Opening - Print Clarion Progress
- After Start List and Label - InitReport Routine

Incidentally, the most common error number is -23. It means that a layout was designed with a variable which is not supplied at runtime. This may be a manual field which has been added at design time and not included in the report/process template or simply that that a field in a file has been deleted or the attribute N0_LL added after the layout was designed. Please also see error codes for a list of the meaning of error codes.

Options Tab
The options tab allows you to use the options as set out in the List & Label Programmer's reference. The templates set most as default already for, but you can alter these as you wish. Refer to the List & Label Programmer's reference for details on each function call and what it does.

Two options that need extra explanation are:

1. List & Label Debug. ONLY turn this on if you are using Debwin to trace bugs. If you leave it switched on all the time it WILL slow down your reports!
2. Wrap Calls to L&L APIs in LockThread(). This is advisable if you have any locking problems. See also the section on the Previewer for an explanation of these problems.

Preview Options Tab

In previous versions of the templates, calling the List & Label previewer could cause certain systems to hang. This is due the Clarion's handling of Windows which are opened 'outside' of Clarion's threading. This procedure, simply contains the List & Label preview OCX (supplied with versions 7, 8 and 9).

In version 2.6 upwards this has been fixed and no problems have since been reported. However, in the unlikely event that your customers still experience lockups when in the previewer, this procedure can be used to overcome the problem. This is only ever likely to be a problem if the previewer loses focus, (i.e. the user switches to another application whilst the previewer is being displayed)

If you decide the use this procedure, you MUST register the OCX on each end user's system. For more details, see the redistribution document that comes with List & Label.
If you are not using this previewer, you do not need to register the OCX.

Please also note that if you are not using version 9 of List & Label, you will need to change the OCX to point at the correct version (i.e. version 8, 7 or 6). This will appear in the drop list on the general tab of the properties under Object Type. I.e. if you are using version 8, you will need to change this to:
combit List&Label8 Viewer Control

Emailing Reports

Please note that email is only available for exported file formats, (i.e. RTF, HTML, PDF etc.) and also only in version 8 or later of List & Label.

'E-mailing an exported report is achieved by setting a built in variable **LLEMailReport** to true and populating a built in queue **LLEMailQueue** with the details of recipients.

If the variable **LLEMailReport** has been set to true, the Queue **LLEMailQueue** has one or more records, you are using v8 of List & Label and outputting to an Export file (not print or preview), then the exported reports will be sent to the recipients in the queue. The Queue is defined globally in your application.

The structure of the Queue is as follows:

```
LLEMailQueue   QUEUE,PRE(LLQ_)
LLMailTo       CString(225)    !email address of the recipient (one per queue record)
LLMailCC       CString(255)    !CC to (one per queue record)
LLMailBCC      CString(255)    !Blind CC to (one per queue record)
END
```

Two extra fields can also be populated. These are:

```
LLSubject      !The text to place on the email Subject line
LLBody         !The Body/Text to place in the body of the email
```

The following code will send the report to 2 recipients and one CC recipient.

Program Example
At the start of your procedure:

```
LLEMailReport = True    !Set flag to send email
Free(LLEMailQueue)      !Remove any previous recipients
LLQ_:LLMailTo = test@email.com'
LLQ_:LLMailCC = 'test@home.com'
LLQ_:LLMailBCC = ''
Add(LLEMailQueue)
LLQ_:LLMailTo = test2@email.com'
LLQ_:LLMailCC = ''
LLQ_:LLMailBCC = ''
Add(LLEMailQueue)
LLSubject = 'List and Label test'
LLBody = 'This is List and Label export test'
```

Using the Handcode Extension

The hand code extension allows the programmer access to all the template functionality in order to create their own procedures rather than using the report or process templates. It simply adds all the required routines to a procedure together with extracting the fields and files from the dictionary in the same way that the standard Report and Process templates do.
The following are brief explanations of what each routine does and when it should be called.

LL_DefineVariables ROUTINE
This routine calls all the field formatting routines which in turn feed List & Label with the fields and data that it requires. Normally, this routine is called before each line of data is send to List & Label

LL_StandardInit ROUTINE
This routine opens up a new List & Label job and sends any options (as defined on the options tab of the templates). Normally this is called by the routine LL_InitReport

LL_InitReport ROUTINE
Calls LL_StandardInit and then sends information to List & Label depending on the type of report selected, the layout files to use, whether a print preview is required, the type of progress bar to use and any text for the progress bar.

PrintOutToLL ROUTINE
This is the actual routine that does the printing. In the routine is some code which automatically handles pages breaks and the sending of fields and manual variables to List & Label. Calling this routine is the equivalent of calling Print(RTP:Detail) in a Clarion report.

SendVarsToListAndLabel Routine
This routine actually sends the data that has been formatted in LL_DefineVariables to List & Label. This is done for every line of data to be printed.

LL_DefineManualVariables ROUTINE
Any manual fields are formatted in this routine

Therefore a simple hand coded program would look something like this:

```
    do LL_InitReport          !Initialise a new report
    If LL_PrinterOptions = true then          !If you want to have the printer options before printing
            LL_Result = LLPrintOptionsDialog(LL_hJob,LL_HWind,LL_Report_Title)
    end
```

```
!**** The statement below must be used before any data is sent to List & Label if the !**** project
is a List otherwise you will have a blank first page!
If LL_ReportType = LL_Project_List then
        LL_Result = llPrint(LL_hJob)
End

Loop
        Next(BRW1::View:Browse)
                If error() then
                        Break
                end
        do PrintOutToLL         !Printout the data from the view including any manual variables.
end

CASE LL_ReportType
OF LL_Project_List
        LL_Result = llPrintFieldsEnd(LL_hJob)
END

LL_Result = llPrintEnd(LL_hJob,0)

IF LL_Action = 2 and LL_RecordPrinted > 0 and LL_Abort = false !Preview
        LL_Result = LLViewerProhibitAction(LL_hJob,115)
        LL_Result = llPreviewDisplay(LL_hJob,LL_Report_Name,LL_Path,LL_hWind)
        LL_Result = llPreviewDeleteFiles(LL_hJob,LL_Report_Name,LL_Path)
END
LLJobClose(LL_hJob)
```

Using Debwin - List & Label Debugger

In some cases, you may have some problems that you cannot work out why something is not working. In these cases, List & Label comes with a Debugger program called Debwin.
This is a program which is run separately in Windows and then records all the calls to List & Label. To use this, you must switch on the List & Label Debug option on the procedure's extension template on the options tab. When you then run your application and run the report, the output can be seen in the Debwin window which can then be saved to disk for inspection.

Error Codes returned by List & Label
-1 BAD_JOBHANDLE A function is called with a job handle not generated with LlJobOpen().

-2 ASK_ACTIVE Only one designer window can be open for each application, you have tried to open a second window (only if hWnd in LlDefineLayout() is NULL).

-3 BAD_OBJECTTYPE An invalid type was passed to a function which requires the type of object as a parameter. Valid types: LL_PROJECT_LABEL, LL_PROJECT_LIST, LL_PROJECT_CARD

-4 PRINTING_JOB A print function was called although no print job had been started.

-5 NO_BOX LlPrintSetBoxText() was called and the print job was not opened with LlPrintWithBoxStart().

-6 ALREADY_PRINTING LlPrintWithBoxStart() is not multitasking capable (it makes no sense to run several print jobs parallel, as Windows does not support the ABORTPROC in the printer driver (multitasking)).

-7 NOT_YET_PRINTING LlPrint[G|S]etOption[String](), LlPrintResetProjectState(). The print job has not started yet.

-10 NO_PROJECT LlPrint[WithBox]Start(): there is no object with the given object name. Identical to LL_ERR_NO_OBJECT

-11 NO_PRINTER LlPrint[WithBox]Start(): Printer job could not be started as no printer device could be opened.

-12 PRINTING An error occurred during the print.

-13 ONLY_ONE_JOB An application can only open one job.

-14 NEEDS_VB This DLL-version requires Visual Basic.

-15 BAD_PRINTER LlPrintOptionsDialogTitle(): no printer available.

-16 NO_PREVIEWMODE Preview functions: no preview-mode is set.

-17 NO_PREVIEWFILES LlPreviewDisplay(): no preview-files found.

-18 PARAMETER NULL pointer as a parameter is not allowed, possibly also other parameter errors. Please use the debug mode to determine the error.

-19 BAD_EXPRESSION New expression mode: an expression in LlExprEvaluate() could not be interpreted.

-20 BAD_EXPRMODE Unknown expression-mode in LlSetOption().

-22 CFGNOTFOUND LlPrint[WithBox]Start(): Project file not found.

-23 Bad EXPRESSION LlPrint[WithBox]Start(): One of the Expressions used has an error. Use LlExprError() to find the error. Normally this is caused because a layout was designed with a variable which is not supplied at runtime. This may well be a manual field which has been added at design time and not included in the report/process template or simply that that a field in a file has been deleted or the attribute N0_LL added after the layout was designed

-24 CFGBADFILELlPrint[WithBox]Start():Project file has the wrong format or is defective.

-25 BADOBJNAME LlPrintEnableObject(): the object name is not correct.

-27 UNKNOWNOBJECT LlPrintEnableObject(): no object with this object name exists.

-28 NO_TABLEOBJECT LlPrint[WithBox]Start(): No table in the list mode available.

-29 NO_OBJECT LlPrint[WithBox]Start(): the project has no objects, and empty pages can be printed in another way!

-30 NO_TEXTOBJECT LlPrintGetTextCharsPrinted(): No text object in this project.

-31 UNKNOWN LlPrintIsVariableUsed(), LlPrintIsFieldUsed(): the given variable does not exist.

-32 BAD_MODE Field functions were used although the project is not a list project.

-33 CFGBADMODE LlPrint[WithBox]Start(), LlDefineLayout(): the expression mode of the project file the new mode, but the old mode is set (See LlSetOption()).

-34 ONLYWITHONETABLE This error code can only be returned if - in the list mode - the OneTable mode (LL_OPTION_ONLYONETABLE) is chosen, but more than one table is present when loading the project with LlPrint[WithBox]Start()(See LlSetOption()).

-35 UNKNOWNVARIABLE The variable given with LlGetVariableType() or LlGetVariableContents() was not defined.

-36 UNKNOWNFIELD The field given with LlGetFieldType() or LlGetFieldContents() was not defined.

-37 UNKNOWNSORTORDER The sorting order given by the ID for the grouping functions was not defined.

-38 NOPRINTERCFG LlPrintCopyPrinterConfiguration(): file not found or wrong format

-39 SAVEPRINTERCFG LlPrintCopyPrinterConfiguration(): File write error (network rights allow writing/creation, disk full?)

-40 RESERVED function not implemented yet

-41 NOVALIDPAGES 16 bit StgAPI tries to read a 32 bit file

-42 NOTINHOSTPRINTERMODE This command cannot be called in HOSTPRINTER mode (LlSetPrinterInPrinterFile(), LlPrintCopyPrinterConfiguration())

-43 NOTFINISHED One or more objects have not been completely printed.

-44 BUFFERTOOSMALL Ll[G|S]etOptionString(),LlPrint[G|S]etOptionString(), ... A buffer passed to List & Label is not large enough for the data that should be stored in it. The data is not complete.

-45 BADCODEPAGE LL_OPTION_CODEPAGE: The code page is invalid (NLS not installed).

-46 CANNOTCREATETEMPFILE A temporary file could not be created

-47 NODESTINATION List & Label has no valid print medium (see LL_OPTIONSTRING_EXPORTS_ALLOWED)

-99 USER_ABORTED The user aborted the print.

-100 BAD_DLLS The DLLs required by List & Label are not on the required level.

-101 NO_LANG_DLL The required language-DLL was not found and a CM??L8@@.LNG is not available.

-102 NO_MEMORY Insufficient memory to run the report.

-104 EXCEPTION A GPF happened inside a List & Label call. List & Label might be unstable.

-998 LL_WRN_REPEAT_DATA This is just a hint: present data record did not fit onto the page. This return value is required to remind the programmer that he can, for example, bring the number of pages up to date. The record pointer may not be moved.

Selecting Report Layouts from within code

If you wish to call reports directly, i.e. without the user having to select the report layout each time they run a report, you may specify this on the Process and Report extensions on the Select Report Layout tab.

On the radio button at the top of the report select Use Variable to Determine Default

You must then supply a variable which will be used by the templates to look up the field LLL_:Ref in LL_Replay. For this reason the field that you use must be LONG. Please note that it is you to the programmer to ensure that this field is populated before the Report is called.

In a similar fashion, you must also supply variables (BYTE) which will be used to determine whether a print preview is required or if the printer options need to be displayed before a report is generated. These fields will contain the values of either True or False

Changing the Default Language

On the global extension template, select the last option (User Interface) and from the drop down list select the language that you are using. Please note that List & Label comes as standard as in either English or German.

If you do not have either of these versions and wish to use List & Label in any other of the listed languages, you will need to purchase an additional language kit. These can be purchased from combit at http://www.combit.net

List & Label expects dates to supplied to it as Julian format dates, that is to say the number of days since passed since January 1st -4713. As this does not match the way in which Clarion dates are stored, the templates automatically offset dates before passing them to List & Label by 2378857 days.

If however, you store dates in a different way to Clarion's format, you may override this offset with your own.

The List & Label templates use the User Option section of the Field Options Tab in the Dictionary to allow the programmer to override the automatic facilities of the templates. If you wish to specify a new name for a field in the List & Label designer, simply add the line

DateOffSet_LL(NumberOfDaysToOffSet)

to the User Option for each date field that you wish to override the offset.

Preventing Fields from being sent to List & Label

Certain fields do not benefit from being available to an end user. For example, it is common practice to relate fields via a unique primary key which is hidden from users when they run your application. This field is often a Long field with is auto numbered.

The advantage of using this method to uniquely identify records, is that the user does not have access to alter its value. Therefore, when used as the linking field in a relationship, the programmer knows that there is no need to generate cascading update code for the relationship. However, there is therefore little use for the end user to be able to report on this field as they have no access to it elsewhere. Indeed, often having this field could cause more confusion than would be to omit it.

The List & Label templates use the User Option section of the Field Options Tab in the Dictionary to allow the programmer to override the automatic facilities of the templates. If you wish to prevent a field from being available in the List & Label designer, simply add the line
NO_LL=1
to the User Option for each field that you wish to suppress.

Changing the Name of a Field

When fields are sent to the List & Label designer, the actual field name is used. The reasoning behind this, is that once field is defined for a List & Label report, it expects the same field name to be supplied at runtime.

It is more likely that a programmer may want to change a prompt (due to spelling error etc.) than the name of a field. If this was done between versions and a customer had used that field on a report layout, they will get an error and the report will fail to run.

However, not all field names are descriptive enough to be readily identifiable to a customer of what they contain. The List & Label templates use the User Option section of the Field Options Tab in the Dictionary to allow the programmer to override the automatic facilities of the templates. If you wish to specify a new name for a field in the List & Label designer, there are two ways that this can be done.

The first way is to go to the global extension template and then to the Override Tab. The first option instructs the templates to use the description field in the dictionary for each field if one exists. In the box underneath you can then specify the fields that you wish to override.

You will encounter a 'strange' concept here which is a bit alien to the way that you would normally select fields in a Clarion template. You need to firstly select the field that you will be selecting a field from. The reason for this is simple! Unfortunately, the Clarion template language does not allow fields to be selected from global templates. This way gets around that limitation. The alternative would be that you would have needed to enter the field names and prefixes manually, without any verification!

Once you have selected the file and field that you wish to override, simply set the radio button to either:

Do not send to List & Label (In which case the field will not be selectable in the List & Label designer); or

Rename (In which case the field will appear in the List & Label designer with the name that you use below)

If you are renaming the field, you will then activate the Rename To entry box where you can enter how you wish your users to see the field in the designer. Please note that when renaming a field, do NOT include the file prefix. If you do, the prefix will be displayed in the designer.

.g. Filename.CLI:Renamed_Field. When renaming fields, you may enter spaces, but these will be translated into underscores '_' when sent to list & Label.

The other way to achieve this is to simply add the line

Name_LL(FieldNameToUse)

to the User Option for each field that you wish to rename.Please note that you should NOT enclose the name in quotes. For example if a field in your file is called DOB, and you wish to make the field content more obvious, then you could use

Name_LL(DateOfBirth)

in the field's user options.

NOTE: For this version of the Templates please ensure that if you have more than one option in the User Options, that this setting is entered first. *This second option has now been superseded by the global extension and is supplied for backwards compatibility.*

Changing the Name of a File

When file are sent to the List & Label designer, the actual file name is used.
However, not all file names are descriptive enough to be readily identifiable to a customer of what they contain. If you wish to specify a new name for a file in the List & Label designer, there are two ways to do this

The first way is to go to the global extension template and then to the Override Tab. In the bottom listbox you can then specify the files that you wish to rename. These will then appear in the List & Label designer with the new name.

The second method is to simply add the line

Name_LL(FileNameToUse)

in the User Option section of the File Options Tab in the Dictionary for each file that you wish to rename.

Please note that you should NOT enclose the name in quotes

For example if a file in your dictionary is called ArchInvHead, and you wish to make the file content more obvious, then you could use

Name_LL(ArchiveInvoiceHeader)

in the file's user options.

NOTE: For this version of the Templates please ensure that if you have more than one option in the User Options, that this setting is entered first.

Printing RTF text from your Application

As well as its own built in RTF editor, List & Label can also print out RTF text that is stored in your Clarion application.
To mark a variable (normally a memo field) as containing RTF formatted text, add the line

LL_RTF=1

to the User Option in the dictionary for each field that you wish to mark as containing RTF text.

When in the List & Label designer, you will the see the RTF icon next to that field in the list of variables. This field can then simply be added to the report worksheet and resized accordingly. No other action is required within the designer.

NOTE
When using either the report template, process or browse to send data to List & Label, a memo field will need to be added to the templates 'hot fields'. This is because a memo field is not included inside a record structure and therefore will not automatically be retrieved by the view.

Mailmerging using the Built in RTF editor

Open a layout.
Click on the RTF object button (down on left hand vertical toolbar) and draw a RTF object on the layout.
Double click on the RTF object to bring up the RTF editor.
Now simply enter your text and then drag and drop the variables from the variable list onto the RTF editor.
That's it. When you run the report, the fields will be changed for your actual data.

Using Multiple tables in List & Label's Designer

On occasions, you may find that you need to have two tables on a report which are not necessarily related to each other. The templates do not support this feature of List & Label directly, but it can be achieved with a small amount of hand code. This is what you need to do:

Add your tables to the layout and then name them. To name a table, simply right click on the table, and select **Name...** from the popup menu. Enter a name for this table. Do the same with any other tables on your layout. For this example we will have two tables called **Orders** and **Products** In your code, you then need to enable and disable the tables as you write to each one. To do this, you will need to create a local variable with which to pass the name of the table that you wish to enable/disable. This is because the command LLPrintEnableObject() requires that the value is passed by reference and therefore must be a variable. In this example, I have created a variable called

```
L:Table CString(31)
```

to hold the table name. Then in your code you can disable the Orders table from receiving the data by:

```
L:Table = ':Orders'
LL_Result = LLPrintEnableObject(LL_hJob,L:Table,False)        !Disable Orders table
```

and enable the Products table...

```
L:Table = ':Products'
LL_Result = LLPrintEnableObject(LL_hJob,L:Table,True)         !Enable the Products table
```

Having done this you could then loop around the Orders file (or whatever file was being used to populate the Orders table) and call the routine **PrintOutToLL** to do the actual printing.

So the final code would look something like this:

```
L:Table = ':Orders'
LL_Result = LLPrintEnableObject(LL_hJob,L:Table,False)        !Disable Orders table
L:Table = ':Products'
LL_Result = LLPrintEnableObject(LL_hJob,L:Table,True)         !Enable Products table
Set(Products)
Loop Until Access:Products.Next()
```

```
            do PrintOutToLL
    end
```

In the above code, the table named 'Orders' will be disabled and will not receive any output. After printing to the Products table, you could then invert the tables and print to the Orders table.

NOTE: the ':' at the front of the name when being assigned to L:Table. This is not a typo, but is required by List & Label.

Formatting numbers in List & Label's Designer

When you place (or drag and drop) a number field onto a worksheet in List & Label it will default as a decimal with 2 decimal places. Obviously, this is not always the required format. To modify this you need to use the FStr$() function.

To use this double click on the field on the worksheet and then double click on the item again when the Text properties box appears. This will then allow you to edit the field. Click on the FStr$() tab at the top of the screen and select one of the built in formats that closely represents the format that you require.

You do not have to match exactly the format that you need as you can then modify the standard one to suit yourself.

i.e. the built in format

```
FStr$(File.Field,"#####&.##")
```

will display a number with two decimal points. You can simply edit this format to remove the .## at the end

```
FStr$(File.Field,"#####&")
```

so that it shows an integer.

Formatting Dates and Time in List & Label's Designer

Formatting dates in List & Label's designer works in a very similar way to numbers, except that you use the Date$() function to format the date. Again you may modify the format using the different symbols that List & Label provides to construct your own format if the one that you require is not available. i.e.

```
%D = Day of Week in Words (Monday, Tuesday etc.)
%d = Day of month (1, 2 etc.)
%M =Month in Words (January, February etc.)
%y = Year (2001 etc.)
```

Time fields are also formatted using the Date$ function. The templates convert the standard Clarion time structure to one that is understood by List & Label (in this case fractions of a day e.g. 12 Noon = 0.5) When formatting times, simply use the time symbols:

```
Date$(File.Timefield,"%02h:%02i:%02s %PM")
```

The symbols for time are:

%h = Hours
%i = Minutes (don't ask!)
%s = Seconds
%PM = AM/PM

Therefore to format the time 13:20 you would use:

```
Date$(File.Timefield,"%02h:%02i")
```

Printing from a Queue

Printing from a Queue rather than from a file is quite straightforward. Firstly, you will need to add the fields from your queue to the manual fields section that you want your users to be able to report on.

Remember to do this both in the procedure that will be generating the report as well as UpdateRepLay. Then, when you want to print your report, simply add code such as:

```
Loop QRecs# = 1 to Records(MyQueue)
    Get(MyQueue,Qrecs#)
    do PrintOutToLL
end
```

The routine PrintOutToLL handles all the data conversion etc. for you so you simply need to ensure that the data buffers are filled with the correct data.