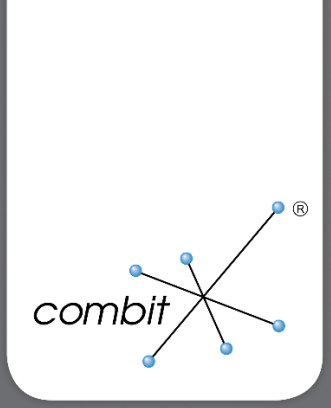


Library for DataFlex V25.0

Created by Bernhard Ponemayr

combit List & Label

Content

How to Use the Library

The library can be used with (Visual) DataFlex 12.0 and higher, .sws files for any (Visual) DataFlex version are included.

Inside the VDF Studio select “Tools > Maintain Libraries…” then “Add Library”, navigate to the installation folder and select the correct .sws depending on your (Visual) DataFlex version (i.e. LL25-VDF19.0.sws for DataFlex 19.0).

If you prefer not to automatically update the library with Service Packs from combit, copy the complete folder LLLibrary25.0 to your preferred library folder and add the copied library to your workspace.

After you have added the library there are two new report templates available in the Create new-View/Report tab window. Create a new Report View based on one of these two templates ("List & Label Report View" or "List & Label Report View with Preview") and add your DataDictionaries to the Report View. For a quick start, this is all you have to do. Compile your project, open the Report View, select the needed report type (List/Label/Card), hit the "Edit Layout" button, provide a filename for your layout and start creating your first List & Label report with the Designer.

The enclosed sample program is designed to run with Visual DataFlex 16.0 and higher; take care to select the correct .sws when opening the workspace. Compile the project Order.Src, all reports and the "Print order" button located in the Order View are using List & Label reports. If the application hangs on startup, you have to re-index all tables because you use a different Collate Sequence. Additionally, there are some features added to the Report Views to show you some (but really not all) features you can use with List & Label. Sample layouts are also included, just play around with the program.

If you do not want to use the library, it is possible to use the standalone LL25.pkg just like in prior versions of the class. In this case, copy the LL25.pkg and the cListLabel.pkg from the LLLibrary25.0\AppSrc Folder into the AppSrc Folder of your workspace. It is recommended to use the library.

General Notes

Not all List & Label functions are already implemented.

You may use the library on an "as is" basis on your own responsibility. I cannot give E-Mail support, on problems contact combit or post at DAW's Newsgroups.

Take care to understand the difference between List & Label fields/variables and DataFlex fields/variables. In List & Label variables are values that are printed (and also evaluated) once per page. Fields are values that are printed (and evaluated) more than once per page. Fields are only available in List-projects, where more than one record can be printed on one page. If you define fields in a project other than a list, they are translated to "Variables" automatically.

More Information can be found in the cListLabel.pkg file and in the enclosed Sample.

Notes for Updating From a Previous Revision

If you have already working applications that should now use List & Label 25 you just have to change all of the following lines in your source code

```
use LLxx.pkg // xx is the old Revision i.e. 8, 9, 10, 11, 12, 13, 14, 15
```

to

```
use cListLabel.pkg
```

in every source file. I recommend doing a "Find in Files" over your complete workspace. If you already use a previous version of the Library, you also have to remove this version from your Workspace and add the new Library again as described above.

The existing interface to List & Label is not changed so you do not have to change anything else. Just be careful to set the psLicensingInfo property to your personal license key for List & Label 25.

If you consider using the library (which I would recommend) you have to remove any llxx.pkg and llpreview.dg files from your project's AppSrc Directory(s) to make sure that the correct files are loaded from the library. After removing these files and adding the library to your workspace your project should compile and run as usual.

What's New

List & Label Library 25.0

- Library and Sample Application have been updated to List & Label 25.

List & Label Library 24.0

- Workspace Files for DataFlex 19.1 have been added for the library and the sample application.
- Because of the Language and Code Cleanup starting with DataFlex 19.1 and the resulting Compiler Warnings I decided to cleanup the Library Code, too. All obsolete Code Techniques are now replaced with the actual Techniques recommended by DataAccess.
- Since the WindowsLibrary was extended with 19.1 the MoveTo() Function in the cListLabelDataObject Class resulted in an Compiler Error because of the Global MoveTo() Function from the Windows GDI Library. The only way to solve this was to rename the MoveTo() Function to MoveToRecord(). If you use cListLabelDataObjects in your existing Projects search for a call to the MoveTo() Function and replace it with MoveToRecord().

List & Label Library 23.0

- Workspace Files for DataFlex 19.0 have been added for the library and the sample application.

List & Label Library 22.0

- Workspace Files for DataFlex 18.2 have been added for the library and the sample application.

List & Label Library 21.0

- Workspace Files for DataFlex 18.1 have been added for the library and the sample application.
- The Preview Control (cVDFListLabelViewer) has a new Property piMouseMode which can be set to LS_OPTION_MOUSMODE_MOVE or LS_OPTION_MOUSEMODE_ZOOM. This controls whether the Control scrolls through the Preview or draw a Zoom Rectangle on dragging the Mouse inside the Preview.
- The Preview Control also has a new Method ZoomPageWith which set's the Zoom to the whole Page Width.
- Added a new DefineFileStruct Method which makes it easier to use List & Label without DataDictionaries. There is also a new Sample View "Report->Special Samples->Reports without DataDictionary" Take a look into the File LLEmbeddedWithoutDD.rv which shows how to use List & Label without any DataDictionary.
- The Sample Application is now useable for VDF16.0 and above. The Samples for DataFlex Versions below 16.0 are removed.

List & Label Library 20.0

- Workspace files for Visual DataFlex 18.0 have been added for the library and the sample application.

List & Label Library 19.0

- Workspace files for Visual DataFlex 17.1 have been added for the library and the sample application.

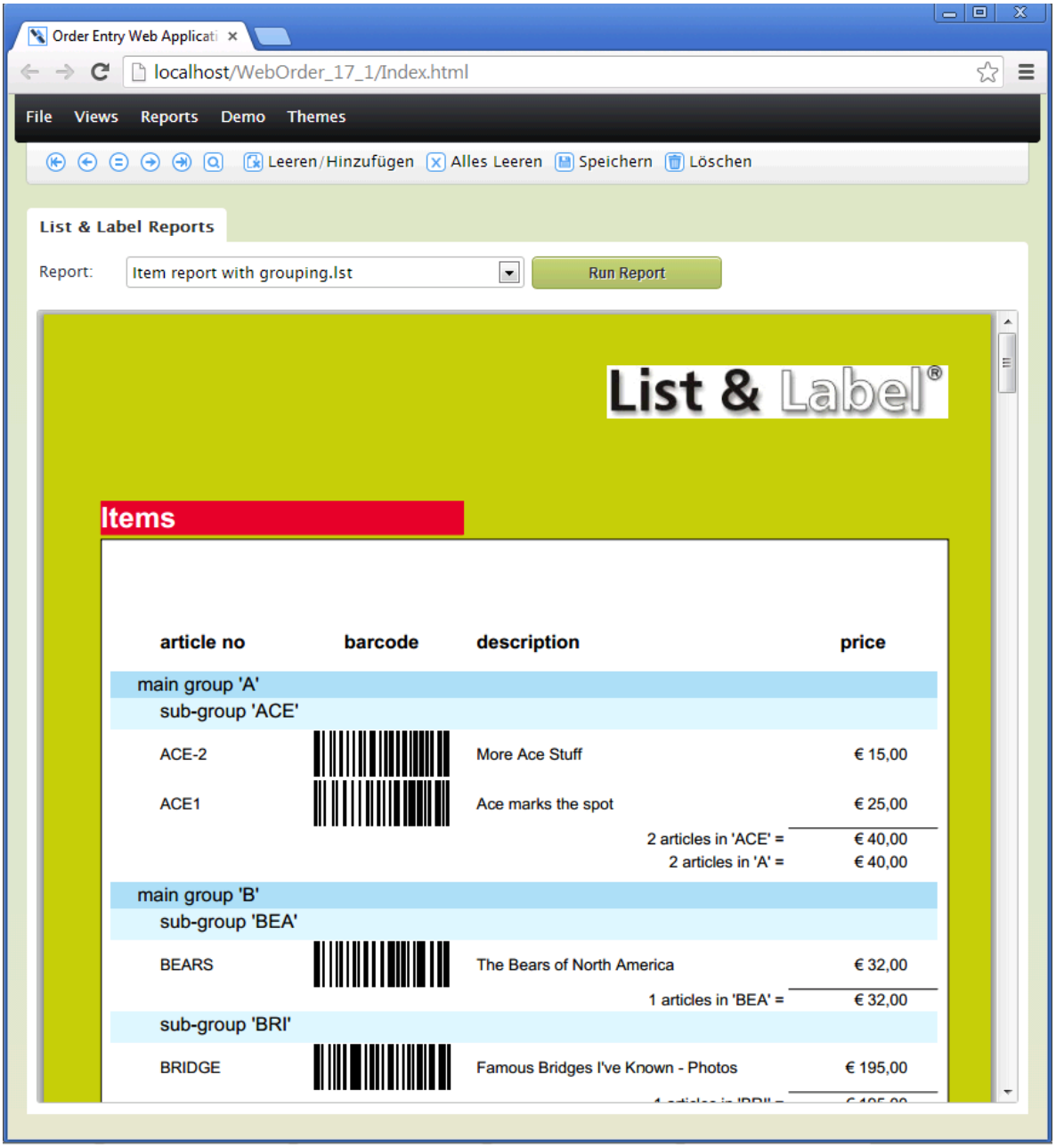
- Starting with the new Web Framework of Visual DataFlex 17.1 List & Label can be used for Web Reporting. If you want use List & Label in a Web Project, you just have to add the library the same way as in a Windows project. This adds a new option "List & Label Web Report" to the "Create new…" wizard. When you select this option you will get a new Web-View containing a Combo Box that is filled with all existing projects, a button "Run Report" which starts the selected report and a cWebIFrame which displays the report result.

  Just add all needed Data Dictionaries to the Web View and copy the List & Label project files into the appropriate folder (per default: {Workspace-Home}\Reports\Layouts). You may have to create the folder If it doesn't exist. Currently the List & Label Designer can't be used in Web Projects. For creating and editing your reports you have to use a Windows project which uses the same Data Dictionaries as the Web project. The resulting projects can be used for Windows and for Web.

  In general, the Web report quietly exports the report into a PDF which is streamed into a cWebIFrame Control afterwards. Here is a code fragment out of a Web Report object:

```
Object oLLReport is a cListLabelReport
    // Uncomment this Line and set the Property to your personal
    // License Code (perslic.txt located in your List & Label Folder)
    // Set psLicensingInfo to "123456"

    // Initialize Settings for Webreporting.
    Procedure DoInitSettings
        String sPath
        // All Preview-Features who need to interact with the Report during the Preview like
        // Incremental Preview, Drilldown Reporting have to be disabled.
        Set pbIncrementalPreview to False
        Set pbUseDrilldown to False
        Set pbUseExpandableRegions to False
        Set pbDesignerPreview to False
        // Set the Directory containing the Report Projects (psDefaultLayoutDirectory).
        // Defaults to Workspace-Home\Report\Layouts
        Move (psHome(phoWorkspace(ghoApplication))) to sPath
        If (Right(sPath,1)<>"\") Move (sPath+"\") to sPath
        Move (sPath+"Reports\Layouts") to sPath
        Set psDefaultLayoutDirectory to sPath
    End_Procedure
    Send DoInitSettings
End_Object
// This Method exports the Report as a PDF and loads the Result in an cWebIFrame
Procedure RunReport
    String sUrl sProject
    // Get the selected Project from the Combo-Box
    WebGet psValue of oProjectSelection to sProject
    If (Trim(sProject)="") Procedure_Return
    // Export the Project and deliver the Result as a Temporary Download-URL
    Get ExportReportForWeb of oLLReport sProject to sUrl
    If (sUrl="") Begin
        Send ShowInfoBox "The Report could not be generated"
    End
    Else Begin
        // Load the PDF-Result in the cWebIFrame
        WebSet psUrl of oReportResult to sUrl
    End
End_Procedure
```



- New Properties

  pbUseExpandableRegions
  pbUseReportParameter
  pbUseInteractiveSorting

  You can set these properties to True if you want to use the new expandable regions, report parameters or interactive sorting features from List & Label 19. These properties are False by default so your existing projects would not change.

- New property pbReportIsInteractiveRequest: When the user requests an expandable region, new

report parameter or an interactive sorting the report is started again. If needed you can check the property pbReportIsInteractiveRequest to see if the report is running "normally" or a second time because of a user request.

List & Label Library 18.0

- Workspace files for DataFlex 17.0 have been added for the library and the sample application.
- LLDesignerAddAction function added to the library. This function was added to support your own actions on the new Ribbon in the Designer.

List & Label Library 17.0

- Workspace files for Visual DataFlex 16.1 have been added for the library and the sample application.
- Real-data preview inside the Designer: It is now possible to preview your project directly in the Designer with real data just as you would print in a preview window. To use this feature, you have to set the property pbDesignerPreview to true in your List & Label Object. Additionally, you have to tell the Designer which message should be used to start the report. This is done via the property piDesignerPreviewStartMessage. Normally you set this property to the same message that you use to start your report. So if you add the lines

  set pbDesignerPreview to True
  set piDesignerPreviewStartMessage to msg_StartReport

  to your List & Label Object you can click on the Preview-tab directly inside the Designer. The Designer then sends the message StartReport to your object and the report will be printed inside the Designer. So you can quickly change the layout and see what happens on the fly, without exiting the Designer and starting a preview-print. The reports in the sample have this feature activated, so you can easily try it. If you need to determine if the report is running inside the Designer preview you can check the property pbDesignerPreviewRunning which is true when the report runs inside the Designer, or false if it was started normally.
- Localization support: It is now possible to use the localization features of List & Label. With this feature you can for example use the same project in English or German. Look at Localization.Rv in the sample to see how you can use the functions

  LLLocAddDesignLCID
  LlLocAddDictionaryEntry
  DoSetPrintLanguage

  for supporting multi language projects.
- Better support for non-Recnum Tables: There where some problems with non-Recnum tables that are fixed now.
- When printing in preview there could have been problems if the report was inside a modal window. Modal windows are now fully supported.
- The OnDefineIndexForTable notification has been extended. Now the index selected in the Designer is

delivered as last parameter for this notification. Additionally, two new return values are possible: LL_INDEX_OPTIMIZED_UP and LL_INDEX_OPTIMIZED_DOWN. When returning one of these values the DataDictionary selects the best index just like setting the ordering of a dictionary to -1.

- The OnLL_CMND_Evaluate notification (called when the External$() Designer function is used) now supports values greater than 100 characters which was a limitation until this release.

List & Label Library 16.0

- With version 16.0 of the library the filename was changed to cListLabel.pkg. When only using the cListLabel.pkg you always compile the latest revision of List & Label into your application. If you still want to use List & Label 15 you have to add the following Source BEFORE the first use of cListLabel.pkg

  Define USELLVERSION for 15
  use cListLabel.pkg

  The Library supports List & Label 17, 16 and 15, but you can only use one revision of List & Label in each Project, not both.

- Easier support for "User defined Tables". User defined tables are an easy way to deliver calculated or whatever values to List & Label. You can also deliver DataFlex databases without DataDictionaries to List & Label this way.

  How to use User defined Tables:

  First of all you have to tell List & Label about the new table you want to add inside the DefineAllFields notification of the cListLabelReport object. Here is an example of how to define a table named "USERDEFINVT" with two possible sort orders. The table fields are defined inside a helper Procedure (in this Example this Helper is called DoDefineUserdefInvtValues) because this definitions are needed again when actually filling the data:

```
Procedure DefineAllFields
    Integer iRet
    Handle hoDo
    Boolean bRet
    Forward Send DefineAllFields
    // Add the Table USERDEFINVT when the current Project is "Item Label with userdefined Table.lbl"
    If (Pos("ITEM LABEL WITH USERDEFINED TABLE.LBL",(uppercase(psCurrentProjectFile(Self))))<>0) Begin
        Get LlDbAddTable "USERDEFINVT" "USERDEFINVT" to iRet                        // Add the Table "USERDEFINVT"
        Get LlDbAddTableSortOrder "USERDEFINVT" 0 "First to last Record" to iRet    // Define two Sortorders for this Table
        Get LlDbAddTableSortOrder "USERDEFINVT" 1 "Last to first Record" to iRet
        Send DoDefineUserdefInvtValues                                              // And define the Fields for this Table
    End
End_Procedure

// Helper Procedure for defining all Fields for the "USERDEFINVT" Table
Procedure DoDefineUserdefInvtValues
    Set LlValue "USERDEFINVT.MyItemId" LL_TEXT to (trim(Invt.Item_ID))
    Set LlValue "USERDEFINVT.MyItemDescription" LL_TEXT to (trim(Invt.Description))
    Set LlValue "USERDEFINVT.MyItemPrice" LL_NUMERIC to (trim(Invt.Unit_Price))
    Set LlValue "USERDEFINVT.MyVendor" LL_TEXT to (Trim(Vendor.Name))
End_Procedure
```

  You also have to react on both notifications OnFindFirstRecord and OnFindNextRecord to fill in the actual data needed:

```
Procedure OnFindFirstRecord Integer hoDDo Integer hoParentDDo Integer iTableLevel String sRelation Integer iIndex Integer bF
    If ( (hoDDo=0) and (sTableName="USERDEFINVT") ) Begin         // Whenever the Table is "USERDEFINVT" and the hoDDo is 0
        If (iIndex=1) Send Find to oInvt_DD LAST_RECORD Index.1   // Find the first or last record depending on the Index
        Else Send Find to oInvt_DD FIRST_RECORD Index.1
        If (FindErr) Procedure_Return LL_REPORT_EOF               // No record found, return LL_REPORT_EOF
        Send DoDefineUserdefInvtValues                            // Define the Fields for the Table
        Procedure_Return LL_REPORT_RECORDFOUND                    // Tell the Report that a Record was found
    End
    Procedure_Return LL_REPORT_USEDEFAULT                         // in all other Cases let the Class do the Job
End_Procedure

Procedure OnFindNextRecord Integer hoDDo Integer hoParentDDo Integer iTableLevel String sRelation Integer iFindMode Integer :
    If ( (hoDDo=0) and (sTableName="USERDEFINVT") ) Begin         // Whenever the Table is "USERDEFINVT" and the hoDDo is 0
        If (iIndex=1) Send Find to oInvt_DD LT Index.1            // Find the next or previous record depending on the Index
        Else Send Find to oInvt_DD GT Index.1
        If (FindErr) Procedure_Return LL_REPORT_EOF               // No more records found, return LL_REPORT_EOF
        Send DoDefineUserdefInvtValues                            // Define the Fields for the Table
        Procedure_Return LL_REPORT_RECORDFOUND                    // Tell the Report that a Record was found
    End
    Procedure_Return LL_REPORT_USEDEFAULT                         // in all other Cases let the Class do the Job
End_Procedure
```

The screenshots are taken out of the sample, source file LLReports.rv. Just search for USERDEFINVT inside this file. The delivered label project "Item Label with userdefined Table" is using this table.

- Added support for using a CDO style interface. This interface was built just like the Crystal Reports CDO class and you should be able to keep your existing CDO sources and TTX files for transferring data to List & Label.

  How to use the new List & Label DataObjects:

  Inside the DefineAllFields notification of a cListLabelReport you have to create one or more DataObject, either based on an existing .TTX file or by creating a blank DataObject and manually adding fields to it. Here you see an example of adding one DataObject using the field definition out of the existing "Label.ttx" file and another DataObject "ItemLabel" where the field definition is programmatically built:

```
Procedure DefineAllFields
    Integer iRet
    Handle hoDo
    Boolean bRet
    Forward Send DefineAllFields
    // Create a List&Label Dataobject from a .TTX File if the Project is Addresslabel with Data from TTX.lbl"
    If (Pos("ADDRESSLABEL WITH DATA FROM TTX.LBL",(uppercase(psCurrentProjectFile(Self))))<>0) Begin
        Get CreateLLDo "Label.ttx" to hoDo                        // Create the DataObject from the TTX File
    End
    // Create a List&Label Dataobject if the Project is "Item Label with Dataobject.lbl"
    If (Pos("ITEM LABEL WITH DATAOBJECT.LBL",(uppercase(psCurrentProjectFile(Self))))<>0) Begin
        Get CreateLLDo "ItemLabel" to hoDo                        // Create the DataObject from the TTX File
        Get AddField of hoDo "ItemId" LL_TEXT to bRet             // and add Fields to the DataObject
        Get AddField of hoDo "Description" LL_TEXT to bRet
        Get AddField of hoDo "Price" LL_NUMERIC to bRet
        Get AddField of hoDo "Vendorname" LL_TEXT to bRet
    End
End_Procedure
```

When List & Label requests one of your DataObjects you have to react inside the OnStartTable notification. You also can fill your DataObjects already in the DefineLLFields notification, but maybe the data are not needed in the whole report and therefore filled useless. The OnStartTable notification is the point where the table is really needed by List & Label. It is also possible to create DataObjects, fill in the data and attach it to a cListLabelReport object. As in CDO you have to transfer the data in an Variant[ ][ ] array, where the first dimension of the array holds one record, and the

second dimension holds the date for every field defined in one record.

Here you can see how to fill the "Label" and the "ItemLabel" DataObjects defined above:

```
Procedure OnStartTable Integer hoDDo Integer hoParentDdo Integer iTableLevel String sRelation String sTableName
    Integer iItem
    Variant[][] vData

    // Fill in the Data for the Label.ttx DataObject
    If (sTableName="Label") Begin
        Move 0 to iItem
        Send Find to oCustomer_DD FIRST_RECORD Index.1   // Loop through the Database
        While (Found)
            Move Customer.Name to vData[iItem][0]        // Fill in each Record into the vData Array of Type Variant[][]
            Move Customer.Address to vData[iItem][1]
            Move Customer.City to vData[iItem][2]
            Move Customer.State to vData[iItem][3]
            Move Customer.Zip to vData[iItem][4]

            Send Find to oCustomer_DD NEXT_RECORD Index.1
            Increment iItem
        Loop
        Send SetLLDoData hoDDo vData                     // Set the Data of the DataObject to the Variant[][] Array, the D
    End

    // Fill in the Data for the ItemLabel DataObject
    If (sTableName="ItemLabel") Begin
        Move 0 to iItem
        Send Find to oInvt_DD FIRST_RECORD Index.1       // Loop through the Database
        While (Found)
            Move Invt.Item_ID to vData[iItem][0]         // Fill in each Record into the vData Array of Type Variant[][]
            Move Invt.Description to vData[iItem][1]
            Move Invt.Unit_Price to vData[iItem][2]
            Move Vendor.Name to vData[iItem][3]
            Send Find to oInvt_DD NEXT_RECORD Index.1
            Increment iItem
        Loop
        Send SetLLDoData hoDDo vData                     // Set the Data of the DataObject to the Variant[][] Array, the D
    End
End_Procedure
```

Again, the screenshots are taken out of the sample, source file LLReports.rv. The delivered label project "Addresslabel with Data from TTX" is using the label table based on the label.ttx file and the project "Item Label with Dataobject" uses the second Dataobject "ItemLabel" where the fields are defined programmatically.

- New class cListLabelDataObject which supports the following methods (just like the original CDO object):

  Function AddField String sName Integer iFieldType Returns Boolean
  Is used to add fields to the DataObject

  Function DeleteField String sName Returns Boolean
  Can be used to delete a field

  Function GetFieldName Integer iColumn Returns String
  Function GetFieldType Integer iColumn Returns Integer
  Delivers the name or the type of the contained fields

  Function GetFieldData Integer iColumn Returns Variant
  Delivers the value of the current active record-column

Function ColCount Returns Integer
Delivers the number of columns in the DataObject

Function RowCount Returns Integer
Delivers the number of rows (=records) inside the DataObject

Procedure AddRows Variant[ ][ ] vData
Adds the data vData[ ][ ] to the DataObject

Procedure DoSetData Variant[ ][ ] vData
Replaces the existing data in the DataObject with the vData[ ][ ]

Function MoveFirst Returns Boolean
Function MoveNext Returns Boolean
Function MoveTo Integer iRecord Returns Boolean
Can be used to manipulate the data pointer for the current record

Function GetEof Returns Boolean
Returns True when the data pointer is on the last record in the DataObject.

- New methods:

  Set LLValue {Integer LL_Type} {String ValueToSet}
  This method is used with "User defined Tables" to transfer values of user defined data fields to List & Label

  Function LLDoForTable String sName Returns Integer
  Delivers an Object-ID of a DataObject for a table or 0 if no DataObject exist for this table

  Function CreateLLDo String sName Returns Integer
  Is used to create a DataObject either based on a TTX file (sName is the ttx-filename) or a blank DataObject where you have to add the fields yourself

  Procedure AppendLLDoData Integer hoDo Variant[ ][ ] vData
  Appends data in vData to a DataObject

  Procedure SetLLDoData Integer hoDo Variant[ ][ ] vData
  Replaces the data for a DataObject with the data delivered in vData

  Procedure AttachLLDo String sName Integer hoDo
  Attaches an already existing DataObject named as sName to a List & Label report

  Procedure DetachLLdo Integer hoDo
  Detaches a previously attached DataObject from a List & Label report

- New notifications:

  OnBeforePrintFirstRecordForTable
  Is called when the first record for a table was searched and is ready for printing

  OnAfterPrintLastRecordForTable

Is called after the last record of a table was printed but before the table is finished

OnBeforePrintDrilldown
Is called before a Drilldown-report get's actually printed. The parent record is already found here

- LLSelectFileDlgTitleEx / LLDefineLayout extended to support the "New" Button:

  As the OpenDialog now has a "New" Button for creating new List & Label Projects the LLSelectFileDlgTitleEx now returns the fixed Value "CREATE_NEW_PROJECT" if the user selects this Button. The LLDefineLayout is accepting the Value "CREATE_NEW_PROJECT" in the Project File Parameter to create a new List & Label Project. The user has to enter the Filename of the new Project on closing the Designer or the first Time the Project is saved. By doing it this Way, your existing Sources which uses this Dialog should be still functional.

List & Label Library 15.1

- First release of the library, usable with Visual DataFlex 12.0 and higher.

- Embedded preview and cListLabelReportView class:
  The Preview-Object can now be placed directly into your Report-View. This eliminates the popup of the additional Preview window; you can change Report-Options directly in your Report View and reprint the report. To use this, the Report View has to be based on the new cListLabelReportView class. This class is based on the standard ReportView class and adds some extension needed for holding the Preview Object inside. The new properties piEmbeddedPreviewObject and piEmbeddedPreviewCancelBtn must also be set. The provided template "List & Label Report with Preview" is using this.

- Change table settings directly in the Designer for label and card projects:
  When creating or editing label and card projects there is a new menu item in the "Project" menu of the List & Label Designer. This menu calls a DataFlex popup (LLTableSelect.dg) and lets you change the table settings for the report directly. Note that this feature doesn't work with the Standard Edition of List & Label because the DOM editing feature is not included in the Standard Edition. With the Professional or Enterprise Edition of List & Label this feature works. This popup is also displayed before a new label/card project is created.

- Debug output added for the report. When the piDebugMode property is set to any value other than 0 the Report Class now adds debug information that can be logged with Debwin (the Debug Tool from combit). This can be helpful if your report doesn't work like expected, so if you have problems with reports set the piDebugMode property to any supported mask and start Debwin before printing your report. All debug lines from the DataFlex report start with the text VDFREPORT.

List & Label Library 15

- Drilldown:
  The new Drilldown-feature of List & Label 15 is implemented. To use it you have to set the new property pbUseDrilldown to TRUE. With Drilldown reporting you get some new notifications:

  - OnStartDrilldownReport
  This notification is called whenever the user try to start a Drilldown-report. You can build constraints,

find the requested parent-record and whatever you need to do before the Drilldown-report is started. Return LL_REPORT_USEDEFAULT to let the Report-class do the whole job or 0 if you have found the parent-records yourself. Any other return value will cancel the Drilldown-report.

- OnFinishedDrilldownReport
Is called when the Drilldown is finished. Here you can revert the changes to constraints that you have done in OnStartDrilldownReport

- OnDrilldownAdditionalTables
- OnDrilldownUseTable
These notifications are only called when the pbDrilldownQueryDatabases property is set to True. In OnDrilldownAddtionalTables you can define additional databases, variables or fields that are only needed and used in subreports. In OnDrilldownUseTable you can hide defined tables when a subreport is opened in the Designer. Return LL_DRILLDOWN_HIDETALBE to hide a requested table or LL_DRILLDOWN_SHOWTABLE if the table should be displayed.

- OnDrilldownUseTableAsVariable
This notification is only called when the pbDrilldownBasetableAsVariable is set to True. By setting the pbDrilldownBasetableAsVariable to True the direct parent of a subreport is defined as variables. Additionally you can return 1 in this notification for each table that should be displayed as variables in the subreport.

Take a look at the sample to see how Drilldown is working.

- Most of the report notifications now get additional parameters. These parameters can be used if you have defined some databases which are not automatically defined by a data dictionary.

- Define databases AS:
It is now possible to define one database more than one time by the use of the

DefineLLField {Database} AS "Aliasname"
or
DefineLLVariable {Database} AS "Aliasname"

syntax. This way you can define the same database a second time. i.e.
DefineLLField CUSTOMER
DefineLLField CUSTOMER as "SHIPPINGCUSTOMER"
So you have the complete Customer database as CUSTOMER in List & Label, and a second time as SHIPPINGCUSTOMER. This way you can print more than one record out of the same database into one table line. If you use the "AS" syntax you are self responsible for providing the correct database values for the alias database. Normally inside the OnBeforePrintRecord notification you have to

- store the original records
- find the needed records for the alias database
- call the DoDefineUsedAliasFields or DoDefineUsedAliasVariables with the defined alias table name
- restore the original records

```
Procedure OnBeforePrintRecord Integer hoDDo Integer hoParentDDo Integer iTableLevel String
sRelation String sTableName
   RowId rId
   boolean bFound
   // Only if a Customer Record is Printed
```

```
    if (hoDDO=customer_dd) begin
        // Store the original Record
        Move (GetRowId(Customer.File_Number)) to rId
        // Find another Customer Record
        clear Customer
        move anyid to customer.id
        find eq customer by index.1
        if (finderr) clear customer
        // Transfer the Values into the Alias "SHIPPINGCUSTOMER"
        send DoDefineUsedAliasFields "SHIPPINGCUSTOMER"
            // Important!: Restore the original Customer Record
        Move (FindByRowId(Customer.File_Number, rId)) to bFound
    end
End_Procedure
```

- Structures:
  It is now possible to use Structure properties with List & Label. This makes it easier to transfer a bunch of data with only one call and you don't have to define every single desired value as a property.
  i.e.

```
// Define the normal DataFlex Structure
Struct tShippingAddress
    String sStreet
    String sCity
    String sState
    Integer iZip
End_Struct

// inside the List & Label object define a property based on the desired Structure
Property tShippingAddress paShippingAddress

// Since I was not able to read out the Structure definition directly you have
// to tell List & Label about the Structure

LLStruct tShippingAddress
    LLStructField sStreet "Street" LL_Text
    LLStructField sCity   "City"   LL_Text
    LLStructField sState  "State"  LL_Text
    LLStructField iZip    "Zipcode" LL_NUMERIC
End_LLStruct

// Now define the property in List & Label

Procedure DefineAllFields
    Forward Send DefineAllFields
    DefineLLVariable paShippingAddress tShippingAddress "ShippingAddress"
End_Procedure
```

  Now you can fill data into a variable based on tShippingAddress and simply set the property paShippingAddress to the variable.

- New property pbKeepJobOpen: When this property is set to TRUE a print job is always active until the

report object is destroyed. This mainly can speed up the report. If you want to use this test your report carefully. I've found no side effects in doing this, but it is not extensively tested.

List & Label Library 14

- It is now possible to use the same simple report logic for all types of projects (Lists, labels and cards). Since label and card project don't have the Report Container object where the needed Table, Index and Relation are selected these values are saved in three user defined Project-parameters for this types of projects:

  VDF.Report.Table
  VDF.Report.SortOrder
  VDF.Report.Relation

  The parameters can be set directly in the Designer or if either Table or SortOrder are undefined a dialog is shown after start of the report in which all possible Tables, Indexes and Relations can be selected. The selected values are the saved directly in the project file (via DOM objects). So if you have to change Tables, Indexes or Relations you can either edit the values directly in the Designer (then you have to know the possible values) or you can simply set the VDF.Report.Table parameter to blank and afterwards start the report again and select the desired values.

  If you have the Standard Edition of List & Label it is not possible to save the selected values into the project file automatically. After printing is finished you will get a Message-Box which tells you the values needed to be saved into the project file. Open the Designer, set the project parameters to the desired values and save the project. When the parameters are saved in the project you will no longer get the question about Table, Index and Relation.

- The ll15.pkg now has included all Meta-Data tags needed for Codesense and the Object Properties window in DataFlex Studio. It should now be much easier to work with the component and the Preview-Control.

List & Label Library 13

- The new List & Label "Preview while Printing" (Incremental Preview) option is implemented in the VDF class. By default, the option is disabled. If you want to use this option, set the new property pbIncrementalPreview to TRUE. If the standard preview of List & Label is used there is nothing else to do. If you are using the VDF control for displaying the preview, there are some additional steps needed. Before List & Label 14 the preview window normally was a modal window. Now the window cannot be modal any longer since it must be displayed during printing report. The preview window handling is now included in the VDF class. To use a VDF window as preview window you have to create a normal VDF View Object and put a List & Label preview object "cListLabelViewer" into the view. The view must be non-modal and can't be a deferred object. Then set the properties piPreviewPanel and piPreviewObject to the object IDs of your view and the cListLabelViewer object. When both properties are set you just need to call LlPreviewDisplay after the report. If you use the incremental preview option, make sure that LlPreviewDisplay is called after the report regardless if the user cancelled the report or not. This is important to clean up some internal stuff! The sample uses a VDF preview window so look into LLContRpt3.rv and LLPreview.Dg for more Information.

- There are two new notifications for the Preview object (cListLabelViewer):

OnIncrementalPreviewStart Integer iLLObj
and
OnIncrementalPreviewEnd Integer iLLObj

The notifications are sent when an incremental preview starts and just before the report and the preview is finished. Both notifications get the Object ID of the List & Label object that created the report as parameter.

- The List & Label objects (cListLabelWrapper and cListLabelReport) have a new property piTrimDbValues. When set to false (Default) all database values are defined untrimmed as in previous versions. When set to true all database values get trimmed before they are defined to List & Label.

- The new LLGetErrorText can be used to translate List & Label error constants to clear text error messages that can be displayed to the end user.

- Support for the new static table object is included. Look at the List & Label documentation how this object can be used in your reports. When a static table is about to be printed the notification OnPrintStaticTable is fired. As parameter this notification gets the name of the last "normal" database that was printed. You can return LL_REPORT_SKIPTABLE to suppress the printing of the static table or LL_REPORT_CANCELPRINT to completely cancel the whole report. If nothing or 0 is returned the static table is printed normally.
- All DOM functions are implemented. They can be used to manipulate or to create projects at runtime. Take a look at the List & Label documentation and in the sample to see how to use these functions.

List & Label Library 12

The LLGetUsedIdentifiers function is now used to improve the performance of larger projects. Every time a print starts the DoFillUsedIdentifiers method is called to set the property psUsedIdentifiers with a semicolon separated list of all actually used fields, variables and user-variables. If you have declared a database with i.E. 160 fields and only one field from this database is used in the project this will drastically improve the performance of your report since it is no longer necessary to define 159 fields that are not used in the report. Since the LLGetUsedIdentifiers function may take some time on large projects you can also override the DoFillUsedIdentifiers method to save the used identifiers in your database if you detect a change in the layout of the project file (using a timestamp of the file or something else). Otherwise set the psUsedIdentifiers property to the same value that it had the time of the last save of the project file. If you have a real large project this would improve the loading time of a project.

List & Label Library 11

- New class cListLabelReport. The class is completely based on the original cListLabelWrapper class and supports all functionality of the original class. Compared to the old class there are several advantages:

  - The new List & Label table structures are implemented

  - You no longer need to define your database files yourself. As long as you forward the DefineAllFields message all DataDictionaries that you have defined into the ReportView are automatically defined within List & Label. For list projects all parent/child structures and possible sort orders are also automatically defined.

If you don't forward the DefineAllFields message the new class behavior is the same as in the old cListLabelWrapper class.

- For list projects, you can run your report with a single line of code. The complete Report-Loop, all database findings and all the reporting logic is encapsulated in the class. This was done for easier using of the new table structures of List & Label 14. You just have to start the print job, let the user select the needed layout file and the printing options, set the appropriate constraints in your DataDictionaries and start the report. You don't have to take care about sub tables, appended tables, table levels and even the sort order. You can use several notifications to add additional functionality during the report.

combit GmbH Buecklestr. 3-5 78467 Konstanz Germany

combit.com