www.Young-Consulting.de
Wirtschafts- und EDV-Beratung

Informationstechnologie
Wirtschafts-Know-how
Sozialkompetenz

Telefon: +49 (0) 30 – 24 35 74 – 88
Telefax: +49 (0) 30 – 24 35 74 – 87
E-Mail: Thomas.Young@Young-Consulting.de

1 Demo

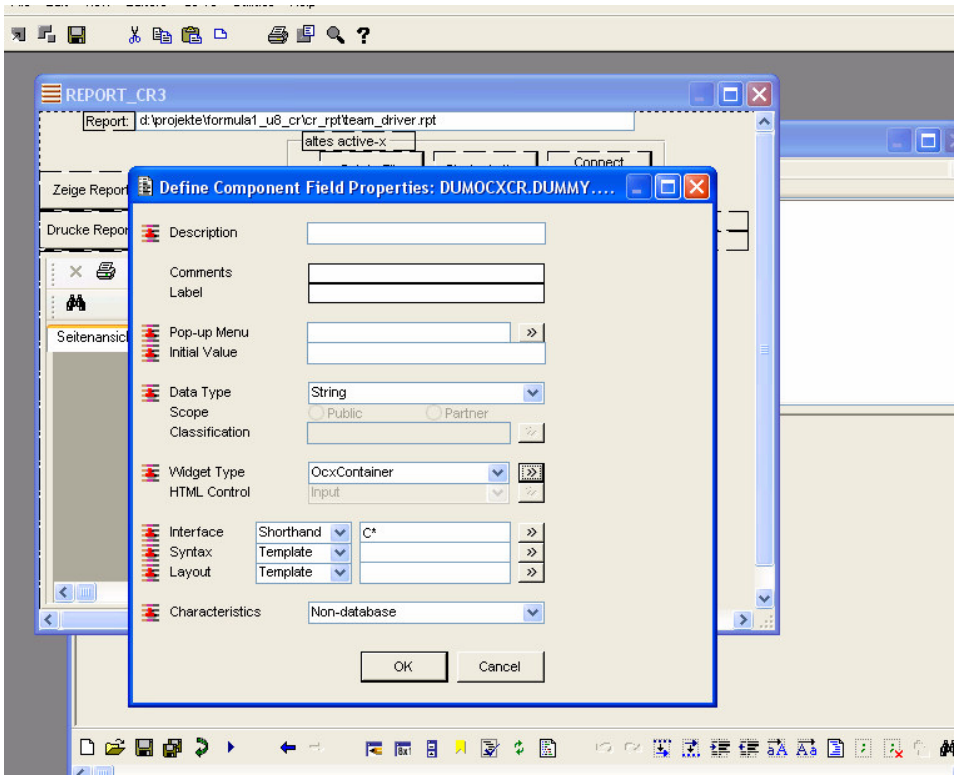
This document describes the integration of UNIFACE and Crystal Report. It is a summary of a German consulting document

1.1 OCX-Integration

1.1.1 Component Design

First, there will Crystal Report integrated via OCX into UNIFACE. In this case there is the UNIFACE component REPORT_CR3 which has an OCX painted. The OCX itself is from the version 9 of Crysal Report and available as “CRViewer.CRViwer”

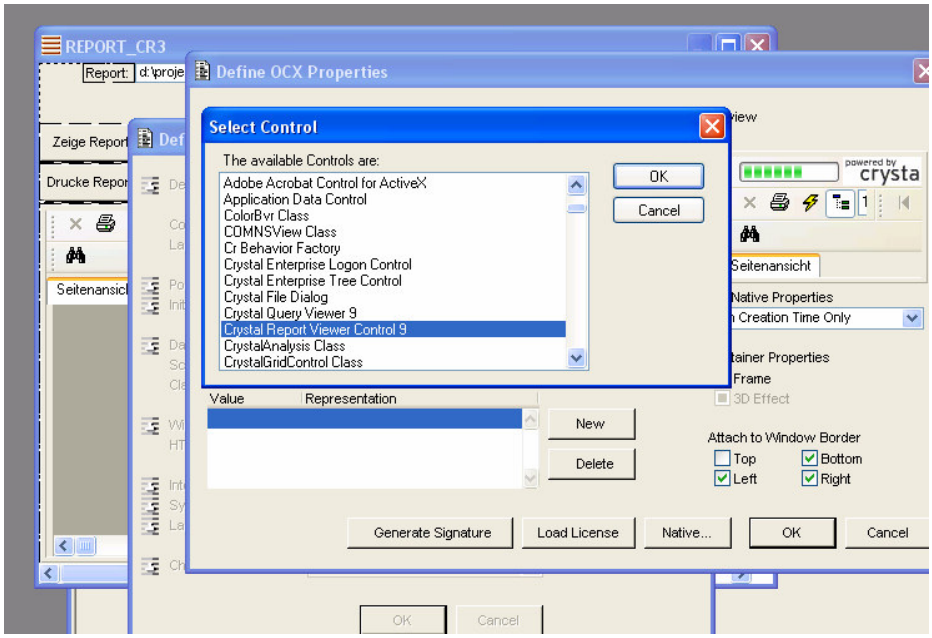
Paint an Entity named „DUMMY“ in a UNIFACE Form. There also paint a field called „DOMOCX”. These are dummies, because they are non-database-objects.



Picture 1: OCX-Widget

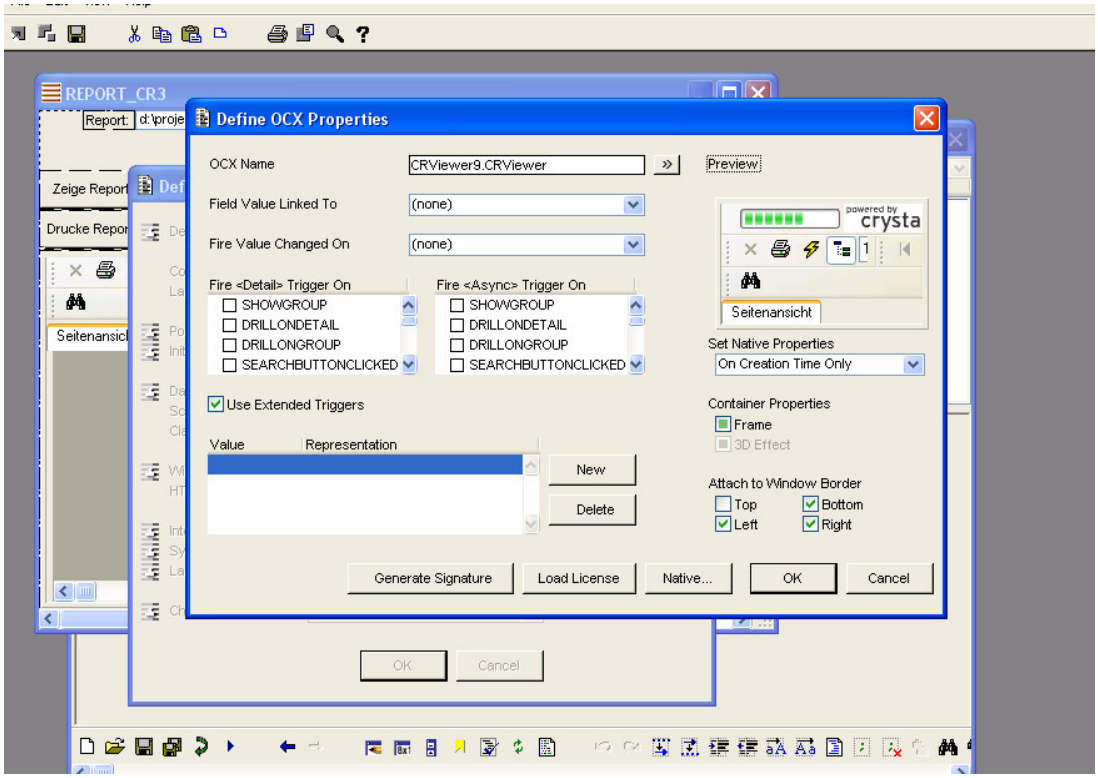
The properties of the field are string as datatype, ocxcontainer as widgettype. With clicking the detail-button nearby the widgettype-element you come to the propertie-mask of the widgettype, in this case the properties of the ocx.

At the first time, there isn't an ocx specified, so a form is popped up to select a specific ocx. They will all ocx listed which are installed on the system. Just select the one you wanted.



Picture 2: Selection OCX

For the Integration of Crystal Report you ought to select the "Crystal Report Viewer Control 9". If you have selected one, UNIFACE load the importants properties, events and method into the ocx-properties window.



Picture 3: Widget Type (OCX) Properties

Now you shall activate the extended Trigger (just click the checkbox on). Extended Triggers are fired by the events of the ocx. You can enhance the extented Trigger by your own programming. Now you can import the signature of the ocx through the button Generate Signature. (But be carefull, UNIFACE import only in a standard way…a better solution is to import via /sti-switch in the command line – it is descripted later)

Now the ocx is ready to tested and work. But every action is handled by the ocx itself. Most of the times, it is more comfortable to controll the ocx through UNIFACE.

To do this paint some more buttons and fields:

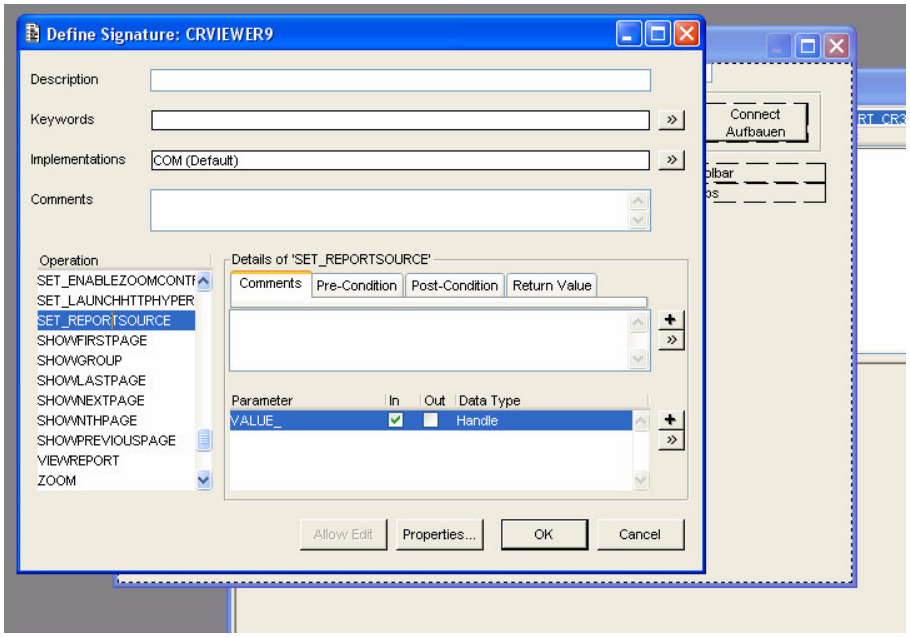
Feld	Widget Type	Aufgabe
DUMFILE	Editbox	To edit the path to a spezific report
BUT4	Commandbutton	Connect to Crystal Report and activate the report
BUT5	Commandbutton	Print the report
BUT6	Commandbutton	Go to the first page
BUT7	Commandbutton	Go to the previous page
BUT8	Commandbutton	Go to the next page
BUT9	Commandbutton	Go to the last page

CHECK1	Checkbox	Control if the toolbar is displayed
CHECK2	Checkbox	Control if tab are displayed

Picture 4: Field for OCX-Controlling

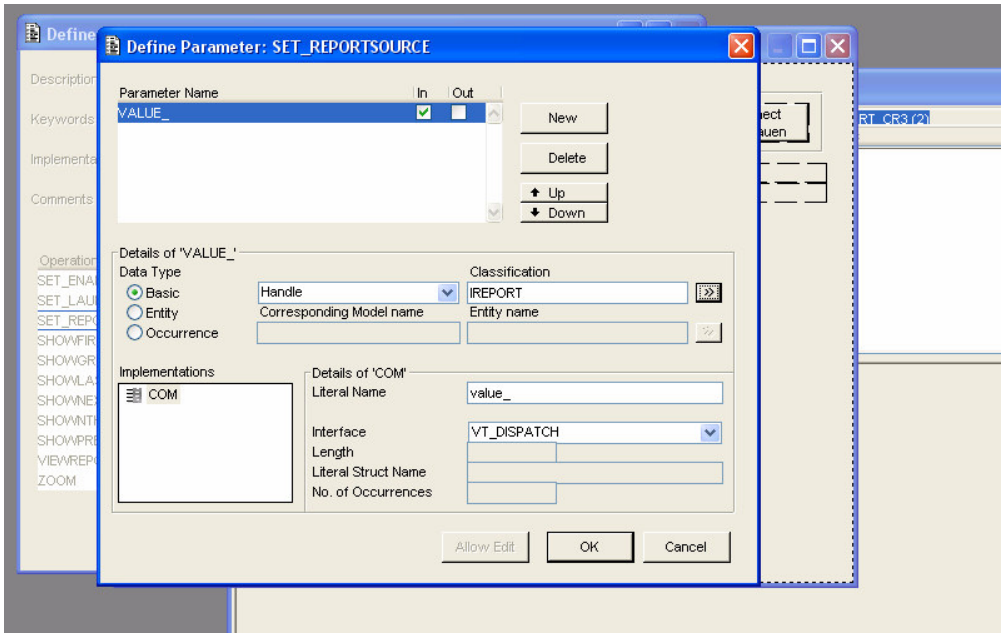
Methods and properties of the ocx are descripted in the handbook of Crystal Report. Have a look in the TechGuide for detailed description, but also the UserDevGuide you find an overview.

In the Signature Editor you find the imported Method to controll the ocx. They are mapped to the methods and properties of the ocx. Here you find all operation you can activate from UNIFACE.



Picture 5: Signature for OCX

If you use the standard import you find the signature in the subsystem COM. Most of the names for operations are identically to the methods. For properties there is a name-mapping. To get a propertie UNIFACE created a GET_PROPERTIE operation and to put it generates a SET_PROPERTIE operation.



Picture 6: Parameter Mapping

Also you find here a mapping for datatypes. There you can change some mappings because of problems with COM datatypes especially the VT_VARIANT type.

1.1.2 Need of import Crystal Report Application

Bevor you can work with the CR-OCX there is a big problem. The ocx needs a handle for the report. This handle can't create through UNIFACE self. So you need a different mechanism to create the handle. One solution is to use the CR Application to do this. The Crystal Report Application (Runtime) has a method to create a handle to a report. So we can use this method.

So the next step is to import the Crystal Report Application. This is described in the COM-Integration Chapter (Import with Command-line-Swith).

1.1.3 Programming

In UNIFACE Eight you can deal with handles. As OCX are objects it is natural way to deal with them. That's also the biggest different in way you integrated in UNIFACE Seven or UNIFACE Eight.

An object is a piece of code. It hat properties and method. An object can only invoked through a reference called handle. Properties can only reached through methods of the object.

An object can also create and involve other object. So for an application there exists a hierachical building of object. This is described by the object model. You should be familar whith the object model of the application you want to integrate.

BUT4 - <DTLF>

The first step is to create an instance of the Crystal Report Application (not the ocx). The Crystal Report Application enables us to create a handle to a report (see later). The new_instance-statement gives back a handle which is stored in the component-variable $myapp$.

```
new_instance "application", $myapp$
```

With the method logonserver crystal reports connects to a specific database. As parameters you give the database-driver (dll), the dns (database-name), a user and a passwort. Optional parameters are passed by the „–„sign.

```
$myapp$->LogonServer("PDSODBC.DLL","Formula1 Solid",-,"formula1","formula1")
```

The method openreport creates an handle for the report, which should activated. This handle is needed for the ocx to deal with a report. The handle is stored in the variable $myreport$. The main problem is that UNIFACE self can't create a handle to a report-file, so you have to use the openreport method from CR.

```
$myapp$->openreport($myreport$, DUMFILE.DUMMY, -)
```

Now, there create a handle for the ocx. With this handle you can program the ocx.

```
if (!$myocx$)
  $myocx$=$ocxhandle("DUMOCXCR.DUMMY")
endif
```

The ocx need a handle of the report. Only with such a handle it can activate a report. This is described in the object model of the report.

```
$myocx$->set_reportsource($myreport$)
```

Through discardsaveddata there will cleaned the report form old datas

```
$myreport$->discardsaveddata()
```

Now the ocx shall show the report

```
$myocx$->viewreport()
```

BUT5 - <DTLF> start a printout dialog

```
$myocx$->printreport()
```

BUT6 - <DTLF> show the first page of a report

```
$myocx$->showfirstpage()
```

BUT7 - <DTLF> show the previous page of a report

```
$myocx$->showpreviouspage()
```

BUT8 - <DTLF> show the next page of a report

```
$myocx$->shownextpage()
```

BUT9 - <DTLF> show the last page of a report

```
$myocx$->showlastpage()
```

CHECK1 – VALC set the property display toolbar

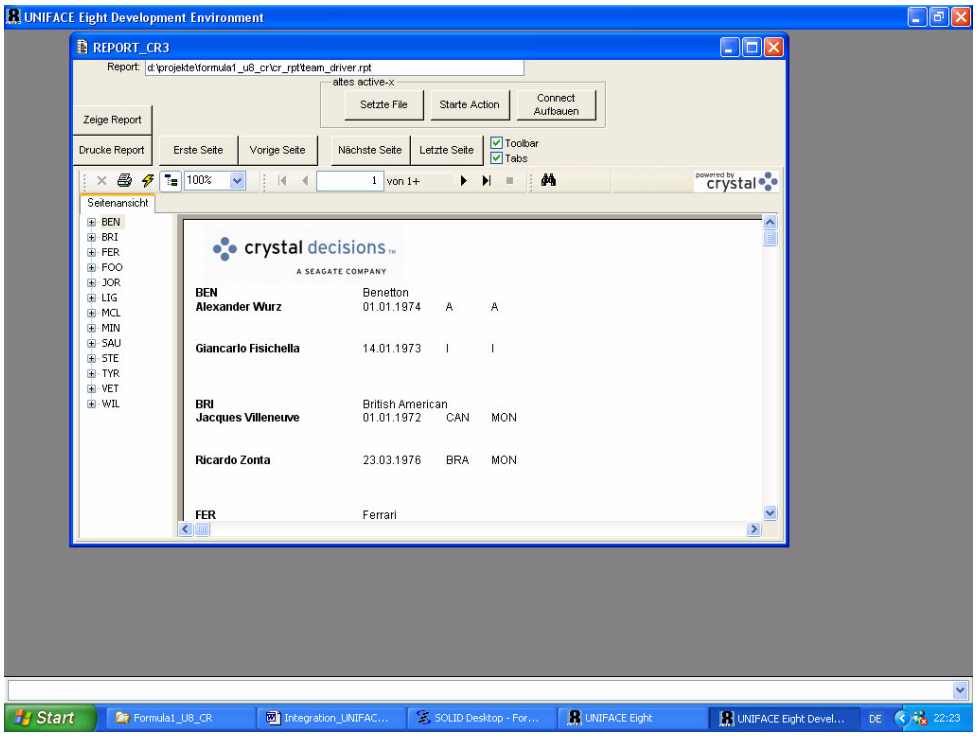
```
if (CHECK1.DUMMY = "T")
  $myocx$->set_displaytoolbar("TRUE")
else
  $myocx$->set_displaytoolbar("FALSE")
endif
```

CHECK2 – VALC set the property display tabs

```
if (CHECK2.DUMMY = "T")
  $myocx$->set_displaytabs("TRUE")
else
  $myocx$->set_displaytabs("FALSE")
endif
```

1.1.4 Testing

UNIFACE Forms with OCX have to test in a separat environment. So you have to start a new IDF with /tst or a runtime. Testing within the same idf process will failed.



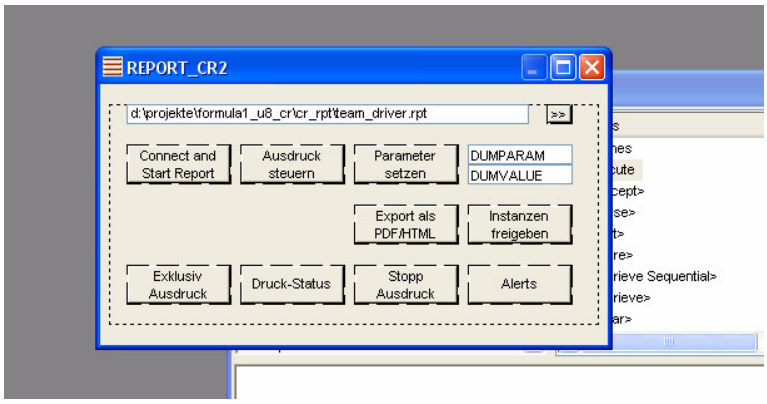
Picture 7: Test OCX

In this example there is the setting in the preferences used. Test modus is set to test in a separate developement environement. Have a look under Preferences->Test Modus.

1.2 COM - Integration

1.2.1 Component Design

The Form for the COM-Communication has a complete different layout. First there isn't any OCX painted. So in this component there isn't any disply of the report. Everythink is done in the backend.There are only several buttons for controlling the action:



Picture 8: COM throughCommandbutton

The commandbuttons are responsible for controlling the Crystal Report Application through UNIFACE:

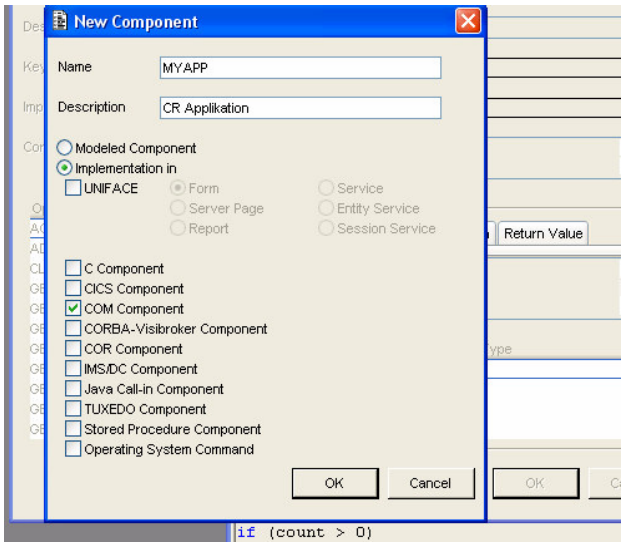
Feld	Widget Type	Aufgabe
DUMFILE	Editbox	Set a path to the report
DUMPARAM	Editbox	Set a name for a parameter to the report
DUMVALUE	Editbox	Has the value for the reportparameter
BUT1	Commandbutton	Creates a connection to Crystal Report and set the report (in DUMFILE) as active report
BUT2	Commandbutton	Start printing
BUT3	Commandbutton	Set the parameter, which are set in DUMPARAM and DUMVALUE
BUT4	Commandbutton	Delete the Report and Crystal Report Application
BUT5	Commandbutton	Start an export to pdf and xml
BUT6	Commandbutton	Cancel the printing
BUT7	Commandbutton	Open the File-Select-Windows from Windows to select a report
BUT10	Commandbutton	Information about the report and printing

		state

Picture 9: Fields for COM-Controlling

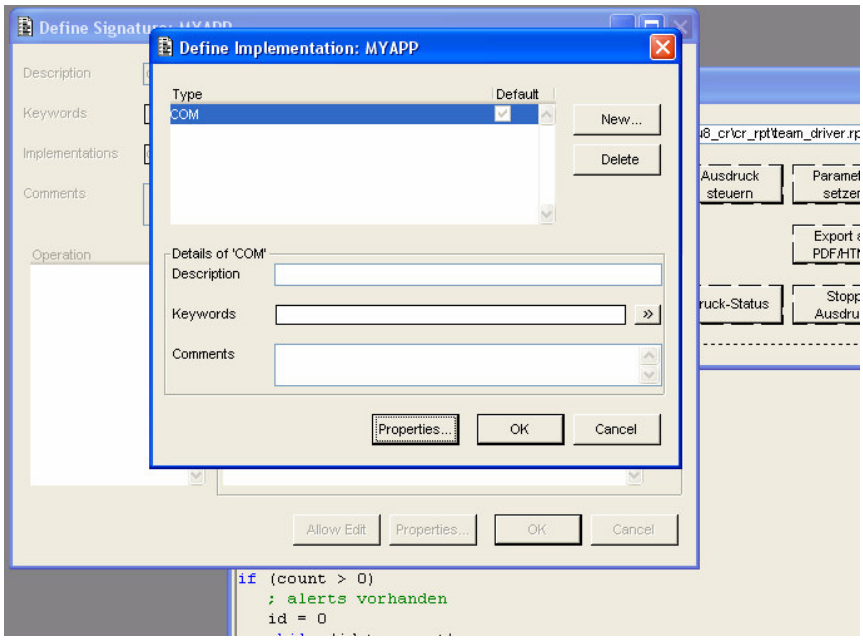
To implement this action, you need the accordingly object in UNIFACE. There are more objects than you need for the OCX. But also you have more opportunities to integrate with Crystal Report and have more control about the application. But it is necessary to have knowledge about the different objects and how they work together, so you should be familiar with object-model of Crystal Report.

The esiest way, to import all signatur you need, is to work with the signature-editor (be aware that this is not the best way (have a look at Import with Command-line-Switch).



Picture 10: Creating Signature

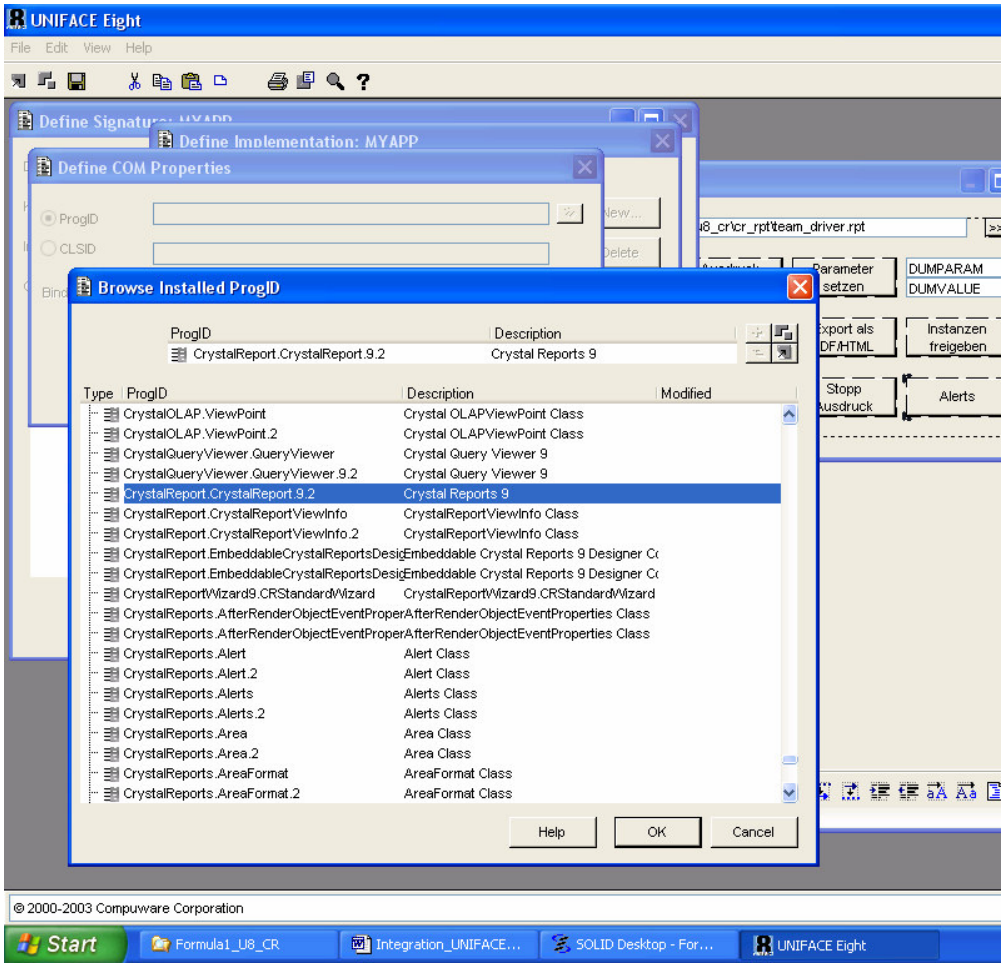
Just create a new component in the signature editor. Go to the implementing type and change this one from modeled to COM. Now the implation type has to be more specific. What kind of COM and which object. Press the detail button nearby the implementation type and you get the more properties form.



Picture 11: COM-Signature

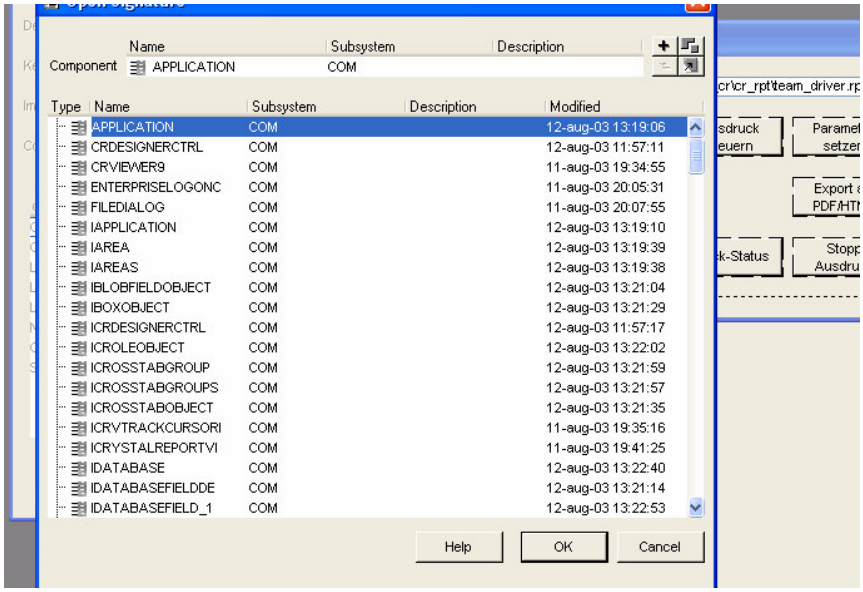
In this form you click on the button Properties… to get the mapping between UNIFACE component-name an COM-Object. Since there isn't any specific COM-Object selected you have to select one object. So all COM-Objects, which are registered in the Microsoft registry are displayed.

A COM-Object can be invoked through its prog-id or through its class-id. Normally you would reference it through the prog-id, which is more flexible. As prog-id you can search for the Crystal Report application, which is called „**CrystalRuntime.Application.9“**. For this information have a look in the TechGuide from Crystal Report.



Picture 12: Selection COM-Objekte

So start the import just click OK. Now the application-object and all objects below (belonging to this application) are imported. To know what all is imported have a look in the message frame and compare it with object model.



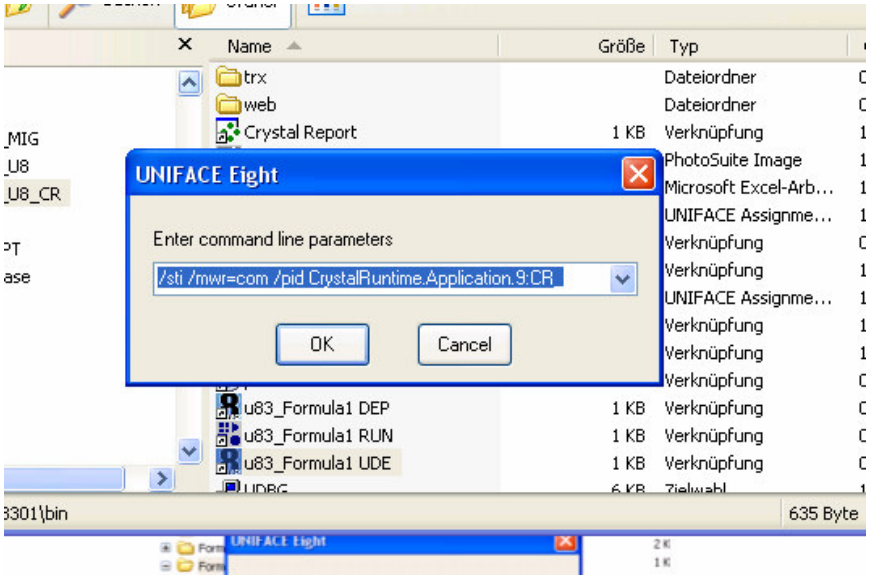
Picture 13: Result COM-Import

All signatures are imported and the mappings are set accordingly. Notice that there are several warnings in the message frame. These warnings are potentially problems when integrated with UNIFACE.

1.2.2 Import with Command-line-Switch

Not alway the esiest way is the better way, so to import signatur throught the signature editor has one important disadvantage. You can't influence the import process, especially the naming of the UNIFACE components, which will created. A better way is to import through the command-line switch **/sti**.

Start UNIFACE with a questionmark (?) and then the form for command-line parameters appears. In this form you insert parameters which will execute by UNIFACE. One of these parameters is the switch **/sti** (for stub in) with **/mwr=COM** (for Middleware = COM) and then the prog-id of the program which should imported (e.g. /sti /mwr=COM /pid CrystalRuntime.Application.9). Only this is the same as described in the chapter bevor. But now, you can enhance this by adding a prefix for the signature. So just add a ":CR_" after the prog-id (without any spaces). All signatures are imported with the prefix CR_ in there component-name.



Picture 14: COM-Import with Commandline

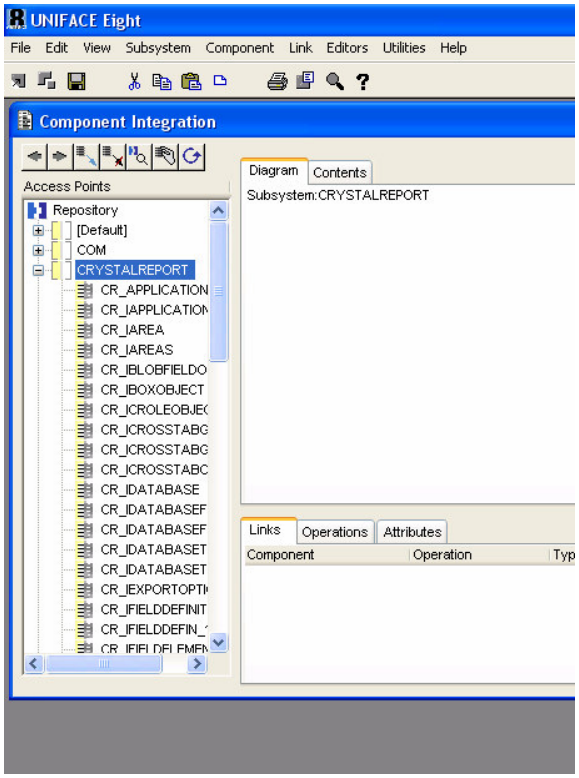
Now it is easier to distingt all signatures in your environment. Also you avoid conflicts because other object (e.g. MS Excel) also you use application as name-object.

After importing it make sense to admire all signatures in different subsystems. A subsystem is a logical system in UNIFACE where you can orgnized Signature which belong semantical together. Normally all imported Signatures (for COM objects) are placed in the subsystem COM.

So just go to the Assembly Workbench (in UNIFACE Eight called Component Integration). If there isn’n any subsystem called COM, create one which called COM. Now all object are displayed in this subsystem when make it active.

The next step is to move all CRYSTALREPORT signature to an own subsystem. So create a new subsystem called CR.

Make the subsystem COM active and go to the menu SUBSYSTEM->MOVE. Make a selection for all Crystal Report Signature. Now the prefix which is use at the import helps. Put CR_·* (·* is Gold Star) in the field selected component. As target insert your new subsystem CRYSTALREPORT.



Picture 15: Signature and Subsystems

Now the import is ready and also the signature are organized in an own subsystem.

1.2.3 Programming

Programming now is similar to the programming of the OCX. Of course we have already used at the ocx com-elements. To integrate Crystal Report into UNIFACE you need, as mentioned bevor, detailed information about the component object model of CR. Especially which objects exists and how you can create or invoke them.

In a short way there exists follow hierachical building:

APPLICATION

-> REPORT

-> EXPORTOPTIONS

-> PARAMETERFIELDLIST

-> PARAMERTERFIELD,

where the APPLICTION (imported as CR_APPLICATION) is the outer most object. It is the runtime-environment which contains all other objects. The main object we deal with is the REPORT object

(imported as CR_IREPORT). There we find the most importants properties and method for us. The report object itself contains also other object like EXPORTOPTIONS or PARAMETERFIELDLIST.

Every object is referenced through UNIFACE by an own handle.

BUT1 - <DTLF> Creates an instance of CR

The first step is to create an instance of the Crystal Report Application. The Crystal Report Application enables us to create a handle to a report (see later). The new_instance-statement gives back a handle which is stored in the component-variable $myapp$.

new_instance "application", $myapp$

With the method logonserver crystal reports connects to a specific database. As parameters you give the database-driver (dll), the dns (database-name), a user and a passwort. Optional parameters are passed by the „–„sign.

$myapp$->LogonServer("PDSODBC.DLL","Formula1 Solid",-,"formula1","formula1")

The method openreport creates an handle for the report, which should activated. This handle is needed for the ocx to deal with a report. The handle is stored in the variable $myreport$. The main problem is that UNIFACE self can't create a handle to a report-file, so you have to use the openreport method from CR.

$myapp$->openreport($myreport$, DUMFILE.DUMMY, -)

Through discardsaveddata there will cleaned the report form old datas

$myreport$->discardsaveddata()

BUT2 - <DTLF> Start the print-dialog

;simple print out throught the standard printer

The method PRINTOUT starts the printing process with follow parameters:

UserPrompt - if a dialog should appear
NumberOfCopy - numbers of copies
Collected - sort of copies
StartPage - Start-Page
EndPage - End-Page

myreport_handle->printout("FALSE","1","FALSE","1","0")

;PrintOut with selecting the printer

Starts the Windows PrinterSelect-Form, where the User can select a different printer

$myreport$->printersetup(0)

Die Eigenschaft PAPERSOURCE lässt sich über SET_ und GET_ auslesen, bzw. setzen. Hier kann das Through the Papersource you can selecte a different tray of the printer. In UNIFACE you can get the propertie through the SET_PAPERSOURCE and GET_PAPERSOURCE operations. Notice that Crystal has is own enum of values. So have a look in the documentation to set the right numbers.

myreport_handle->set_papersource(14)

At the printout method the endpage-parameter should have a very hight number. Unfortanetly using the optional sign (-) doesn't work.

myreport_handle->printout("FALSE","1","FALSE","2","1000")

;print out throut automatic printerselection

The methode SELECTPRINTER enables you to select a different printer. Therefor you need follow parameters:

DRIVERNAME - Printmanager (under Windows: winspool)

PRINTERNAME - Name of the printer

PRINTERPORT - Port of the Printer (LPT1 is lokal), IP-Adress or Netpath for Remote-Printer

$myreport$->selectprinter(„winspool“,“ Lexmark X84“,“LPT1“)

Setting the first parameter to TRUE. The user will get a window where he can change all parameters or just confirm them.

myreport_handle->printout("TRUE","1","FALSE","1","1000")

BUT3 - <DTLF> Set Report-Parameter

Declaration of variables for proc-coding

```
variables
  numeric count
  any valuetype
  any myparam
endvariables
```

The report-objec contains a properties PARAMETERFIELDS, which is a handle to a list of all Paramater, which are available for that report. Through the GET_PARAMETERFIELDS operation you get the handle.

$myreport$->Get_ParameterFields($myparams$)

The propertie COUNT gives you information how many parameters the report needs.

$myparams$->Get_Count (count)

```
count = $typed("$number(%%count%%%)")
if (count > 0)
  ; params vorhanden
```

The method GETITEMBYNAME looks fort he parameter wiht the name you put in and gives a handle to that parameter back.

$myparams$->GetItemByName($myparam$,DUMPARAM.DUMMY,-)

Parameters can have different datatype. In UNIFACE we have to distinct between different valuetype, because the communication throught the COM-datatype VT_VARIANT offen doesn’t work. So we ask fort he datatype with the operation GET_VALUETYPE. We get back numeric values, which are explained in the UserDevGuide of Crystal Report

```
  $myparam$->Get_ValueType(valuetype)
  selectcase (valuetype)
  case "12"
```

12 = string. This is no problem because the string is handled normaly by the operation ADDCURRENTVALUE. It will correct convert to Crystal Report.

```
   $myparam$->AddCurrentValue(DUMVALUE.DUMMY) ; string
```

All numeric-datatypes have more problems with the UNIFACE-COM-conversation. So we can’t use the normal operation with the VT_VARAINT as datatype. Just create a new operation

(ADDCURRENTVALUE_N) as copy of the orginial operation (ADDCURRENTVALUE) and set the datatype of the parameter to numeric on UNIFACE side. See also Tips & Trick.

```
  case "1"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "2"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "3"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "4"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "5"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "6"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "7"
   $myparam$->AddCurrentValue_N($typed("$number(%%DUMVALUE.DUMMY%%%)")) ; numeric
  case "9"
   $myparam$->AddCurrentValue_B($typed("$boolean(%%DUMVALUE.DUMMY%%%)")) ; boolean
```

Similar to the numeric datatype you have to create own operation for Boolean, Date and Time parameters.

```
  case "10"
   $myparam$->AddCurrentValue_D($typed("$date(%%DUMVALUE.DUMMY%%%)")) ; date
  case "11"
   $myparam$->AddCurrentValue_T($typed("$time(%%DUMVALUE.DUMMY%%%)")) ; time
  elsecase
    myparam = $typed("$number(%%DUMVALUE.DUMMY%%%)")
    $myparam$->AddCurrentValue_N(myparam)
  endselectcase
else
  message/hint "Keine Parameter vorhanden!"
endif
```

BUT4 - <DTLF> Delete the instance CR

To destroy an object, you have to nullify its entire handles. You can achive this by assigning NULL to the handle.

```
$myreport$ = “”
```

You should Logoff from the database bevor you close the application

```
$myapp$->logoffserver("PDSODBC.DLL","Formula1 Solid","FORMULA1","formula1","formula1")
```

Just use the delete_instance statement to destroy the application object.

```
delete_instance $myapp$
```

BUT5 - <DTLF> Generate some exports

```
; Export into XML-Format
```

With the method READRECORDS you force CrystalReport to read all datas. That is like setocc „*“, -1 in UNIFACE. I don’t really know if and when it is necessary.

```
$myreport$->readrecords()
```

Crystal Report controls the export through some own attributes. These are collected in an own object called EXPORTOPTIONS. This object is a child to the report-object. Through the properties (use the GET_EXPORTOPTIONS operation) you get a handle to that object.

```
$myreport$->Get_ExportOptions($MYEXPORTOPTION$)
```

CrystalReport is able to export directly to another application or to file. So1 as DestinationType is set to file.

```
 $MYEXPORTOPTION$->Set_DestinationType("1")
```

Paht and file-name can be set throught the propery DISKFILENAME. The filename must have the complete path.

```
$MYEXPORTOPTION$->Set_DiskFileName("%%DUMFILE.DUMMY%%%.xml")
```

Through FORMATTYPE you can set the kind of export. What format will use. Crysal Report use an own enum for the different formats (e.g. „37“ stands for XML)

```
$MYEXPORTOPTION$->Set_FormatType("37")
```

With XMLALLOWMULTIPLEFILES Crystal Report can create more than the xml-file. It can also create a DTD or XSD if necessary.

```
$MYEXPORTOPTION$->SET_XMLALLOWMULTIPLEFILES("FALSE")
```

Through the property XMLFILENAME there again is set the target file-name

```
$MYEXPORTOPTION$->SET_XMLFILENAME("%%DUMFILE.DUMMY%%%.xml")
```

The method EXPORT starts the export accordingly to the attributes set bevor. The parameter set to FALSE suppress the user-dialog.

```
$myreport$->export("FALSE")
```

; now also a pdf export

For the export to pdf you have only to set a different format type (pdf = “31”) and some additional pdf-properties.

```
; if (!$myexportoption$)
   $myreport$->Get_ExportOptions($MYEXPORTOPTION$)
endif
$MYEXPORTOPTION$->Set_DestinationType("1")
$MYEXPORTOPTION$->Set_DiskFileName("%%DUMFILE.DUMMY%%%.pdf")
$MYEXPORTOPTION$->Set_FormatType("31")
$MYEXPORTOPTION$->SET_PDFEXPORTALLPAGES("TRUE")
;$MYEXPORTOPTION$->SET_PDFFIRSTPAGENUMBER(xx) ---nicht mit export all pages
;$MYEXPORTOPTION$->SET_PDFLASTPAGENUMBER(xx) ---nicht mit export all pages
```

The method EXPORT with the first parameter =TRUE start the export with a User-dialog. In this dialog the user sees all settings and can confirm them or change them.

```
$myreport$->export("TRUE")
```

BUT6 - <DTLF>

The method CANCELPRINTING allows you to stop printing a report. This seems only to work as programming, not interactive because of the UNIFACE Structure-editor and that Crystal Report is started as synchronous process.

```
; cancel the printing
$myreport$->cancelprinting()
```

BUT10 - <DTLF>

```
Some variables for text message
variables
  string text, ausgabetext, seperator
endvariables
Initialize
seperator = " - "
text = ""
ausgabetext = ""

The Printing State is a separate object in CR. This object is a child of the report-object.
$myreport$->Get_PrintingStatus($mystatus$)

Throught the handle of Pinting State you can get several properties about the prining state.
$mystatus$->Get_NumberOfPages(text)
ausgabetext = "Pages %%text%%%"
$mystatus$->Get_NumberOfRecordPrinted(text)
ausgabetext = $concat(ausgabetext, seperator, "Print Record ", text)
$mystatus$->Get_NumberOfRecordRead(text)
ausgabetext = $concat(ausgabetext, seperator, "Read Record", text)
$mystatus$->Get_NumberOfRecordSelected(text)
ausgabetext = $concat(ausgabetext, seperator, "Select Record ",text)
$mystatus$->Get_Progress(text)
ausgabetext = $concat(ausgabetext, seperator, "PROGESS ", text)

message/info ausgabetext
```

1.2.4 Testing

For testing there no special hints. Just use the debugger as usal. An importing think is error-handling. $status seems not the important variable. Have a more close look in $procerror, which is set through UNIFACE. Every error from CR is reported also in the message-frame.

1.3 Tips & Tricks

This chapter show some hints for working with the com-driver.

Handle über Komponentenvariable

Objects are referenced through handles. These handles can be stored in variables in UNIFACE. You should use for the outermost object (most of the times the application object) a component variable. Because of trigger-variables lose there scope when the proc is finished the object also is deleted from memory. That can be OK for several child-objects but not for the application object which you need the whole time. In the example above we use component-variables for application and report.

$procerror überprüfen

All actions you perform again the crystal report application should be controlled with some error handling. Normally in UNIFACE you use $status for that, but in this case it is better to control $procerror, which is set to -150 in an error-case. More information about the kind of error you find in the message-frame or in $procerrorcontext.

Datentyp VT_VARIANT

COM declares the datatyp VT_VARIANT which is comparable to the any-datatype in UNIFACE. Unfortanetly the mapping between UNIFACE->COM->Crystal Report doesn't work that way. So CR offen give error, when the datatype is not as expected.

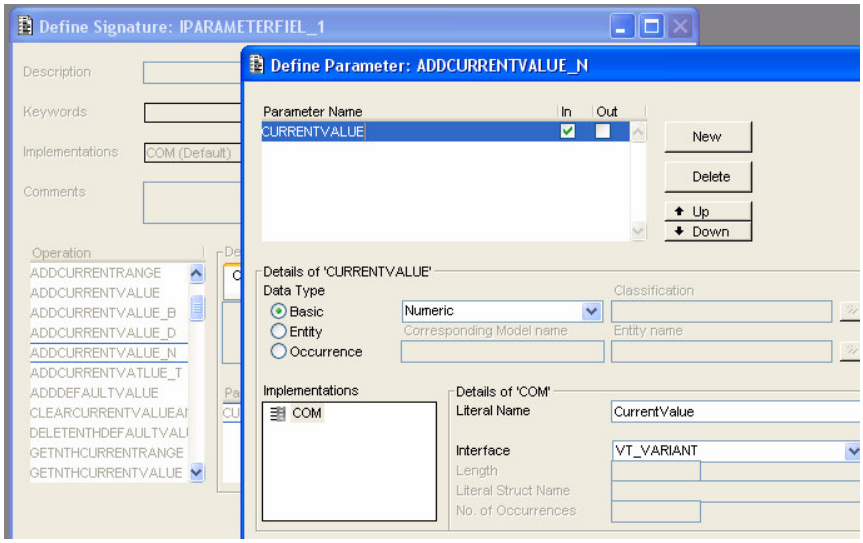
In most cases when you work in UNIFACE with a string this works fine, but if Crystal Reports expect numeric or boolean, than there is a mismatch.

e.g. Methode PrintOut

The method PrintOut has to parameters (PromptUser und Collected) which have VT_VARIANT as datatype for COM. Crystal Report but expect a Boolean-type. VT_VARIANT is offen in UNIFACE a string and so the conversion of string to boolean doesn't work. To correct this, you have to change the signature of this operation. Go to the more properties-form for parameters in the signature-editor. There change the COM-Interface from VT_VARIANT to VT_BOOL and compile the signature.

e.g. Methode AddCurrentValue

The method AddCurrentValue set a parameter for a spezial report. This parameter can have different valuetypes. So Crystal Report use a variant datatyp. In UNIFACE this is map to a string and for string values this works fine. But if you use other then string types in Crystal Report there will be an error occurre. To avoid this you have to create several operations with the same parameters. Just set in each case the UNIFACE-datatyp to the spezific datatype. So create a numeric for numeric_values aso. On COM-side the Interface will show VT_VARIANT.



Picture 16: Multiple Operations

ENUM Datatyp

Many application declare there own datatypes. Offen there are numeric values, ranges and enumerations.This kinds of datatypes allways mapped to the Interface VT_VARIANT and on UNIFACE-side to String. But that's not always ok. So for numeric values you have to change it.

```
uniface.log - Editor
Datei  Bearbeiten  Format  Ansicht  ?
2003-08-21 11:44:10.00 - Uniface session started

ProgId  [CrystalRuntime.Application.9]
Library [CRAXDRT]


CoClass           : CR_Application
Message : Using interface [CR_IAPPLICATION]
Interface         : CR_IReport
Warning : Function AddGroup, parameter pConditionField has an unknown HANDLE type VT_DISPATCH, please supply one
Warning : Renamed to [GET_ENABLEPERFORMQUERIESASYNCHRO]
Warning : Renamed to [SET_ENABLEPERFORMQUERIESASYNCHRO]
Warning : Renamed to [GET_ENABLEGENERATINGDATAFORHIDDE]
Warning : Renamed to [SET_ENABLEGENERATINGDATAFORHIDDE]
Interface         : CR_IAreas
Warning : Function _NewEnum, parameter result_ has an unknown HANDLE type VT_UNKNOWN, please supply one
Interface         : CR_IArea
Warning : Function GroupConditionField, parameter result_ has an unknown HANDLE type VT_DISPATCH, please supply one
Warning : Function GroupConditionField, parameter value_ has an unknown HANDLE type VT_DISPATCH, please supply one
Warning : Renamed to [GET_ENABLEHIERARCHICALGROUPSORTI]
Warning : Renamed to [SET_ENABLEHIERARCHICALGROUPSORTI]
Interface         : CR_ISections
Warning : Function _NewEnum, parameter result_ has an unknown HANDLE type VT_UNKNOWN, please supply one
Warning : Function Parent, parameter result_ has an unknown HANDLE type VT_DISPATCH, please supply one
Interface         : CR_ISection
Warning : Function BackColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BackColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function AddUnboundFieldobject, datatype VT_USERDEFINED used by parameter 1 is unsupported, using VT_BSTR instead
Interface         : CR_IReportObjects
Warning : Renamed to [CR_IREPORTOBJECT]
Warning : Function _NewEnum, parameter result_ has an unknown HANDLE type VT_UNKNOWN, please supply one
Warning : Function Item, parameter result_ has an unknown HANDLE type VT_DISPATCH, please supply one
Interface         : CR_IFieldObject
Warning : Function BackColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BackColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BorderColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BorderColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function TextColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function TextColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function Font, datatype VT_USERDEFINED | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function Font, datatype VT_USERDEFINED | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function Field, parameter result_ has an unknown HANDLE type VT_DISPATCH, please supply one
Interface         : CR_ITextObject
Warning : Function BackColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BackColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BorderColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function BorderColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function TextColor, datatype VT_UI4 | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function TextColor, datatype VT_UI4 used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function Font, datatype VT_USERDEFINED | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
Warning : Function Font, datatype VT_USERDEFINED | VT_BYREF used by parameter 1 is unsupported, using VT_BSTR instead
```

Picture 17: Script Import Warnings

The message-frame after the import shows many of these problems. Just notice them and if an error occur at test, change the signature.

Also not all handles are converted right. They are set to VT_UNKOWN instead to the right UNIFACE signature. These handles should be changed manually.

Optional Parameter

In other language there exist optional parameters. These are not directly supported by UNIFACE. In previous version you have to specified all parameters. In UNIFACE Eight you can use the "–" (without quotationmarks) as sign for optional. At runtime UNIFACE change this to "UT_OPTIONAL" and then the driver leave it away.

Depend on datatype there can offen problems (errors) occur. So in this case you should insert a valid value for that parameter.

Attribute and Name-Mapping (Set_ and Get_)

Attribut are offen available by PUT and GET method from the object. UNIFACE imported these standard methode and change them to operations with the prefix GET_ and SUT_ followed by the propertie-name.

2 Abbreviation

Abbreviation	Description
COM	Component Object Model
CORBA	Component Object Request Broker Architecture
EAI	Enterprise Application Integration
HOST	Mainframe
URB	Universal Request Broker
SDK	Software Development Kit
U8	UNIFACE Eight
WASV	Web Application Server
W2K	Windows 2000
WNT	Windows NT
MSS	Microsoft SQL Server
MS	Microsoft
CR	Crystal Report
IDL	Interface Definition Language
ODBC	Object Database Connector

3 Sources

UNIFACE: http://www.compuware.com/products/uniface/

Crystal Report: http://www.crystaldecision.com/products/crystalreport/

COM: http://com.microsoft.com/docs

UNIFACE-COM-Handbook: uniface-installation/help/docs/com.pdf

CR Technology Handbook: CR-Disc/docs/techref.pdf

CR Developer-Help Guide: CR-installation/Developer Files/Help/en/CrystalDevHelp.chm

Other UNIFACE Sources:

LISTSERVER http://groups.google.com/

$UUU http://www.puuu.org/

TURF http://www.webinacan/turf/

4 Disclaimer

The usual disclaimer…

This document is owned by his author (Thomas Young). All rights lay by him. Publicition is not allowed and need a confirmation by the author.
There is no garantuee about the statements and proc-coding. The document is only for discussion and not a complete technical description.

This document is the result of consultancy and not a direct work for Compuware or Crystal Report.

All product-names and company-name are names of their owner.

5 Picture sverzeichnis

Picture 9: OCX-Widget 1
Picture 10: Selection OCX 2
Picture 11: Widget Type (OCX) Properties 3
Picture 12: Field for OCX-Controlling 4
Picture 13: Signature for OCX 4
Picture 14: Parameter Mapping 5
Picture 15: Test OCX 7
Picture 16: COM throughCommandbutton 8
Picture 17: Fields for COM-Controlling 9
Picture 18: Creating Signature 9
Picture 19: COM-Signature 10
Picture 20: Selection COM-Objekte 11
Picture 21: Result COM-Import 12
Picture 22: COM-Import with Commandline 13
Picture 23: Signature and Subsystems 14
Picture 24: Multiple Operations 20
Picture 25: Script Import Warnings 21