

Matthias MEUSBURGER  
MSG Software  
Avril-Mai 2001

Comparatif :

*Crystal Reports vs Report Maker*

Partie 2 :

L'intégration à *Visual Basic*

## 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 le logiciel propriétaire de ce rapport, sans y apporter aucune modification.

### 1) Installation des composants

Après installation des deux logiciels, les composants nécessaires à l'intégration de rapports dans Visual Basic se retrouvent dans les menus de ce dernier et il suffit alors de les sélectionner pour en disposer dans la barre d'outils.

### 2) Intégration des composants dans l'application

#### **Crystal Reports :**

Dans Crystal Reports, un concepteur permet d'automatiser la tâche d'intégration d'un rapport à Visual Basic, bien que dans mon cas, la connexion au serveur ait été infructueuse en utilisant cet assistant.

Il est également possible de procéder manuellement en insérant l'objet « CRViewer » qui permet la visualisation du rapport, mais dans ce cas, la saisie de quelques lignes de code est inéluctable.

Note :

Lors d'un chargement relativement long, l'utilisateur peut-être tenté de redimensionner la feuille contenant l'objet de visualisation avant l'apparition du rapport. Si tel est le cas, la taille dudit objet ne sera pas adaptée à la nouvelle taille de la fenêtre tant que l'utilisateur n'aura pas provoqué un nouveau rafraîchissement en redimensionnant la fenêtre.

#### **Report Maker :**

L'intégration se passe ici sans aucun problème. En effet, il suffit d'insérer l'objet Report Maker dans une feuille et de configurer les paramètres de connexion dans une boîte de dialogue. Il n'y a aucune ligne de code à saisir, la configuration est rapide et intuitive.

## II L'intégration avec passage d'un jeu d'enregistrements au rapport

### 1) Conversion du fichier rapport

Dans le cas où le rapport n'avait pas été prévu pour être intégré à Visual Basic de manière interactive, il est nécessaire de le convertir. En effet, un rapport utilisant le connecteur ODBC ne peut recevoir de jeu d'enregistrement. Il doit pour cela utiliser le connecteur ADO (Active Data Object) ou tout autre connecteur de type Active Data.

#### **Crystal Reports :**

Le logiciel possède une fonction permettant de convertir le pilote de base de données. Cependant, après plusieurs heures passées à essayer de comprendre les tenants et les aboutissants de cette conversion, n'obtenant pas le résultat escompté, je me suis rendu compte que la conversion d'un rapport ODBC vers un rapport ADO suppose deux choses :

- Le rapport ne fait pas entrer en jeu des champs de plusieurs tables.
- Le rapport ne comporte pas de sous-état.

Si le rapport ne respecte pas une de ces règles, la seule solution est alors de le recréer. L'utilisation de plusieurs tables ou de sous-états dans un rapport étant relativement fréquente, cette fonction de conversion du pilote de base de donnée devient quasiment inutile !

#### **Report Maker :**

Ici, la conversion d'un rapport ODBC en un rapport ADO ne pose quasiment aucun problème. En effet, lors de la conversion, si le logiciel ne reconnaît pas un champ à partir de la nouvelle source de données, il faut procéder à une opération de mappage, c'est-à-dire faire correspondre les champs du rapport avec les champs de la nouvelle source de données.

### 2) Mise en œuvre

#### **Crystal Reports :**

Cette opération est relativement simple. En effet, elle se décompose en trois parties :

- création ou récupération d'un jeu d'enregistrement.
- ouverture du rapport existant
- redirection de la source de données du rapport au profit du jeu d'enregistrement.

#### Notes :

- Pour que les éventuels tris ou filtres du jeu d'enregistrement s'appliquent lors de la visualisation de l'état, il faut que les options de tris ou de filtre soient désactivées au sein du fichier rapport, sinon, ceux-ci seront pris en compte prioritairement. Il faut notamment penser, lors de la création d'un groupe, à éditer ses propriétés pour que le champ « pilote » du groupe soit trié dans l'ordre d'origine.
- Le fichier rapport doit être enregistré sans le jeu de données correspondant.

#### **Report Maker :**

Contrairement à Crystal Reports, la simple conversion du rapport de ODBC à ADO ne suffit pas pour pouvoir envoyer des données à l'état. En effet, selon l'aide de programmation de Report Maker, il faut pour cela que :

- le bloc "détail" du rapport soit auto-bouclant
- les champs du rapport soient de type "programmé"

Il faudrait ensuite au niveau de l'application boucler au sein de l'événement "détail" sur chaque bloc, et sélectionner les champs programmés pour leur affecter une valeur.

Malgré de nombreux essais, j'ai été dans l'incapacité de mettre en œuvre cette procédure, faute de précisions. En effet, l'aide ne propose ni données supplémentaires ni exemple pour ce cas de figure.

#### Note :

Bien qu'il soit possible d'utiliser deux interfaces de programmation différentes pour intégrer un fichier Report Maker dans Visual Basic, seule la première, l'interface ActiveX, a fonctionné. En effet, l'utilisation de l'interface DLL provoque un message d'erreur de type « Violation d'accès dans le module ... »

### III Qualité de l'aide

L'évaluation de l'aide est réalisée sur trois critères :

- 1) L'aide fournie avec le logiciel : exhaustivité et clarté
- 2) L'aide disponible sur Internet
- 3) Les exemples fournis

#### **Crystal Reports :**

- 1) L'aide fournie avec le logiciel : exhaustivité et clarté

L'aide de Crystal Reports est très complète. En effet, la documentation fournie avec le logiciel existe dans beaucoup de formats :

- HTML (\*.HTM)
- Acrobat Reader (\*.PDF)
- HELP (\*.HLP)
- Word (\*.DOC)
- HTML compilé (\*.CHM)

Ces fichiers couvrent beaucoup de domaines, tout en proposant des approches par niveau. Chacun y trouve son compte, de l'utilisateur débutant au programmeur averti. Cependant, si les aides concernant l'utilisation du logiciel sont en français, toutes les aides concernant la programmation sont en anglais.

L'utilisateur de base trouvera principalement un guide de démarrage (HTML), un guide de l'utilisateur (PDF) ainsi qu'une aide en ligne (CHM).

Le développeur quant à lui aura à sa disposition au format PDF un guide du développeur, ainsi que plus de 1100 pages de références techniques !

Cependant, cette profusion de documentation peut nuire quelque peu. En effet, un index des aides disponibles eût été le bienvenu, car on s'y perd facilement. Par exemple, le guide du développeur existe au format PDF et au format HELP, et curieusement, celui au format HELP est beaucoup plus complet que l'autre, alors que l'on pourrait croire que les deux sont équivalents !

- 2) L'aide disponible sur Internet (<http://www.crystaldecisions.net/products/crystalreports/>)

Rien à redire : le site de Crystal Reports est très riche en documentation annexe ainsi qu'en faq (Frequently Asked Questions), qui, bien qu'en anglais, constituent une aide précieuse.

### 3) Les exemples fournis

Les exemples de programmes Visual Basic fournis avec le logiciel sont relativement nombreux (19), mais malheureusement, le cas précis du passage d'un recordset par ADO n'était pas inclus dans ces exemples.

Finalement, la quantité de documentation disponible sur Crystal Reports est impressionnante, mais retrouver une information précise dans ce magma de documents peut se révéler fastidieux. En effet, j'ai dû passer plusieurs jours à lire des centaines de pages pour trouver comment passer un jeu d'enregistrement à un objet ADO, en piochant des éléments de réponse au hasard des documents trouvés. La principale faiblesse de l'aide de Crystal Reports est très certainement son manque d'organisation. Cependant, l'exhaustivité de l'aide est très appréciable, puisqu'on peut trouver une explication sur tout ce qu'il est possible de faire avec Crystal Reports.

## **Report Maker :**

### 1) L'aide fournie avec le logiciel : exhaustivité et clarté

L'aide fournie avec Report Maker présente un avantage par rapport à celle de Crystal Report : tout est en français, ce qui peut se révéler appréciable pour la compréhension de sujets un peu délicats.

Cependant, si l'aide concernant l'utilisation du logiciel est correcte, celle concernant la programmation m'a paru très légère. En effet, bien que toutes les fonctions ainsi que leur description soient disponibles dans un index, seules les procédures d'implémentation basique sont décrites, et le tout manque vraiment d'exemples concrets.

### 2) L'aide disponible sur Internet ([www.synactis.com/fr](http://www.synactis.com/fr))

Le site de Synactis ne m'a été d'aucun secours lors de ma recherche d'information ! En effet, aucune documentation supplémentaire n'est disponible et la foire aux questions est rachitique. De plus le site est doté d'une fonction de recherche...qui ne fonctionne pas !

### 2) Les exemples fournis

Les exemples fournis sont au nombre de quatre :

- le premier illustre l'utilisation de l'interface de programmation ActiveX
  - le second illustre l'utilisation de l'interface de programmation DLL
  - le troisième illustre la visualisation d'un rapport sans utiliser le Viewer par défaut
  - le quatrième illustre le passage de paramètres au rapport
- ...mais aucun ne marche !

En effet, l'exemple concernant la DLL provoque un message d'erreur de type « Violation d'accès au module xxxx.dll » et dans les autres exemples, les objets utilisés dans le code n'ont pas été créés !

## IV Codes source et explications détaillées

### 1) Intégration brute

#### **Crystal Reports :**

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
```

```
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
```

### **Report Maker :**

a) Il faut tout d'abord ajouter au projet le composant nécessaire à ReportMaker qui est le contrôle Synactis Report Maker 3 (rmaker30.ocx).

b) Ensuite, placer sur une feuille l'objet RptMaker, qui sera normalement appelé rptMaker1.

c) Editer les propriétés de cet objet, et indiquer les paramètres de connexion ainsi que le nom et le chemin du rapport à ouvrir.

d) Finalement, dans le code d'un bouton ou du form\_load, rajouter simplement :  
RptMaker1.Preview

*Alternative à l'édition des propriétés de l'objet RptMaker :*

On peut également utiliser des lignes de code à la place de la boîte de dialogue pour définir les paramètres de connexion :

```
Private Sub Form_Load()  
    ' On spécifie le chemin vers le fichier à ouvrir  
    RptMaker1.ReportDir = "c:\chemin\du\fichier"  
    ' On spécifie le nom du fichier à ouvrir  
    RptMaker1.ReportName = "Nom_du_fichier.rpm"  
    ' On spécifie les paramètres de connexion  
    RptMaker1.SetLoginParams "Nom_de_la_base", "Nom_de_user", "Mot_de_passe"  
    ' On lance la visualisation du rapport  
    RptMaker1.Preview  
  
End Sub
```

#### Notes :

- Dans Crystal Report, si on ne spécifie pas de chemin, mais juste un nom de fichier, le fichier en question sera recherché dans le répertoire courant, tandis que dans ReportMaker, il faut absolument spécifier un répertoire, quitte à ce que ce soit "." (répertoire courant)
- Il n'est pas possible de spécifier les paramètres de connexion concernant le nom d'utilisateur et le mot de passe dans la boîte de dialogue des propriétés de l'objet ReportMaker.  
Ainsi, pour éviter l'apparition d'une boîte de dialogue demandant ces paramètres lors de l'exécution du programme, ils devront être spécifiés dans le code.
- Il est possible de surdéfinir le type de connecteur ainsi que la base de données à utiliser par rapport à ce qui est défini dans le fichier rapport en utilisant respectivement les fonctions "DatabaseName" et "ConnectorName".

## 2) Intégration avec passage d'un jeu d'enregistrements au rapport

### **Crystal Reports :**

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

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 chaîne 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 chaîne 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

```

Il convient ensuite de libérer la mémoire allouée pour le jeu d'enregistrement :

```

Private Sub Form_Unload(Cancel As Integer)
    Set RecordSet = Nothing
End Sub

```

## TABLE DES MATIERES :

I	L'intégration brute d'un rapport .....	27
	1) Installation des composants .....	27
	2) Intégration des composants dans l'application .....	27
II	L'intégration avec passage d'un jeu d'enregistrements au rapport .....	28
	1) Conversion du fichier rapport .....	28
	2) Mise en œuvre .....	28
III	Qualité de l'aide .....	30
	1) L'aide fournie avec le logiciel : exhaustivité et clarté .....	30
	2) L'aide disponible sur Internet .....	30
	3) Les exemples fournis .....	30
IV	Codes source et explications détaillées .....	32
	1) Intégration brute .....	32
	2) Intégration avec passage d'un jeu d'enregistrements au rapport .....	35