ProFile Software Development Kit

The ProFile Software Development Kit (SDK) helps you use technology built into ProFile to access data in ProFile files. Any user with Windows® programming skills (in languages such as Delphi®, Visual Basic® or C++) can write programs, macros or plug-ins that interact with ProFile.

Generally, software developers write programs with proprietary data structures. This means you can't get to your data unless you use their interface or their rules. We decided to do something different with ProFile. Our Open Standard of Data Integration (OSDI) is that difference. OSDI makes it easy to get to your data and expands what you can do with ProFile.

Use reliable ProFile calculations from inside other software or from a web page. Take data from custom applications and turn it into ProFile files. Read on to learn how!

Contents

Use the ProFile SDK to take your business and processes to the next level.

The uses of the ProFile SDK are an extension of the vision you have for your firm.

- Integrate ProFile into other applications you use.
- Plug other software into ProFile.
- Use ProFile as the calculation engine on your website.

The possibilities are endless, so start brainstorming.

- Extract the time spent (or the billing amount) from each ProFile T1/TP1 return to your time and billing package, similar to the built-in ProFile-QuickBooks integration.
- Transfer data from your corporate working paper software to the GIFI forms in ProFile T2, avoiding the default export-import steps required.
- Post the corporate tax liability calculated in ProFile T2 back to your working papers.
- Pull current portfolio data from your back-end system for mutual fund transactions and post it to the Net Worth statement in ProFile FP.
- Post planning results to secure web pages for client review.
- Gather simple personal tax planning information and provide a secure online tax plan for the year.

Microsoft® Excel spreadsheets provide examples.

The three examples in the ProFile SDK:

- Import ProFile data into Excel.
- Export data from Excel to ProFile.
- Use ProFile to perform complex calculations in Excel.

You are welcome to use these samples in your own business, modify them for custom purposes, or simply refer to them as you create your own tools.

Use ProFile on the web to attract new clients or offer new services to existing clients.

Try our web example for calculating tax and marginal rates at:

www.greenpoint.ca/active/osdiwebsample.dll

To use ProFile as a Financial Application Server, you must obtain a license from Intuit Canada. Your web application must not compete with existing ProFile applications unless you have made special arrangements with Intuit Canada.

Licensing - OSDI is built into every ProFile application we ship so that users can write applications to access ProFile. However, to connect to ProFile through OSDI either on an enterprise level or a web server for a web site, additional or modified licensing conditions may apply. Please contact us for more details.

Technical Support - ProFile software support does not include training or support in Excel spreadsheets, website design, programming languages or the ProFile data structure. To implement your ideas, we recommend you seek the advice of a programmer who understands Microsoft® COM objects and can work from this document.

Intuit CANADA
400-138 4 Ave SE Calgary AB T2G 4Z6
Telephone 1.800.452.9970
Fax 1.800.792.4044
www.accountant.intuit.ca

ProFile SDK examples

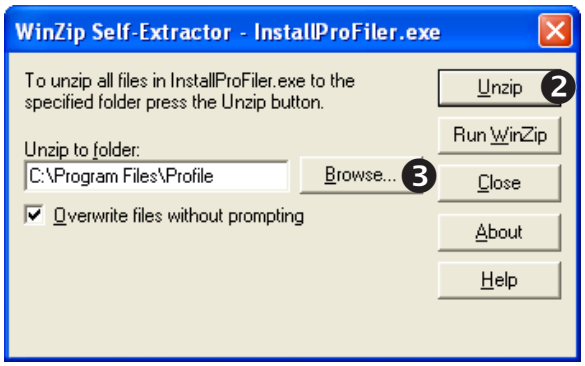
The ProFile SDK includes Microsoft® Excel tools and examples to show you how you could use OSDI technology. You must have Microsoft® Excel installed on your computer to run them. You are welcome to use these samples in your own business, modify them for custom purposes, or simply refer to them as you create your own tools. The spreadsheets contain Visual Basic® macros that perform a few simple tasks using OSDI technology.

1 - Import ProFile data into Excel

For this example, you must first install two files by running InstallProfiler.exe:

1 Double-click Profiler.exe, in the ProFile SDK folder, to launch the self-extracting archive.

2 In the self-extractor dialog box, click [Unzip] to save the files to the default (C:/Program Files/ProFile) ProFile folder.



3 If you run ProFile from a different folder, click [Browse] to change the path. Once the correct path shows in the Unzip to Folder field, click [Unzip].

This executable installs importdata.dat and profiler.xls into your ProFile folder. The importdata.dat file must remain in your ProFile folder. You can run profiler.xls from anywhere on your hard drive; it contains a sample macro for importing data from your ProFile T1 files.

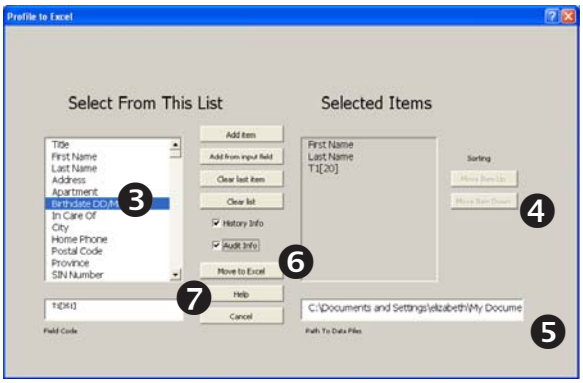
Files in the Toolkit

- **Profiler.exe** contains two files, *importdata.dat* and *profiler.xls* (Example 1 on page 2).
- **2005 Excel examples.xls** contains two spreadsheets, *ProFile T1 MTR calculations* (Example 2 on page 3) and *Employees to ProFile FX T4 slip* (Example 3 on page 3).
- **2005 Slips from Excel to ProFile.xls** contains spreadsheets for importing slips data from other software into ProFile via an Excel spreadsheet (Example 4 on page 4).
- **2005 Preparer Average.xls** contains a spreadsheet which produces a report on the types of returns your firm is preparing and the fees each preparer is charging for their returns (Example 5 on page 5).

To run the example in profiler.xls:

1 Double-click on profiler.xls to open the spreadsheet in Microsoft® Excel.

2 Click the [Run Macro] button to open the ProFile to Excel dialog box.



3 From the list on the left, select fields to import. Simply click a field name and then click the [Add item] button. The name of the field appears in the list on the right.

To import amounts from a field that is not included in the list, type a field code, and click the [Add from input field] button.

4 To change the sequence of the fields, use the sorting buttons right of the selected items list. Highlight any field to move it up or down in the list.

5 The macro gathers data from ProFile T1 files in the default folder: My Documents\My ProFile Data\2005T1. If you save your T1 files in a different folder, enter a new path in the field.

6 Select the History Info or Audit Info checkboxes to transfer all audit or file history messages to the spreadsheet.

7 Click the [Move to Excel] button to transfer the information to the Excel spreadsheet.

8 A message tells you how many files were found and asks you to wait. Once the import is complete, the dialog box disappears and the spreadsheet contains data from those ProFile files.

9 Finally, save the spreadsheet under a new file name. Otherwise, the next time you run the macro, you will overwrite the data you imported previously with the latest data in your ProFile files.

ProFile SDK examples (cont.)

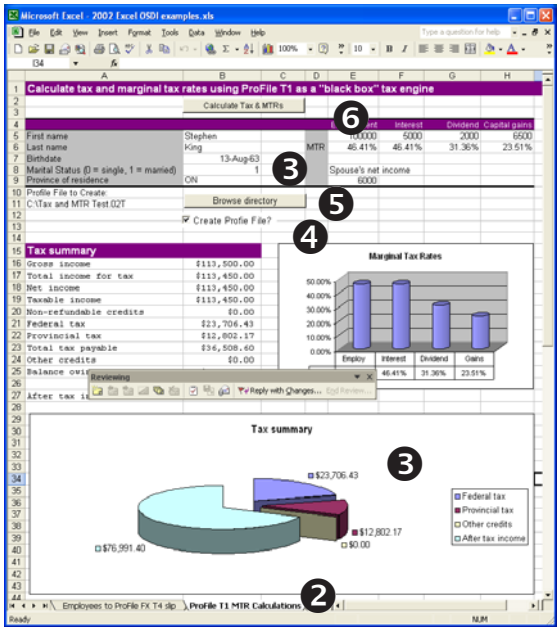
Example 2 - Use ProFile as a tax engine in Excel

This example uses ProFile T1 as the tax engine for calculating the tax and the marginal tax rate for an individual based on the information you provide. When you enter amounts on the Excel spreadsheet and run the macro, the macro calls and uses the ProFile calculations to generate the results.

To try out this example:

1 Open 2005 Excel examples.xls.

2 Click the ProFile T1 MTR calculations tab at the bottom of the screen.



3 At the top of the screen, you can enter client data on which to base the calculation. By default, we have provided personal, spousal and income information for a sample client.

To see how ProFile integrates with this example, change some of this information. For example, change the Employment income to $50,000. You will see an automatic change in the pie chart, but the marginal tax rate and tax amounts do not change until you run the macro.

4 If you would like to create a ProFile T1 file based on this information at the same time as you calculate the tax and the marginal tax rate, select Create ProFile File?.

5 To save the new file somewhere other than in the default location or with a different file name (the default is C:\Tax and MTR Test.05T), click [Browse directory] to select a location and enter a name for the file.

6 Click [Calculate tax and MTRs] to see the results of your changes. You will see changes in the marginal tax rates, the amounts in the tax summary and in the pie chart in the tax summary.

7 To view the ProFile T1 file that was created, go to the directory where you saved the file (by default C:\) and double-click on the file (Tax and MTR Test.05T) to open it in ProFile T1.

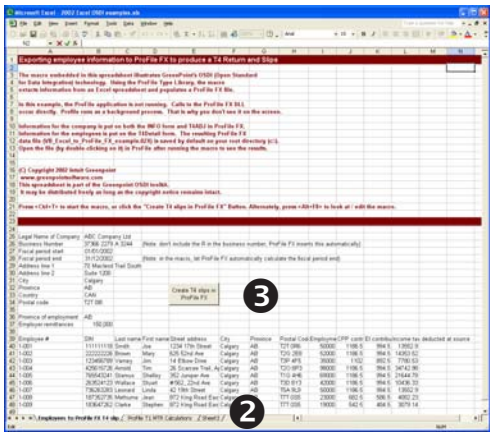
Example 3 - Export data from Excel to ProFile

This example shows you how to export T4 information from an Excel spreadsheet to a ProFile file. The macro extracts the employee and company information from the spreadsheet and saves it in a ProFile FX file.

To create a ProFile FX file using this example:

1 Open 2005 Excel examples.xls in Excel.

2 Click the Employees to ProFile FX T4 slip tab at the bottom of the screen. Notice the list of employees a the bottom of the spreadsheet.



3 Click [Create T4 slips in ProFile FX] to save the data in a new ProFile FX file. You will not see ProFile run, since the macro calls and runs ProFile FX in the background to create and populate the file.

4 To view the new FX file, go to C:\ and double click on VB_Excel_to_ProFile_FX_example.05X.

5 Use File > Save as to rename this file using the company name. If you do not rename the file, the next time you run this example, you will overwrite the original data.

ProFile SDK examples (cont.)

Example 4 - Import slips data from Excel

Use this Excel macro to import slips information (previously exported from other software) from Excel into ProFile FX/Q. Import information for the following slips: T4, T5, NR4, T101, T2202A, T4A, T4A-RCA, T4ANR T4F, T4PS, T5013, T5018 and RL4.

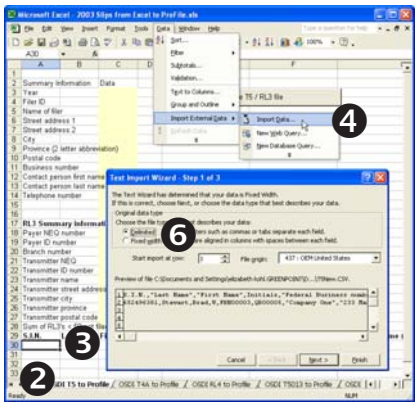
Step 1 - Export information from Informatrix to Excel

This sample file is particularly useful if use Informatrix and can export. If you are not using Informatrix, use these as a rough guide to understanding the process needed for exporting data into the proper format.

1. Launch Informatrix.
2. Select File > Open and choose a client file which includes slip data that is ready for export.
3. Select File > Export and select type of slip to export.
4. On the Export Specifications dialog box, select the individual slips you wish to export by moving them to the right side of the dialog box.
5. Select Add all the fields and do not select Export 1st line as header.
6. Click [OK].
7. On the Export Save dialog box, enter a name and specify a location to save the export file. Remember where you saved the file.
8. Click [Save]. The exported data is saved as a .CSV file (Comma Separated Values) file.

Step 2 - Import the CSV data into Excel

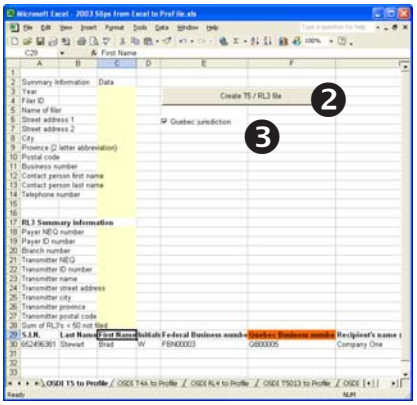
1. Open 2005 Slips from Excel to ProFile.xls.
2. Select the spreadsheet tab that corresponds with the type of slips information you exported into the .CSV file.
3. At the bottom of the spreadsheet, there is a series of column headings for your data. Click once in the first blank cell under the first column heading on the left.
4. Select Data > Import External Data > Import Data.
5. Browse to where you saved the .CSV file in Step 1, select the file and click [Open].
6. OnText Import Wizard dialog box:
   a. Select Delimited as the file type and click [Next].
   b. Select Comma and clear Tab as the Delimiters for your data. Click [Next].
   c. Select Text as the Column data format and click [Finish].



7. Click [OK] on the Import Data dialog box. Slip information appears under the column headings at the bottom of the spreadsheet.

Step 3 - Import slip information into ProFile FX/Q

1. Close ProFile.
2. Review the information your imported into the Excel spreadsheet in Step 2. Make sure the information imported correctly.
3. If the slip information includes amounts for Quebec, select Quebec jurisdiction.



4. Click on the [Create] button on the spreadsheet. For example, on the OSDI T5 to ProFile tab, click [Create T5/RL3 file].
5. Browse to the folder where you will save the file, enter a filename and click [Save].
6. Using Windows Explorer, browse to the folder where you saved the file and double-click on the file to open it in ProFile.
7. In ProFile, review each slip. Make sure the information imported correctly.

Step 4 - Consolidate information for different types of slips into a single file. *Optional*

The process described in Step 3 creates a new ProFile file for each type of slip. If you are importing several types of slips for a single company, you will likely want to consolidate the different types of slips into a single ProFile file.

For example, you are producing T4, T5 and T4A slips for a single company and you created three ProFile files using the Excel spreadsheet: T4.05X, T5.05X and T4A.05X. To combine these three ProFile files into a single file, complete the following steps:

1. Choose a one fo the files to be the file into which you will place the other slips. In this example, we will use the file that contains the T4 slip information, T4.05X.
2. Open the ProFile file from which you will extract the slip information, T5.05X. Go to T5Detail and select Edit > Copy Form.
3. Open the ProFile file into which you will insert the slip information, T4.05X. Go to T5Detail and select Edit > Paste Form.
4. Repeat steps 2 and 3 to copy and paste form information for the slip summary form, T5Sum.
5. Select File > Save.
6. Repeat steps 2 through 5 for each type of slip you need consolidate into a single file.

ProFile SDK examples (cont.)

Example 5 - Calculate preparer averages

This example gives you a way to quickly assess the sucess of your firm and its preparers at any time throughout the tax season. It looks at T1 returns to see how many were prepared by each preparer, whether they prepared business, professional, rental or farming statements and calculates the average fee for the return.

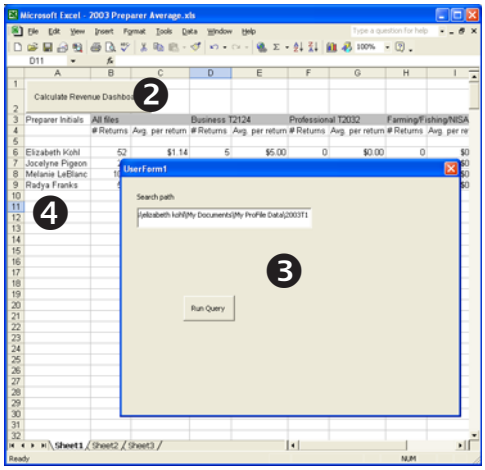
Use this spreadsheet to see which preparers have developed a niche of clients who seek them out for certain types of returns. Review whether your preparers are over- or undercharging certain types of returns. This can help you standardize your pricing and make sure you are not missing revenue opportunities.

To produce this report in Excel:

1 Open 2005 Preparer Average.xls.

2 Click the [Calculate Preparer Averages Button].

3 Enter the path to a folder where you keep your 2005T1 files and click [Run Query].

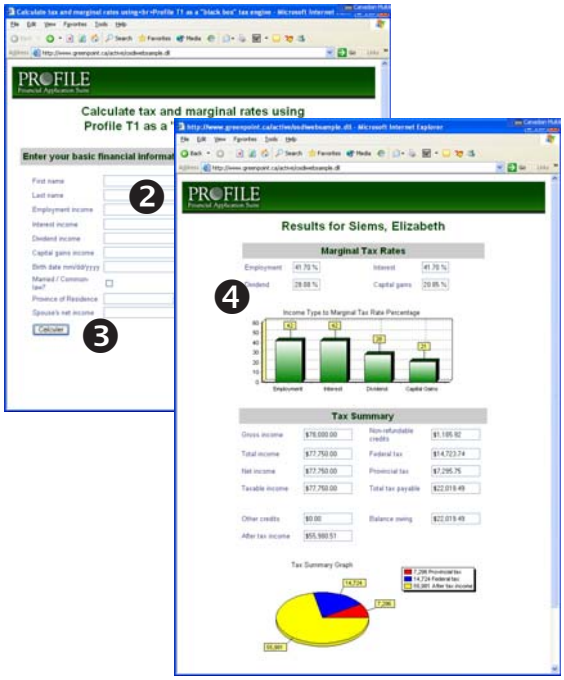


4 Excel will take a few minutes to look at all your files, extract the data and calculate the averages. Then, you will see the summary in the table on the screen.

Example 6 - Use ProFile as web application

You can create web forms to submit data to applications that access ProFile through the type library. ProFile becomes the engine for the web application.

For example, you could create a T1 Tax Planner web application with a web form that gathers simple tax planning information from your clients. When a client submits information, the browser sends data to a custom application that connects to ProFile. That application then communicates with ProFile to create a new T1 tax return and enter the numbers from the web form into the PLAN form in the tax return. ProFile calculates the return and the application retrieves the results for posting to the web page.



We created an example on the web to show how you could use ProFile to calculate tax and marginal rates:

1 Go to www.greenpoint.ca/active/osdiwebsample.dll.

2 Enter basic financial information required to calculate your marginal tax rate (MTR).

3 Click the [Submit] button.

4 The data is sent to a web application running on a server which interfaces with the ProFile T1/TP1 software. It calculates the results and displays a report on a new page.

ProFile SDK implementation

Now that you have explored the examples included in the ProFile SDK, you are probably wondering how to go about implementing your own visions for the technology. Please be aware that you will need more than basic computer skills to undertake these projects. If you have little or no programming experience or spreadsheet knowledge, we recommend that you hire a programmer / consultant to implement your ideas. A few hours of an expert's time may be all it takes to modify one of the examples to meet your needs. If you have the programming skills, or you are a programmer who has been contracted to implement your ideas, the following information will help you get started.

ProFile Type Library

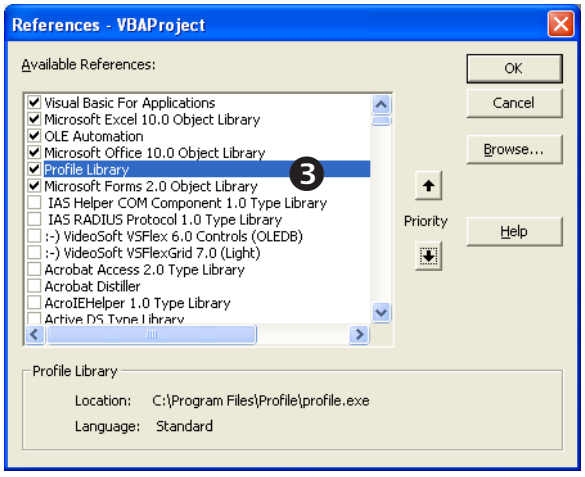
A type library is a file that contains information about objects and types that a component supports. The specification for coding resides in the type library associated with the ProFile executable, profile.exe.

A type library contains type information for classes and interfaces (including the required methods and arguments) that an application supports externally. By importing a type library into your software development environment, you can generate a native code version of the classes and interfaces to work within your chosen programming language.

To incorporate the ProFile data dictionary information into your code, you must have access to the ProFile type library.

To link the ProFile type library into the Microsoft® Visual Basic® Editor:

1. Open the Microsoft® Visual Basic® Editor by opening a macro through Tools > Macro > Visual Basic Editor.
2. Once the Microsoft® Visual Basic® Editor is open, go to Tool > References.
3. In the References dialog box make sure there is a checkmark in the box next to ProFile Library and click [OK].



4. If the ProFile Library does not appear in the list of available references, click [Browse] and go to your ProFile folder (by default C:\Program Files\ProFile).
5. Select profile.exe in that folder and click [OK].
6. Now select Profile Library and click [OK].

ProFile Data Dictionary

We expose the ProFile data structures to give you access to ALL fields in ANY ProFile data file, as well as File > Properties and option settings.

ProFile applications have hundreds of forms and data structures. The process of manipulating all this data by doing calculations, flowing numbers between forms, expanding tables, carrying forward data, attaching memos and calculator tapes, flagging diagnostics, maintaining override field status, etc. is very complex. To help in this process, ProFile includes the Data Dictionary to keep track of fields.

The Data Dictionary is an index of all the data variables, the parameters of these variables, cross references, where the data is used on a form, order of operation flags and more. It is the central repository for the unique field codes. All data flow and calculations move through the Data Dictionary. When writing data to and reading data from the ProFile data structures, you are using the ProFile Data Dictionary.

Field Codes

If you look at the Excel macro source code you will see ProFile field codes that link the macro to the fields within ProFile. These are the same field codes that you can display on the ProFile Data Monitor, and that you use to customize the ProFile templates or to query the database. Each field code is a reference to the data structure that uniquely identifies a field on a form. These codes are stored within the ProFile Data Dictionary.

A ProFile data file includes many different groups, most of which correspond to a form. Each group has three types of data arrays: currency amounts (reals), text (strings) and bytes (or booleans).

To access a single data element, use the group name, followed by a period, a single character representing the type of data and an index into the data array. For example:

- to access the last name of the client on the T1 Info form, use the field code: T1Info.S[4]
- to access the marital status of the client on the T1 Info form, use the field code: T1Info.B[10].

When using ProFile T1, you can access the equivalent data field in a coupled spouse return by typing "Spouse" before any field code. For example, to access line 101 of the spouse's 2005 T1 jacket, use the syntax: Spouse.T1[20].

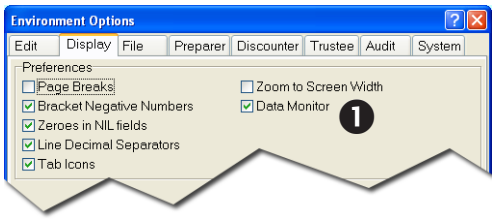
For examples of how to access data elements in expandable ProFile tables or on forms that may have multiple copies in a single data file, refer to the Excel examples included in the ProFile SDK.

ProFile SDK implementation (cont.)

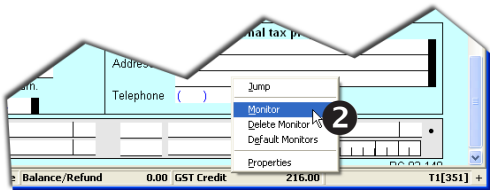
Display Field Codes

Use the Data Monitor to view the field code:

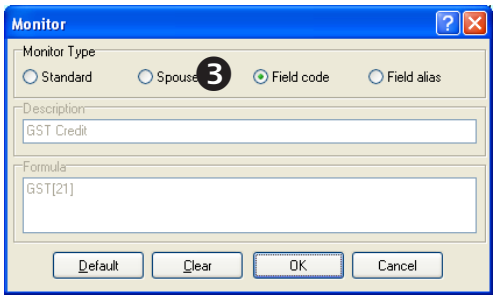
1 Select Options > Environment > Display > Data Monitor. When you click [OK], the Data Monitor will appear across the bottom of the screen.



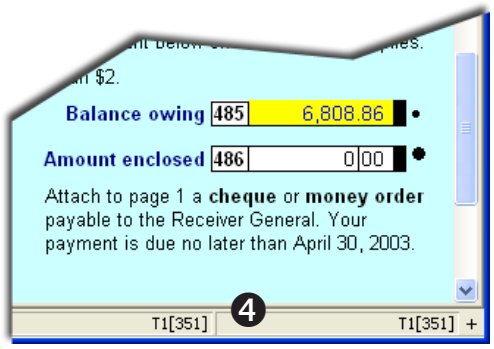
2 Right-click on a cell in the Data Monitor and select Monitor.



3 In the Monitor dialog box, select Field code from the radio buttons at the top and click [OK].



4 Move your cursor to a field on a form. The Data Monitor cell that you modified will show the field code.



Print Field Codes

To print all the field codes on a form, right click-on the form. Hold down <Ctrl + Shift> and select Print from the context-sensitive menu. ProFile will print the field codes instead of the values in the fields.

Copy Field Codes

To copy a field code:

1 Select Options > Environment > Edit > Right Click Cut Copy Paste.

2 Open a form.

3 Right click in a field and select Copy field code from the menu.

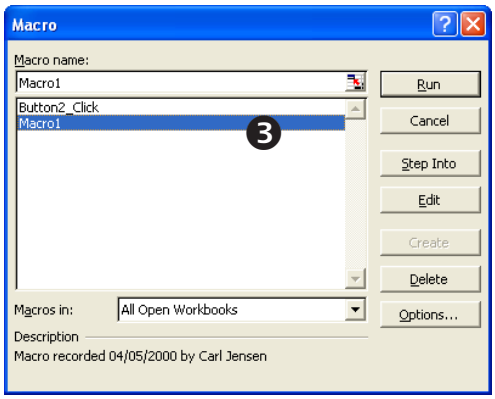
Formatting Dates

Dates in ProFile are stored as whole numbers. This means that dates must be converted when they are written to ProFile or read from ProFile. There is an example of a date conversion in profiler.xls:

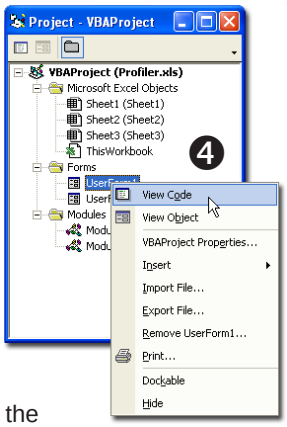
1 Open profiler.xls.

2 Go to Tools > Macro > Macros.

3 In the macro dialog box, select Macro1 and click [Edit]. This will launch the Microsoft® Visual Basic® Editor.



4 Once the Microsoft® Visual Basic® Editor opens, go to the panel on the left, right-click on UserForm1 and select View code. The code for the form will appear on the right.



5 Go to Edit > Find and search for Delphidate.

That search will take you to the section of source code that converts the date from the number stored in ProFile.

ProFile plug-ins

The plug-in mechanism in ProFile allows software developers to hook into actions and events that occur while you use ProFile. For instance, a plug-in could:

- replace or enhance the action taken when you select an item from a menu such as File > Save;
- prompt the user for security or login information when starting ProFile in order to prevent unauthorized access; or
- prepare to update or query a database as the user interacts with ProFile.

What is a Plug-in?

A plug-in is an auxiliary program that works with a software package to modify or enhance its functionality. In order to "plug into" an application, a plug-in must adhere to strict specifications defined by that application. This allows the application to talk to the plug-in without being aware of the implementation details of the plug-in.

Plug-ins are very widely used in software today. Many popular software packages allow plug-in functionality developed by third-party software designers. For example:

- WinAmp, a popular music/media player allows others to create skins and visualizations to plug into its interface. Skins change the appearance of the application interface. Visualizations provide animation as the music plays.
- Adobe Photoshop, an advance image editing program, allows other software creators to build tools to plug into its existing functionality.
- Microsoft® Internet Explorer, the most popular web browser, permits many types of plug-ins to display special content on HTML pages. For multimedia, you can install a QuickTime plug-in to show movies or a RealAudio plug-in to listen to live music.
- Microsoft® Word, the industry standard word processor, accepts numerous third-party plug-ins to assist with writing. The Adobe Acrobat PDFMaker allows you to create a PDF from a Word document.

Basic Steps to Creating a ProFile Plug-in

To create a ProFile plug-in:

1. Create a COM object that implements the interface IProfilePlugIn, as found in the ProFile type library.
2. Fill in desired customizations of ProFile by writing code within the methods defined by IProfilePlugIn.
3. Compile the plug-in object (usually into a DLL).
4. Register the COM plug-in object in the registry so that ProFile can find it. This is usually done by calling regsvr32 with the name of the DLL that contains the plug-in object.
5. Edit the registry to include a data value called PlugIn in the key:

   HKEY_LOCAL_MACHINE\Software\GreenPoint\Profile

   The data value "PlugIn" is a string value and contains the GUID (Global Unique Identifier) of the plug-in object (as registered with the regsvr32 step). This data value has a format similar to this:

   {661A4EF4-4869-11D2-A337-006008A9956D}
6. Run ProFile. If everything is correct, the plug-in will react to the user's actions within ProFile.

Microsoft® COM objects

The data interface component of OSDI is a Microsoft® COM object. ProFile plug-ins are implemented as COM automation objects (Microsoft® 's Component Object Model standard) embedded in a DLL.

COM stands for Component Object Model and is a Microsoft® binary standard that allows a component to describe any of the constants that it supports in a type library. COM has been part of the Windows® family of operating systems for many years as the underlying framework that makes OLE (Object Linking and Embedding), and more recently ActiveX, possible. COM is an object-based framework for developing and deploying software components.

- Defines a binary standard for component interoperability;
- Is programming-language-independent;
- Provides for robust evolution of component-based applications and systems;
- Is extensible by developers in a consistent manner;
- Uses a single programming model for components to communicate within the same process, and also across process and network boundaries.

You can code COM objects in any programming language that is able to create COM objects. If you want more information on COM, surf to this URL: www.microsoft.com/com/.

ProFile plug-ins (cont.)

IProfilePlugIn

The ProFile plug-in mechanism depends on an interface called IProfilePlugIn. To create a plug-in compatible with ProFile you must create a COM automation object that supports the IProfilePlugIn interface and its associated methods.

Although this can be done in any language that supports COM, and the methods will be slightly different in each language, the fundamentals remain the same. Our examples will use Delphi - the language we use for coding ProFile. Once IProfilePlugIn is in Object Pascal (the native language of Delphi) you can create a COM automation object that supports the IProfilePlugIn interface.

IProfilePlugin in Object Pascal

When you create an object that supports an interface, you are creating a programmatic promise that the object has certain methods with certain arguments, called in a certain way. Once a plug-in is availabile to ProFile, the software can create the plug-in object and start calling the IProfilePlugIn methods used by the object.

For instance, the plug-in code for the AppOpen method is called when you launch ProFile. You could use this code to display a welcome dialog or a login page. Using the ShowQuickStart method argument (in the AppOpen method), you can also control whether or not the QuickStart dialog box displays and whether or not ProFile should continue to open (method argument = Continue) if a particular condition is met or not met (for instance, the login failed).

Sample 1 - IProfilePlugIn, imported into Object Pascal

```
IProfilePlugIn = interface(IDispatch)
    ['{CE694B21-9547-11D3-8606-BB9193E3F22C}']
    procedure  Initialize; safecall;
    procedure  FieldChangeNotification(const ProFileClient: IProfileClient;
               UniqueID: Integer); safecall;
    procedure  ClientNotification(const ProFileClient: IProfileClient;
                                  const FileName: WideString;
                                  const ClientID: WideString; Action: Integer;
                                  var Response: Integer); safecall;
    function   GetAboutBoxBitmap: Integer; safecall;
    function   GetSplashBitmap: Integer; safecall;
    function   GetProfileDialog: IProfileDialog; safecall;
    procedure  ExecuteAction(const AActionID: WideString;
                             var Handled: WordBool); safecall;
    procedure  AppCloseQuery(var CanClose: WordBool); safecall;
    procedure  AppClose; safecall;
    procedure  AppIdle; safecall;
    procedure  AppOpen(var ShowQuickStart: WordBool;
                       var Continue: WordBool); safecall;
    function   RTUNotification(const ProFileClient: IProfileClient;
                               RTUType: Integer;
                               out AError: WideString): WordBool; safecall;
    procedure  GetLicenseName(var LicenseName: WideString); safecall;
    procedure  GetLicenseCode(var LicenseCode: WideString); safecall;
    procedure  GetEncryptionType(var EncryptionType: SYSINT); safecall;
end;
```

ProFile plug-ins (cont.)

Plug-in Code

The Delphi code resembles the IProfilePlugIn interface definition; however, the type is called TProfilePlugIn. TProfilePlugIn is a class based on TAutoObject (a COM automation object) that supports IProfilePlugIn. By including the IProfilePlugIn interface in the class definition, we are requiring the class to define methods that match the methods of the IProfilePlugIn interface.

Sample 2 - Plug-in class written in Delphi

```
type
    TProfilePlugIn = class(TAutoObject, IProfilePlugIn)
    Public
        procedure  Initialize; safecall;
        procedure  FieldChangeNotification(const ProFileClient: IProfileClient;
                   UniqueID: Integer); safecall;
        procedure  ClientNotification(const ProFileClient: IProfileClient;
                                const FileName: WideString;
                                const ClientID: WideString; Action: Integer;
                                var Response: Integer); safecall;
        function   GetAboutBoxBitmap: Integer; safecall;
        function   GetSplashBitmap: Integer; safecall;
        function   GetProfileDialog: IProfileDialog; safecall;
        procedure  ExecuteAction(const AActionID: WideString;
                               var Handled: WordBool); safecall;
        procedure  AppCloseQuery(var CanClose: WordBool); safecall;
        procedure  AppClose; safecall;
        procedure  AppIdle; safecall;
        procedure  AppOpen(var ShowQuickStart: WordBool;
                         Continue: WordBool); safecall;
        function   RTUNotification(const ProFileClient: IProfileClient;
                                 RTUType: Integer;
                                 out AError: WideString): WordBool; safecall;
        procedure  GetLicenseName(var LicenseName: WideString); safecall;
        procedure  GetLicenseCode(var LicenseCode: WideString); safecall;
        procedure  GetEncryptionType(var EncryptionType: SYSINT); safecall;
end;
```

Steps for Coding a ProFile Plug-In in Borland Delphi 7

Within Borland Delphi 7:

1 Select File > New > Other.
2 Select ActiveX Library from ActiveX page.
3 With the newly created DLL active in the Project Manager select File > New > Other.
4 Select Automation Object from ActiveX page.
5 Enter a Class Name. In this example, name it TestPlugIn.

Within the unit that contains the automation object:

1 Select Project > Import Type Library.
2 Select profile.exe to generate the Profile_TLB.pas file.
3 Add PROFILE_TLB.pas to the interface uses class. This makes the IProfilePlugIn definition accessible to the new unit.
4 Remove the default interface from the TestPlugIn automation object, the class definition for TTestPlugIn and the type library.
5 While in the type library editor, add IUnknown to the Implements list for the TestPlugIn CoClass.
6 Add the IProfilePlugIn interface and methods to the class definition (Copy/Paste from Profile_TLB).
7 Select the class definition and press <Ctrl+Shift+C> to create the method stub.
8 Fill in the code for the different stubs.
9 Compile and correct errors.
10 Register the server via Run > Register ActiveX Server or by using "regsvr32 TestPlugIn.dll" from the Start Menu > Run box.
11 Edit the registry to include a data value called PlugIn in the key HKEY_LOCAL_MACHINE\Software\GreenPoint\Profile
The data value PlugIn is a string value and contains the GUID of the plug-in object (as registered with the regsvr32 step). This data value has a format like this: {661A4EF4-4869-11D2-A337-006008A9956D}
12 Run ProFile to use the Plug-In.

ProFile plug-ins (cont.)

Methods defined by IProfilePlugIn

Initialize

Called	Immediately after the plug-in loads.
Parameters	None
Result	None
Notes	ProFile services are not fully available through this method. Make any calls requiring IProFileServices in AppOpen

AppOpen

Called	After ProFile starts and immediately before the QuickStart dialog appears.		
Parameters	*ShowQuickStart*		WordBool Set to TRUE to show the QuickStart dialog box.
	Continue	WordBool	Set to FALSE to shut ProFile down immediately.
Result	None		

AppIdle

Called	When the application is idle (when it is not processing).
Parameters	None
Result	None

AppCloseQuery

Called	Just before the main ProFile window closes.		
Parameters	*CanClose*	WordBool	Default = TRUE Set to FALSE to prevent ProFile from shutting down
Result	None		

AppClose

Called	Just as the main ProFile window closes.
Parameters	None
Result	None

GetLicenseName

Called	When the ProFile licensing information is needed (Help > License).		
Parameters	*LicenseName*	WideString	Set the licensee name.
Result	None		

GetLicenseCode

Called	When the ProFile licensing information is needed (Help > License).		
Parameters	*LicenseCode*	WideString	Set the license code.
Result	None		

GetProfileDialog

Called	Not currently used
Parameters	None
Result	IProfileDialog

GetAboutBoxBitmap

Called	When you select Help > About, just before the About dialog box displays.	
Parameters	None	
Result	*Integer*	Return 0 to use the default Help > About bitmap image. Return the handle to the Windows® GDI Bitmap (HBITMAP) to use as the splash screen.

ProFile plug-ins (cont.)

GetSplashBitmap

Called	On launch, just before the splash screen appears. (The splash screen is the image that appears on the screen as ProFile loads.)		
Parameters	None		
Result	*Integer*		Return 0 to use the default splash screen bitmap image. Return the handle to the Windows® GDI Bitmap (HBITMAP) to use as the splash screen.

GetEncryptionType

Called	When ProFile saves a data file		

Parameters	*EncryptionType* SYSINT		Have the plug-in return one of the following constants to set the encryption type to use when saving a file.
			ofeNoCode = $00000000;
			ofeACode = $00000002;
			This method should be ignored.

Result	None		

ExecuteAction

Called	Upon selection of most menu items.		
Parameters	*AActionID*	WideString	The internal name of a menu item or the supplied name of a menu item added by the plug-in. The ActionIDs that you wish to act upon can be determined by implementing this method, clicking menu items, and logging the AActionID that comes through.
	Handled	WideString	Return TRUE when you handle an action so that ProFile does not execute the default action related to this AActionID.
Result	None		

ClientNotification

Called	When an action affects a client data file (that is File > Closed, File > Created, File > Saved, etc.)		
Parameters	*ProFileClient*	IProfileClient	Interface to the client data object upon which the action occurred.
	FileName	WideString	Name of the file that contains the client data upon which the action occurred. (The data file must have been saved at least once.)
	ClientID	WideString	Social Insurance Number of the client data file upon which the action occurred.
	Action	Integer	The action undertaken. (Constants in the following table (in type library as OClientAction).
	Response	Integer	Not currently used.
Result	None		

FieldChangeNotification

Called	When an onscreen field's value changes.		
Parameters	*ProFileClient*	IProfileClient	Interface to the client data object upon which the action occurred.
	UniqueID	Integer	By using the IProfileModule method RegisterFieldNotification you can let ProFile know that you wish to be notified when certain fields' data changes. In setting up that notification you associate a numeric identifier with the field code or field alias for which you wish to receive notifications. The Parameter UniqueID returns that numeric identifier when the registered field code or alias triggers the FieldChangeNotification call.
Result	None		

RTUNotification

Called	Just before a T1 client is built into an RTU, EFILE On-Line, EFILE On-Line Plus or TP1 NetFile file.		
Parameters	*ProFileClient*	IProfileClient	Interface to the client data object upon which the action occurred.
	RTUType	Integer	
	AError	WideString	Return the text error specified in this parameter.
Result	WordBool		Return whether (TRUE) or not (FALSE) to allow the client to be included in the EFILE file.

ProFile plug-ins (cont.)

Action Constant	Occurs When...
caCreated	A new file is created. A new spouse is added to a T1 file.
caModified	A file is saved
caInPreparerReview	The client status changes to *In Preparer Review* on the File > Properties dialog box or via the Status After Printing command.
caInPartnerReview	The client status changes to *In Partner Review* on the File > Properties dialog box or via the Status After Printing command.
caPrinted	A file is printed.
caCompleted	The client status changes to *Completed* on the File > Properties dialog box or with the Status After Printing command.
caInRTU	A T1 client file is built into an RTU, EFILE On-Line or EFILE On-Line Plus.
caRTUSent	An RTU or EFILE On-Line Plus file is transmitted to the CCRA.
caRTUAccepted	ProFile receives notification that a client's T1 or T2 data was accepted by the CCRA.
caRTUNotAccepted	ProFile receives notification that a client's T1 or T2 data was rejected by the CCRA.
caInSEND	A T1 client file is built into a SEND RTU.
caSENDSent	A SEND RTU is transmitted to the CCRA
caSENDResult	ProFile receives a known SEND result from the CCRA.
caSENDNoResult	ProFile receives an unrecognized or empty SEND result from the CCRA.
caChequeIssued	The Issued Cheque information is entered on the File > Properties dialog box or on the RC71 (*T1 Only*).
caChequeReceived	The Received Cheque information is entered on the File > Properties dialog box (*T1 Only*).
caChequeReconciliation	The Cheque Reconciliation information is entered on the File > Properties dialog box (*T1 Only*).

Action Constant	Occurs When...
caOpened	A file is opened.
caDeleted	A file is deleted
caFileMoved	A file is moved using the Client Explorer or the Classic Database.
caCoupled	A two T1 files are coupled.
caUncoupled	T1 files are uncoupled.
caReadyToPrint	The client status changes to *Ready to Print* on the File > Properties dialog box.
caCarriedForward	A client file is carried forward.
caImportedData	Data was successfully imported into the current client from another file type.
caImportedGIFI	GIFI data is successfully imported in a T2 file.
caTP1EDISent	A TP1 EDI file is transmitted to the MRQ.
caTP1EDIAccepted	ProFile receives notification that a client's TP1 data was accepted by the MRQ.
caTP1EDINotAccepted	ProFile receives notification that a client's TP1 data was rejected by the MRQ.
caTP1EDIInERU	A T1 client file is built into a TP1 EDI file.
caClosed	A client file is closed.
caOnlinePaymentReceived	An online credit card payment is successfully processed.
caQBInvoiceAdded	An invoice is added to QuickBooks