



CRM Setup Factory Installer V 3.0

Developers Guide

Who Should Read This Guide

This guide is for ACCPAC CRM solution providers and developers. We assume that you have experience using:

- Microsoft Visual Studio .NET
- MS SQL Server 2000+
- Visual Basic .NET or Visual C#

How this guide is organized

This guide is designed to show you how to utilize the “CRMSetupFactory.dll” in your own projects.

This will be accomplished through a series of step by step guides, diagrams and text that should approximate the environment you will experience.

Table of Contents

What is “CRM Setup Factory”?	4
How to use “CRM Setup Factory”	4
Referencing the class library.....	4
Instantiating the “Parser” class	6
Setting up the “Parser” class	7
Running the process.....	9
Try It Out	10
Accessing log files	10
F.A.Q. (Frequently Asked Questions)	11
How Do I Embed Resources?.....	11
How Do I Use A StreamWriter?.....	12
Code Samples	13
Visual Basic .NET	13
Visual C#	14

What is “CRM Setup Factory”?

CRM Setup Factory is a class library written in Visual Basic .NET using Microsoft Visual Studio .NET 2003. CRM Setup Factory was designed to give developers a way to extract metadata from a given CRM database and merge it into another with as little effort as possible.

This is a two part process, the first of which is to use “CRM Setup Factory: Extractor” (a windows based application that connects to the source CRM database and extracts the data into script files with the .c21 extension). “CRM Setup Factory: Installer” (CRMSetupFactory.dll) is the second part which actually parses the scripts and alters the destination CRM database.

How to use “CRM Setup Factory”

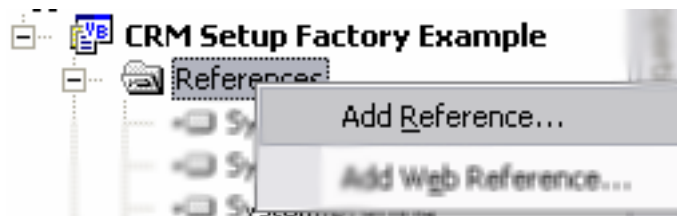
This section will explain how to use “CRM Setup Factory” in a project. We will accomplish this by creating an example project in Visual Studio .NET 2003 using Visual Basic .NET. We could use C# or any other .NET language but for this example we will just stick with VB.NET.

Referencing the class library

In order to use CRM Setup Factory in our project we first need to add a reference to the “CRMSetupFactory.dll”. Doing this allows us to access the objects and methods encapsulated in the class library.

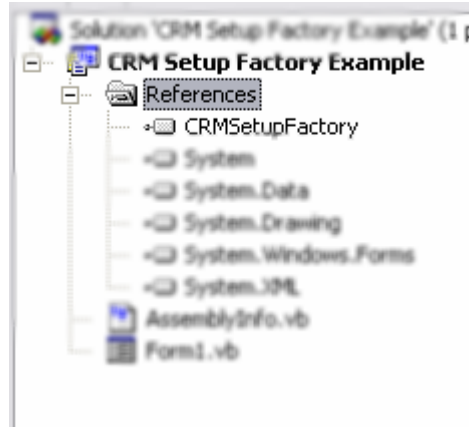
The first thing to do is create a new solution in Visual Studio .NET 2003, for this example we will call this solution “CRM Setup Factory Example”. After we have created a new solution we open the solution explorer, if you cannot see the solution explorer click the “View” menu then click on “Solution Explorer”.

Right click the “References” node and click “Add Reference...”



The “Add Reference” window will display. Select the browse button and navigate to the directory containing “CRMSetupFactory.dll”. Select “CRMSetupFactory.dll” and click the “Open” button then click “Add”. We will now see the reference as illustrated in the image below.

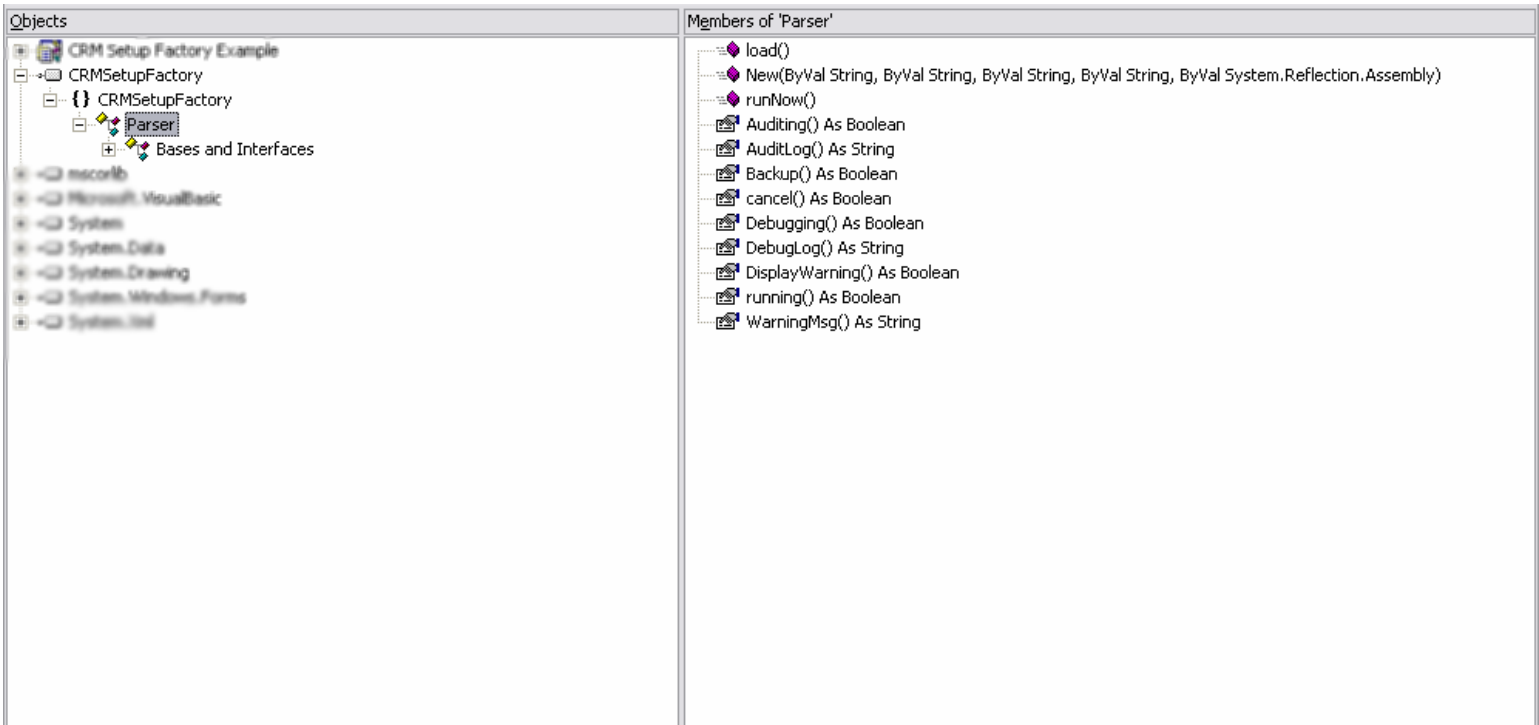
CRM Setup Factory V 1.0 Developers Guide



The last thing to do now is add the “Imports” (or “using” for C#) statement so we can access this object from our code.

```
1 | Imports CRMSetupFactory
```

After we have added the reference to the project and inserted our “Imports” statement we can take a look at the exposed members of CRM Setup Factory by using Visual Studios “Object Browser”. Double click on the “CRMSetupFactory” item under “References” in the solution explorer to open the object browser. Then find the “CRMSetupFactory” node and we expand the tree view until we see “Parser”. Selecting parser in the “Objects” panel allows us to see the exposed members for that object in the “Members of ‘Parser’” panel.



As we can see from the image above, the only class exposed by CRMSetupFactory is “Parser”; this is the workhorse of CRMSetupFactory and the object we will be using to parse our scripts.

There are several properties pertaining to how “CRM Setup Factory” will interact with the user and whether or not to create Debugging/Audit logs. We won’t cover these properties in any detail now we will leave that for the “Setting up the ‘Parser’ class” section later for now it will suffice to say that these properties exist. After we have looked at the members of the “Parser” class it’s time to instantiate it.

Instantiating the “Parser” class

To use CRM Setup Factory we need to create an instance of the object and call the public functions and subs. As you may have noticed from the previous section the only exposed class inside CRM Setup Factory is “Parser”. This is the class that we will need to instantiate in order to parse the scripts into the destination CRM database.

Instantiating the Parser class is a fairly simple process, we simply call the new method passing in a few parameters and we are ready to go. First things first though let’s add a button to our form so we have a way to test our instantiation of the Parser class.

To add a new button to your form, simply double click “Button” in the toolbar. After we place a button on our form, we double click it to take us to the event handler for that button’s click event. Here is where we will declare our Parser object; we can do this as follows:

```
Dim Assem As Reflection.Assembly
Assem = System.Reflection.Assembly.GetExecutingAssembly
Dim SetupFactory As New Parser("servername", "databasename", "username", "password", Assem)
```

Let’s take a look at our code and see how it works. The first thing we do here is create an assembly object.

```
Dim Assem As Reflection.Assembly
```

We need to create an assembly object because the Parser class expects an assembly with the .c21 scripts files embedded in it to be passed as an argument on the call to “New”. This doesn’t have to be the assembly that is calling parser, but just an assembly with the script files embedded. In our example we will just assume that the calling assembly has the scripts embedded. If you are unsure on how to embed resources you can refer to the F.A.Q. section at the end of this guide.

CRM Setup Factory V 1.0 Developers Guide

Next we instantiate our assembly object by setting it as a reference to the executing assembly.

```
Assem = System.Reflection.Assembly.GetExecutingAssembly
```

This is using the “System.Reflection.Assembly” namespaces “GetExecutingAssembly()” method to retrieve a reference to the assembly that is executing. In our example this will be the “CRM Setup Factory Example.exe” executable that is generated when we build our project.

Next we create and instantiate our Parser object.

```
Dim SetupFactory As New Parser(<servername>, <databasename>, <username>, <password>, Assem)
```

This line is simply creating a Parser object and naming it “SetupFactory”. The table below explains the parameters passed into the “New” method of Parser.

servername	The name of the SQL Server on the network
databasename	The name of the CRM Database on the SQL Server.
username	The username with at least “Database Creators” rights
password	The password associated with that SQL Server account
Assem	A Reference to the assembly containing the .c21 script files as embedded resources

Now that we have the object instantiated its simply a matter of setting the appropriate properties and calling the “RunNow()” method. In the next section we will cover what all the properties do and how to use them.

Setting up the “Parser” class

Usually the default properties are going to be sufficient for us, but in some cases we may want to enable the Debug/Audit logs or maybe even change the message that is displayed to the user. To accomplish this we need to change some of the properties from their default values.

With our Parser class instantiated we are now ready to set the properties and begin the process. The table below lists all of the available properties and explains what they do and how to use them.

CRM Setup Factory V 1.0 Developers Guide

Auditing	Boolean value indicating whether or not to maintain an audit log.	Set this value to True to enable the audit log. This is false by default. Ex: SetupFactory.Auditing = True
AuditLog	String value containing the audit log	This value is read only and contains the audit log as a string Ex: dim AuditLog as string = SetupFactory.AuditLog
Backup	Boolean value indicating whether to display the option to backup the CRM Database before beginning the process	This value is true by default. Ex: SetupFactory.Backup = False
cancel	Boolean value indicating that the user has chosen to cancel the process when the warning message was displayed. This property is only checked once right before the process begins	This value is set to false by default but if the user clicks "Cancel" in the dialog that asks them to backup it gets set to True which cancels the process. Ex: SetupFactory.cacel = True
Debugging	Boolean value indicating whether or not to maintain a debug log	Set this value to True to enable the debug log. This is False by default. Ex: SetupFactory.Debugging = True
DebugLog	String value containing the debug log	This value is read only and contains the debug log as a string Ex: dim DebugLog as string = SetupFactory.DebugLog
DisplayWarning	Boolean value indicating whether or not to display the database backup warning message to the user.	Set to True by default, if this value is set to False the Backup and WarningMsg properties are ignored. Ex: SetupFactory.DisplayWarning = False
running	Boolean value indicating the status of the process.	This value is read only. Ex: Dim status as Boolean = SetupFactory.running
WarningMsg	String value containing the non default message to display to the user in the warning dialog.	Assigning a value to this property will result in the assigned value being displayed to the user instead of the default warning. Ex: SetupFactory.WarningMsg = "Please click Yes to back up your database.

CRM Setup Factory V 1.0 Developers Guide

For our example we will just set the most commonly used properties

```
SetupFactory.Auditing = True
SetupFactory.Debugging = True
SetupFactory.Backup = True
SetupFactory.DisplayWarning = True
SetupFactory.WarningMsg = "Click Yes To Backup Your Database"
```

Let's take a look at the preceding code and see what it is doing.

First we set the “Auditing” and “Debugging” properties to “True” to indicate that we want to maintain both of the logs. Then we indicate that we would like to give the user the option to create a backup of their existing CRM database before continuing with the installation by setting the “Backup” property to “True”. Next we set the “DisplayWarning” property to “True” to ensure that the warning message is displayed. Finally we set our custom warning message by using the “WarningMsg” property.

Now we are ready to begin the process, we will cover this in the next section.

Running the process

Now that we have our “Parser” object instantiated (as SetupFactory) it's time to begin the process that will parse the scripts and alter the destination CRM database accordingly.

The first thing we need to do is call the “load()” method on our SetupFactory object. This simply prepares the script to be run and displays the warning message to the user. If we are going to catch any errors and attempt to cancel the process now is the time to do it by setting the “cancel” property to “True”.

```
SetupFactory.load()
```

The second and last thing to do is call the “runNow()” method. This method will do two things the first of which is display a dialog informing the user of the current progress. Secondly it will run the actual parsing of the scripts on a new thread. If for some reason the user closes the dialog displaying the status the second thread will still run until it has completed.

```
SetupFactory.runNow()
```

You can check the status of the process by using the “running” property, this will get set to “False” when the process has completed.

Try It Out

It is now time to try out our code and see if it works. To accomplish this we will simply build and run our solution. Once our solution has been built and is running we simply click on the button that we placed on the form. There are no scripts embedded in the executable that we are passing into “Parser” but that’s ok we are simply trying to see what the user will experience. You will see the warning message that we set

`“Click Yes To Backup Your Database”`

By clicking yes we are presented with 2 dialogs, the first of which asks the user for a directory where they would like to store the backup and the second asks the user for a name to use for the back up (the default is “CRM.BAK”).

Next we will see a dialog, (however brief), that will display the status to the user.

Now that we have run our process the only thing left to do is check for an error log and check our Debugging and Auditing log files if needed.

Accessing log files

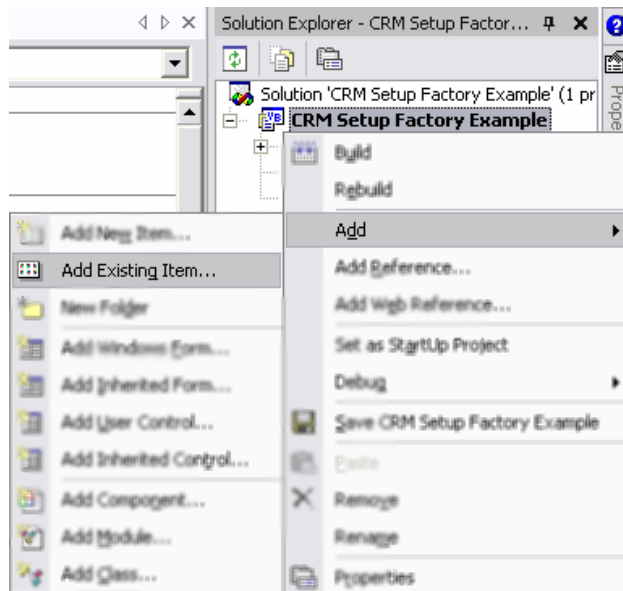
If there was an error during the process of altering the source database “CRM Setup Factory” will create an error log in the directory that it was run from. In our example there wouldn’t be any errors seeing as how there weren’t any scripts to parse (this itself is not considered an error), so it is suffice to say that you could check the directory for “ErrorLog.txt”.

The writing out of the Debug/Audit logs is left solely to the programmer, so if we wanted to view the Debug/Audit logs we would have to either step through our code and wait for the process to complete then hover our mouse over the variable in Visual Studio or we could use a StreamWriter to write the log out to a file of our choice. There is no right or wrong way, when it comes to choosing how to view these logs, just remember that these logs can get fairly big so I recommend the latter.

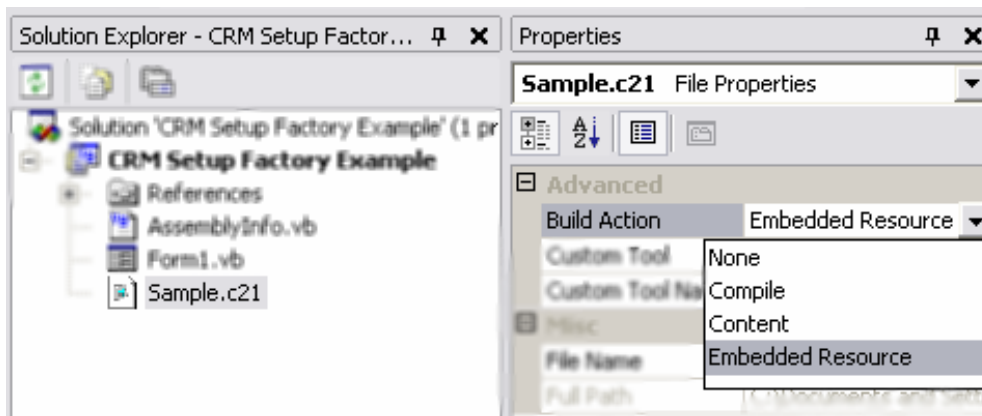
F.A.Q. (Frequently Asked Questions)

How Do I Embed Resources?

Embedding resources is a relatively simple process. First we need to establish which files we would like to embed, in the case of CRM Setup Factory these will be the .c21 script files. We simply add the files to our project by right clicking the projects name and selecting “Add Existing Item”



This will open a dialog and allow you to browse to the file(s) you would like to add, for this example I have chosen to use “Sample.c21”. After we have added our file(s) we simply select the file and set the “Build Action” property to embed.



That's it the file is now an embedded resource. There are many different ways to go about embedding resources but this is the easiest way that I have found.

For more information on embedding resources visit

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbtskaddingresourcestoyourapplication.asp>

How Do I Use A StreamWriter?

StreamWriters and StreamReaders are a fairly advanced topic and I wont cover them in much detail here except how to use them to output the Debug/Audit logs.

We begin by instantiating a StreaWriter object that will write out our logs.

```
Dim SW as New System.IO.StreamWriter(<string containing path  
to file>, True)
```

Lets go over the parameters:

String containing path to file	This is simply a string containing a directory on your machine including file extension Ex: "C:\AuditLog.txt"
True	This is a Boolean value indicating that if the file exists to append to it rather then create a new file.

All that's left to do now is write out the text to the file.

```
SW.Write(SetupFactory.AuditLog)  
SW.Close()
```

We call the Write() method to write out the string in the AuditLog property of the SetupFactory object then we call the Close() method to release control of the file.

For more information on StreamWriter/Readers visit

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemiostreamwriterclasstopic.asp>

Code Samples

Below you will find the complete sample project built in the examples above in both Visual Basic .NET and Visual C#.

Visual Basic .NET

Create a new windows application and place a button on the form. in that buttons click event handler type the following code.

```
Imports CRMSetupFactory

Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    'Create an assembly object
    Dim Assem As Reflection.Assembly
    'Instantiate the object
    Assem = System.Reflection.Assembly.GetExecutingAssembly
    'create / instantiate the parser object
    Dim SetupFactory As New Parser("servername", _
        "databasename", "username", "password",
        Assem)
    'Set the none default properties
    SetupFactory.Auditing = True
    SetupFactory.Debugging = True
    SetupFactory.Backup = True
    SetupFactory.DisplayWarning = True
    SetupFactory.WarningMsg = "Please Click Yes To Backup" _
        & "Your Database"
    'Load the scripts and prepare to install
    SetupFactory.load()
    'Run the process
    SetupFactory.runNow()
    'Get the audit
    Dim AuditLog As String = SetupFactory.AuditLog
    'Create / instantiate a streawriter
    Dim SW As New System.IO.StreamWriter("C:\Audit.txt"
        , True)
    'Write out the audit to a text file
    SW.Write(SetupFactory.AuditLog)
    SW.Close()

End Sub
```

After you have typed the preceding code all that's left to do is embed the script files. If you are unsure on how to embed resources see the section entitled "How Do I Embed Resources" in the F.A.Q section.

Visual C#

Create a new windows application and place a button on the form. in that buttons click event handler type the following code.

```
using CRMSetupFactory;

private void button1_Click(object sender, System.EventArgs e)
{
    //Create the assembly object
    System.Reflection.Assembly Assem;
    //Set the object to a reference of the executing assembly
    Assem = System.Reflection.Assembly.GetExecutingAssembly();
    //Create a Parser object
    Parser SetupFactory;
    //Instantiate the parser object
    SetupFactory = new Parser("servername", "databasename",
        "username", "password", Assem);
    //We need to access these properties through the class
    //because these are static properties.
    Parser.Auditing = true;
    Parser.Debugging = true;
    //Set none default properties
    SetupFactory.Backup = True;
    SetupFactory.DisplayWarning = True;
    SetupFactory.WarningMsg = "Please Click Yes To Backup Your
        Database";

    //Load the scripts and prepare for the installation
    SetupFactory.load();
    //Run the process
    SetupFactory.runNow();
    //Get the audit log
    string AuditLog = SetupFactory.AuditLog;
    //Instantiate a StreamWriter
    System.IO.StreamWriter SW =
        new System.IO.StreamWriter("C:\\\\Audit.txt", True);
    //Write out the audit to a text file
    SW.Write(SetupFactory.AuditLog);
    //Close the writer
    SW.Close();
}
```

After you have typed the preceding code all that's left to do is embed the script files. If you are unsure on how to embed resources see the section entitled "How Do I Embed Resources" in the F.A.Q section.