Matthias MEUSBURGER
MSG Software
Avril-Mai 2001

Seagate
Crystal Reports™

Manuel d'intégration à Visual Basic

Ce document présentes diverses manières d'insérer un état dans une application Visual Basic, avec, à chaque fois, une méthode correspondante. Les différentes méthodes employées ici ne sont pas uniques. En effet, pour chaque cas, il existe différentes méthodes associées à différents modules Visual Basic fournis avec Crystal Reports. Cependant, j'ai essayé, pour chaque cas, de présenter la méthode la plus appropriée, et surtout, la plus abordable.

I L'intégration brute d'un rapport

Le terme « intégration brute » désigne ici le fait d'insérer dans une application Visual Basic un rapport tel qu'il a été conçu dans Crystal Reports, sans y apporter de modifications.

1) Utilisation de Crystal Reports Viewer

a) Il faut tout d'abord ajouter au projet les composants nécessaires à Crystal Reports qui sont :

Références :	Crystal Report Viewer Control	crviewer.oca
	Crystal Report 8 ActiveX Report Designer Run Time Library	craxdrt.dll
Contrôles:	Crystal Report Viewer Control	crviewer.dll
Objets à insérer :	Crystal Report Viewer Control	activex.dll

b) Ensuite, créer une feuille et y placer un composant CRViewer, qui sera normalement appelé "CRViewer1" par défaut.

c) Ensuite, il faut insérer le code suivant :

Déclaration :

```
Dim app As CRAXDRT.Application
Dim Report As CRAXDRT.Report

Private Sub Form_Load()
   ' Change le pointeur de souris en sablier
   Screen.MousePointer = vbHourglass
   ' Instancie l'objet application
   Set app = New CRAXDRT.Application
   ' Ouvre le rapport existant
   Set Report = app.OpenReport("C:\chemin\du\fichier\nom_de_fichier.rpt")
   ' Indique à l'objet de visualisation que les données viennent du rapport
   CRViewer1.ReportSource = Report
   ' Lance la visualisation du rapport
   CRViewer1.ViewReport
   ' Le pointeur de souris redevient normal une fois le chargement effectué
   Screen.MousePointer = vbDefault
End Sub
```

```
Private Sub Form_Resize()
  ' Cette partie permet de redimensionner la fenêtre de visualisation
 CRViewer1.Top = 0
 CRViewer1.Left = 0
 CRViewer1.Height = ScaleHeight
 CRViewer1.Width = ScaleWidth
  ' Note : cette partie restant constante quel que soit le cas de figure, elle ne sera pas réécrite
  dans les exemples suivants.
End Sub

Private Sub Form_Unload()
    ' On décharge la mémoire des objets instanciés précédemment
    Set app = Nothing
    Set Report = Nothing
    Set Me = Nothing
    ' Note : cette partie ne sera pas réécrite non plus dans les exemples suivants
End Sub
```

Cependant, lors d'une connexion sécurisée à la base de données, ce code générera un message d'erreur : « Le serveur n'a pas encore été ouvert ». Pour faire fonctionner l'application, il faut reprendre la partie Form_Load() de cette manière :

```
Private Sub Form_Load()
   Dim i As Integer

   ' Change le pointeur de souris en sablier
   Screen.MousePointer = vbHourglass
   ' Instancie l'objet application
   Set app = New CRAXDRT.Application
   ' Ouvre le rapport existant
   Set Report = app.OpenReport("C:\chemin\du\fichier\nom_de_fichier.rpt ")
   ' Pour chaque table du rapport
   For i = 1 To Report.Database.Tables.Count
            ' On connecte la table à la base de données
            Report.Database.Tables(i).SetLogOnInfo "Nom_du_serveur", _
            ["Nom_de_la_base"], ["Nom_de_user"], ["Mot_de_passe"]
            ' Note : les paramètres entre crochets sont optionnels
   Next i
   ' Indique à l'objet de visualisation que les données viennent du rapport
   CRViewer1.ReportSource = Report
   ' Lance la visualisation du rapport
   CRViewer1.ViewReport
   ' Le pointeur de souris redevient normal une fois le chargement effectué
   Screen.MousePointer = vbDefault
End Sub
```

II Intégration avec passage d'un jeu d'enregistrements au rapport

L'édition brute d'un rapport est certainement la plus évidente à mettre en place. Cependant, elle convient peu à la réalité. En effet, on souhaitera souvent éditer un état incomplet, c'est-à-dire filtré ou trié par l'utilisateur de l'application. Dans ce cas, il faut recourir à l'édition avec passage d'un jeu d'enregistrement.

Note : pour que les techniques décrites ci-dessous fonctionnent, il est impératif que l'option « *Enregistrer les données avec l'état* » du menu *fichier* de Crystal Reports soit désactivée.

1) Pour un rapport ne comportant pas de sous-état

a) Comme pour l'intégration brute, il faut avant tout ajouter les composants nécessaires à Crystal Reports, mais également les composants nécessaires à la création d'un recordset qui sont :

Références :	Microsoft ActiveX Data Objects 2.5 Library	msado15.dll
Contrôles :	Microsoft ADO Data Control 6.0	msadodc.ocx

b) Ensuite, créer une feuille et y placer un composant CRViewer, qui sera normalement appelé "CRViewer1" par défaut.

c) Ensuite, il faut insérer le code suivant :

Déclarations :

```
Dim app As New CRAXDRT.Application
Dim RecordSet As ADODB.Recordset          ' jeu d'enregistrements
Dim Report As New CRAXDRT.Report
Dim lStrSql As String                     ' Contient l'ordre SQL pour le recordset
Dim lStrConnect As String                 ' Contient la chaîne de connection à la base
Dim Connect As ADODB.Connection           ' Objet de type connection
```

```
Private Sub Form_Load()
    ' Change le pointeur de souris en sablier
    Screen.MousePointer = vbHourglass
    ' Définit la requête pour le jeu d'enregistrements
    lStrSql = "SELECT champ1, champ2, champN FROM table1, table2, tableN"
    ' Définit la chaine de connection
    lStrConnect = "Provider=Nom_du_provider;User ID=nom_du_user;" & _
    "Data Source=Nom_de_la_base; Password=Mot_de_passe"
    ' Instancie l'objet Connection
    Set Connect = new ADODB.Connection
    ' Ouvre la connection avec la chaine de connection
    Connect.open lStrConnect
    ' Instancie l'objet Recordset
    Set RecordSet = new ADODB.Recordset
```

```
    ' Initialise le recordset avec la requête et l'objet connection
    RecordSet.open lStrSql, Connect
    ' On peut éventuellement appliquer des tris ou filtres existant sur d'autres recordset
    RecordSet.sort = autreRecordSet.sort
    RecordSet.filter = autreRecordSet.filter
    ' Instancie l'objet application
    Set app = New CRAXDRT.Application
    ' Ouvre le fichier rapport
    Set Report = app.OpenReport("C:\chemin\du\fichier\nom_de_fichier.rpt ")
    ' Indique que la source du rapport est le recordset que l'on vient de définir
    Report.Database.SetDataSource RecordSet
    ' Indique à l'objet de visualisation que les données viennent du rapport
    CRViewer1.ReportSource = Report
    ' Lance la visualisation du rapport
    CRViewer1.ViewReport
    ' Le pointeur de souris redevient normal une fois le chargement effectué
    Screen.MousePointer = vbDefault
End Sub
```

2) Pour un rapport comportant un sous-état

Dans ce cas de figure, le type utilisé pour le rapport change. En effet, concernant les composants utilisés, Crystal Reports Engine 8 Object Library (cpeaut32.dll) vient remplacer Crystal Report 8 ActiveX Report Designer Run Time Library (craxdrt.dll).
En ce qui concerne les données, il faut créer deux jeux d'enregistrements. En effet, l'un est destiné à l'état principal, et l'autre au sous-état.
De plus, les composants utilisés étant différents de ceux utilisés précédemment, la création d'un objet CRViewer est inutile, puisque cette méthode commande elle-même l'ouverture d'une fenêtre de visualisation du rapport.

```
Déclarations :
Dim lStrSql As String
Dim RecordSet As ADODB.RecordSet
Dim RecordSetSubReport As ADODB.RecordSet
Dim app As New CRPEAuto.Application
Dim Report As CRPEAuto.Report
Dim Subreport As CRPEAuto.Report
Dim lStrConnect As String
Dim Connect As ADODB.Connection

Private Sub Form_Load()
    ' Change le pointeur de souris en sablier
    Screen.MousePointer = vbHourglass
    ' Définit la requête pour le jeu d'enregistrements
    lStrSql = "SELECT champ1, champ2, champN FROM table1, table2, tableN"
    ' Définit la chaine de connection
```

```
    lStrConnect = "Provider=Nom_du_provider;User ID=nom_du_user;" & _
    "Data Source=Nom_de_la_base; Password=Mot_de_passe"
    ' Instancie l'objet Connection
    Set Connect = new ADODB.Connection
    ' Ouvre la connection avec la chaine de connection
    Connect.open lStrConnect
    ' Instancie l'objet Recordset
    Set RecordSet = new ADODB.Recordset
    ' Initialise le recordset avec la requête et l'objet connection
    RecordSet.open lStrSql, Connect
    ' Definition de la requete pour le jeu d'enregistrement destiné au sous état
    lStrSql = "SELECT champ1, champ2, champN FROM table1, table2, tableN"
    ' Instancie l'objet Recordset
    Set ProductRecordSetSubReport = New ADODB.RecordSet
    ' Initialise le recordset avec la requête et l'objet connection
    ProductRecordSetSubReport.Open lStrSql, Connect
    ' Ouvre le fichier rapport
    Set Report = app.OpenReport("C:\chemin\du\fichier\nom_de_fichier.rpt ")
    ' on utilise la méthode SetPrivateData pour affecter le recordset au rapport
    Call Report.Database.Tables.Item(1).SetPrivateData(3, RecordSet)
    ' Connecte les tables à la base de données
    Report.Database.Tables(1).SetLogOnInfo "Nom_du_serveur", ["Nom_de_la_base"], _
    ["Nom_de_user"], ["Mot_de_passe"]
    ' Ouvre le sous-état
    Set Subreport = Report.OpenSubreport("nom_du_sous-état")
    ' on affecte le recordset au sous-état
    Call Subreport.Database.Tables.Item(1).SetPrivateData(3, RecordSetSubReport)
    ' Lance la visualisation du rapport
    Report.Preview
    ' Le pointeur de souris redevient normal une fois le chargement effectué
    Screen.MousePointer = vbDefault
End Sub
```

Note :
Le premier argument de la méthode SetPrivateData indique le type de valeur à transmettre. A l'heure actuelle, la seule valeur possible est « 3 ».

III Passage de champs dynamiques

Il peut être utile de pouvoir affecter au rapport des champs dont la valeur est définie à l'exécution de l'application Visual Basic. Pour cela, il faut que le rapport comporte autant de champs paramètre (vides) que l'on veut insérer de champs dynamiques.

Note : il n'est pas possible de passer directement une requête SQL à un champ paramètre. Pour cela, il faut créer sa requête comme vu précédemment, éventuellement filtrer les résultats, et passer une chaîne de caractères au champ paramètre.

1) Pour un rapport ne comportant pas de sous-état

La méthode à suivre est la même pour une édition brute que pour une édition avec passage d'un jeu d'enregistrements.
Dans les deux cas, le code indiqué ci-dessous doit être appliqué juste avant l'édition du rapport.
Le composant utilisé est Crystal Report 8 ActiveX Report Designer Run Time Library (craxdrt.dll)

```
Déclarations :
Dim DynFieldStr As String
Dim CRXPFDs As CRAXDRT.ParameterFieldDefinitions
Dim CRXPFD  As CRAXDRT.ParameterFieldDefinition

Private Sub Form_Load()
    ' Définition du rapport et de sa source de données
    ' …
    ' Définiton de la chaîne de caractères destinée à être inserée dans le rapport
    DynFieldStr = "Votre_chaine_de_caractères"
    ' Récupère la collection des champs paramètre du rapport
    Set CRXPFDs = Report.ParameterFields
    ' Récupère le premier champ paramètre de la collection
    Set CRXPFD = CRXPFDs.Item(1)
    ' Ajoute la chaîne de caractères au champ paramètre
   CRXPFD.AddCurrentValue (DynFieldStr)

    ' De même, pour un autre champ :
    DynFieldStr = "Une_autre_chaine_de_caractères"
    Set CRXPFD = CRXPFDs.Item(2)
    CRXPFD.AddCurrentValue (DynFieldStr)
End Sub
```

2) Pour un rapport comportant un sous-état

a) Si aucun des champs dynamiques ne se trouve dans le sous-état

Dans ce cas, la méthode diffère peu de celle concernant les rapports ne comportants pas de sous-état. Seules les types utilisés et les nom des fonctions changent ; la procédure reste la même.

```
Déclarations :
Dim DynFieldStr As String
Dim CRPEPFDs As CRPEAuto.ParameterFieldDefinitions
Dim CRPEPFD  As CRPEAuto.ParameterFieldDefinition

Private Sub Form_Load()
' Reprendre la méthode précédente et remplacer :
' Set CRXPFDs = Report.ParameterFields
' Set CRXPFD = CRXPFDs.Item(1)
' CRXPFD.AddCurrentValue (DynFieldStr)
' par :
Set CRPEPFDs = Report.ParameterFields
Set CRPEPFD = CRPEPFDs.Item(1)
CRPEPFD.SetCurrentValue (DynFieldStr)

' De même, pour un autre champ :
DynFieldStr = " Une_autre_chaine_de_caractères"
Set CRPEPFD = CRPEPFDs.Item(2)
CRPEPFD.SetCurrentValue (DynFieldStr)
```

b) Si un des champs dynamiques se trouve dans le sous-état

Dans ce cas, on ne peut plus accéder aux champs par leur numéro d'index, mais par une boucle parcourant les différents objets de la collection pour retrouver un champ par son nom.

```
Déclarations :
Dim DynFieldStr As String
Dim CRPEPFDs As CRPEAuto.ParameterFieldDefinitions
Dim CRPEPFD  As CRPEAuto.ParameterFieldDefinition

Private Sub Form_Load()
' Définition du rapport et de sa source de données
' …
' Récupère la collection des champs paramètre du rapport
Set CRPEPFDs = Report.ParameterFields
```

```
' Pour chaque champ dans la collection
For Each CRPEPFD In CRPEPFDs
	' Suivant que le nom du champ est
	Select Case CRPEPFD.ParameterFieldName
		' Le nom du paramètre dans l'état
		Case "dynEtat"
			' On affecte une chaine de caractères à ce champ
			DynFieldStr = "une_chaîne_de_caractères"
			CRPEPFD.SetCurrentValue (DynFieldStr)
		' Le nom du paramètre dans le sous-état
		Case "dynSousEtat"
			' On affecte une autre chaîne de caractères à ce champ
			DynFieldStr = "une_autre_chaîne_de_caractères"
			CRPEPFD.SetCurrentValue (DynFieldStr)
	End Select
Next
```

Note : Bien dans cette méthode ne soit obligatoire que dans le cas où un champ paramètre se trouve dans le sous-état, elle peut également être mise en œuvre dans les cas précédents, c'est-à-dire si le champ paramètre ne se trouve pas dans le sous-état ou si le rapport ne possède pas de sous-état. Si le nombre de champs paramètre est faible, la méthode des index est plus simple à utiliser. Cependant, par souci de lisibilité du code, la méthode de recherche par nom est conseillée dès lors que le nombre de champs paramètre augmente.

3) Passage de valeurs multiples

Il est également possible de passer plusieurs valeurs à un seul champ paramètre. Pour cela, la propriété EnableMultipleValues doit être mise à Vrai. De plus, le champ paramètre du rapport ne doit plus être vide, mais doit comporter une formule conditionnant l'affichage.

```
Déclarations :
Dim DynFieldStr As String
Dim CRXPFDs As CRAXDRT.ParameterFieldDefinitions
Dim CRXPFD  As CRAXDRT.ParameterFieldDefinition

Private Sub Form_Load()
    ' Définition du rapport et de sa source de données
    ' …
    ' Définition de la chaîne de caractères destinée à être inserée dans le rapport
    DynFieldStr = "Votre_chaine_de_caractères"
    ' Récupère la collection des champs paramètre du rapport
    Set CRXPFDs = Report.ParameterFields
    ' Récupère le premier champ paramètre de la collection
    Set CRXPFD = CRXPFDs.Item(1)
    ' Autorise les valeurs multiples pour un champ
    CRXPFD.EnableMultipleValues = True
    ' Ajoute la chaîne de caractères au champ paramètre
    CRXPFD.AddCurrentValue (DynFieldStr)
    ' Définition d'une autre chaîne de caractères
    DynFieldStr = "Une_autre_chaine_de_caractères"
    ' Ajoute la deuxième chaîne de caractères au même champ
    CRXPFD.AddCurrentValue (DynFieldStr)
    ' …
```

En ce qui concerne la formule à insérer dans le fichier rapport, elle doit conditionner l'affichage des valeurs. Voici un exemple qui affiche dans le champ paramètre (appelé « paramDyn ») la première chaîne de caractères quand la valeur d'un autre champ de l'état appelé « Nom_du_champ » est impair et les deux chaînes de caractères quand la valeur de « Nom_du_champ » est pair.

```
If ({Nom_du_champ}) mod 2 <> 0 then
{?paramDyn}[1]
else
 ({?paramDyn}[1] + " " + {?paramDyn}[2])
```

IV Notes Annexes

1) Crystal Reports Engine 8 Object Library (CRPEAuto)

Comme nous l'avons vu dans les exemples, cette méthode commande elle-même l'ouverture d'une nouvelle fenêtre pour afficher le rapport.
Pour implémenter cette méthode dans une application, j'avais d'abord pensé mettre le code directement sur le clic du bouton. Cependant, la fenêtre générée de cette manière disparaît immédiatement après le chargement du rapport.
Il faut donc placer le code sur une feuille qu'on ne rendra pas visible ("Load nom_de_la_feuille" et non pas "nom_de_la_feuille.show")
Ainsi, seule la feuille générée automatiquement apparaitra lors du chargement.
Cependant, il faut veiller à décharger la fenêtre que l'on n'a pas fait apparaître au moyen de la commande "Unload nom_de_la_feuille», sinon, lors d'une seconde visualisation, le rapport ne se chargera pas.

Pour résumer le chargement de la feuille contenant le code :

```
Private Sub CommandButton_Click()
    Load nom_de_la_feuille
    Unload nom_de_la_feuille
End Sub
```

Note : Pour que cette technique marche, il ne faut pas décharger les objets dans la procédure form_unload() de la feuille.

TABLE DES MATIERES :

TABLE DES MATIERES DES ANNEXES :