



GREAT PLAINS

White Paper
Great Plains Dynamics and
Microsoft Visual Basic for Applications

Table of Contents

Introduction	2
Dynamics VBA Product Positioning	3
Customization Strategy	4
Leveraging Dynamics	
Customizing Dynamics	
Integrating Dynamics	
Dynamics Integration with VBA	5
Benefits of the Dynamics architecture	
Dynamics VBA Overview	7
VBA customization environment	
Dynamics project	
Dynamics object model	
Programming Dynamics using VBA	10
Programming assistance	
Dynamics methods, properties and events	
Customizing Dynamics	13
Customizing Windows	
Working with reports	
Leveraging Dynamics	15
Working with the Dynamic User Object Store (DUOS)	
Using ActiveX controls	
Integrating Dynamics	17
Using Automation with Dynamics	
Paradigm Shift: Integration Now	18
Conclusion	18

Introduction

Flexibility, fit, customization and integration are all important considerations when selecting a business management solution for your organization. To meet these needs, Great Plains implements technologies that allow for flexibility in customization and integration of existing components. The addition of Microsoft's Visual Basic for Applications (VBA) allows Dynamics Customers and Partners to customize, extend and integrate their business management systems using the most widely used development and customization environment in the software industry.

The same version of VBA built into Dynamics is found throughout Microsoft Office. This consistency makes it easier to leverage current development resources for creating custom solutions. In addition, Microsoft based the *customization* environment of Visual Basic for Applications on the popular *development* environment Visual Basic. Visual Basic currently has over 3 million developers, providing a virtually unlimited source of Dynamics customization talent.

The Dynamics product architecture allows for quick adoption of new technologies that benefit customization in Dynamics. Great Plains can quickly adopt open standards like Microsoft's Component Object Model (COM) standard, allowing for application interoperability using VBA. The ability to move quickly is imperative as support for VBA extensions, features and standards are introduced over time.

Dynamics VBA Product Positioning

An integral part of Microsoft Office, Visual Basic for Applications is poised to provide the software industry's first shared, common environment for developing and distributing application-specific software. Integrated into Dynamics, VBA allows customers and partners to customize Dynamics to fit particular business practices.

Additional benefits of Visual Basic for Applications include:

- ◆ **VBA is truly shared among applications** – Use commercially available ActiveX controls, and integrate important business information with Microsoft Excel, Word or Access using Automation. Perhaps the most compelling aspect of VBA is the fact that it's truly a shared development environment among applications. The programming experience will be the same regardless of whether the user is customizing Microsoft Excel or Word, Visio, AutoCAD or Great Plains Dynamics. With VBA built into Microsoft Office, millions of PC users already have the same programming and customization environment that's supported in Dynamics.
- ◆ **An abundance of VBA resources** – Since Visual Basic for Applications is a version of Visual Basic, Dynamics immediately gains 3 million Visual Basic developers as development resources. The best news is that they won't need to spend valuable time learning a new development system. With skilled resources already available, VBA will make it easier and faster for developers to build custom solutions for Dynamics, making Dynamics implementations go faster and smoother. In addition, there are literally hundreds of books, trade publications, seminars and trade shows available to VBA developers.
- ◆ **Great Plains leverages Microsoft's COM standard** – Visual Basic for Applications is a component-driven language engine that requires applications to support the Component Object Model (COM) standard. Developed by Microsoft, the COM standard rapidly is becoming the foundation for all application customization and automation. Great Plains widely supports the COM standard, and builds COM objects for Dynamics that are exposed to VBA, and programmable using the VBA customization environment. In addition, all Microsoft Office products support the COM standard, allowing for tight integrations with Dynamics.
- ◆ **Inherit technology on a daily basis** – Because of the wide-support for VBA and COM-based components, there are literally hundreds of new ActiveX components and COM-enabled applications becoming available on a daily basis. As new products are available, you can extend Dynamics with custom controls, or tightly integrate Dynamics with new products and tools that provide support for COM.
- ◆ **Great Plains can focus on delivering business solutions** – Since Microsoft maintains and updates the Microsoft customization environment, Great Plains can spend more time delivering financial management solutions.

Customization Strategy

Great Plains' customization strategy provides maximum flexibility while maintaining total integrity of business rules and data. Great Plains delivers two specific tools that allow for customizations:

- ◆ ***The Dynamics Modifier*** provides the ability to quickly customize Dynamics windows. This includes rearranging window layout and customizing the look and feel of individual data entry windows, as well as making global changes to prompt and message text.
- ◆ ***Visual Basic for Applications*** allows you to imbed your company's business rules within Dynamics application logic, integrate custom ActiveX controls with Dynamics forms, or tightly integrate Dynamics with popular productivity and business packages such as Microsoft Excel, Word or Access. Distributed with the Modifier as a complete customization package, Visual Basic for Applications provides unprecedented levels of customization.

Key to Great Plains' customization strategy is to deliver the ability to leverage, customize and integrate Dynamics to provide maximum benefit to your organization.

Leveraging Dynamics

VBA allows users to display additional information or manipulate existing information in Dynamics, to create additional data entry forms, and to store, retrieve and print user-defined data on any Dynamics window or report.

For more advanced development needs, VBA provides support for ActiveX Data Objects (ADO). ADO provides a uniform interface to data, and when combined with any of commercially-available ODBC drivers for Dynamics, ADO allows direct read/write access to any or Dynamics SQL or Btrieve tables.

Finally, VBA has built-in support for any ActiveX control available today. Thousands of these controls are available as "plug and play" components in a VBA environment, and the number of controls are growing rapidly as the popularity of Visual Basic and VBA grow.

Customizing Dynamics

Both VBA and the Modifier provide extensive capabilities for altering the display of information on a Dynamics window. The ability to change the overall look and feel of Dynamics is supported both in the Modifier and VBA. In addition, VBA allows control over customizations such as field-level security, enforcement of specific business rules and practices, as well as the ability to default field values.

Integrating Dynamics

With VBA, Dynamics users have control over other applications through the use of Automation (formerly OLE Automation). Applications such as Microsoft Excel, Word, Access and Outlook support Automation, and Dynamics users can create direct links from Dynamics windows, reports and fields to any of these applications. For example, you can use Automation to automatically open Excel, export all current customer balances from Dynamics to an Excel spreadsheet, then automatically chart those balances.

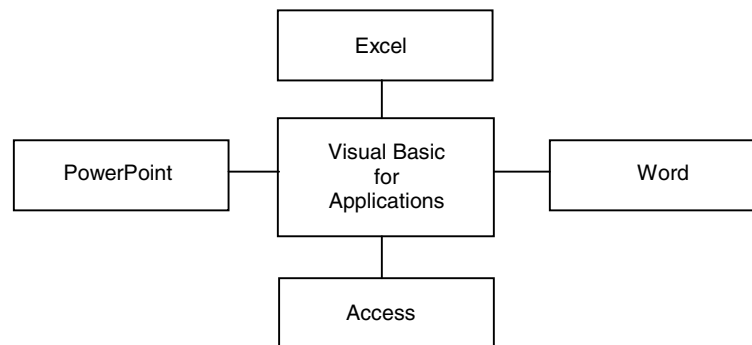
Dynamics Integration with VBA

Microsoft's Visual Basic for Applications is a standard programming language used to customize, extend and integrate Microsoft Office products. Based upon the popular development environment, Visual Basic, VBA provides many of the same development tools and code syntax as Visual Basic, but built into an application.

In addition to VBA, Great Plains reviewed several customization solutions available that use Visual Basic-like syntax. In reviewing these solutions, Great Plains determined that each could be used to achieve similar customization goals, but only VBA provides the same full-featured integrated development and debugging environment as Visual Basic. The customization tools Great Plains reviewed include:

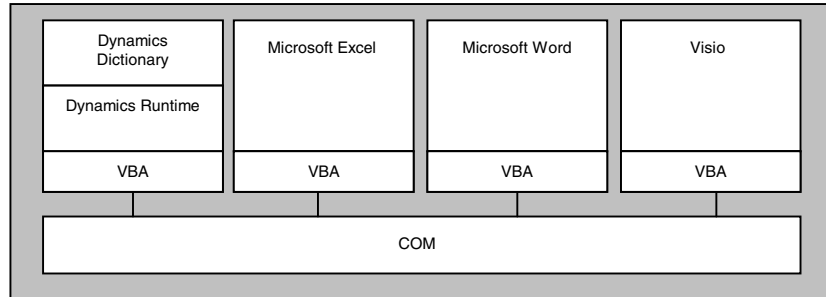
- ◆ **Microsoft's VB Script** – VB Script provides support for VB Syntax. Although a comprehensive forms package is available, the development environment is not as fully developed as VBA, lacking support for debugging, project management, drop-down IntelliSense help, and real-time syntax inspection.
- ◆ **Mystic River's SBL** – SBL implements a VB-like syntax for application customization. In reviewing this product, Great Plains determined that the development environment was limited, and SBL will not support VBA functionality and syntax compatibility.

VBA in Microsoft Office provides more depth and flexibility in building custom solutions by allowing control over application objects such as spreadsheets (Excel), text documents (Word) and databases (Access), and allows for tight integrations between Word, Excel, Access and Outlook. In addition, the unified Microsoft customization environment allows users to learn a single set of tools for all Office applications:



In June of 1996, Microsoft began licensing Visual Basic for Applications 5.0 to ISVs. This meant developers who updated their application to adhere to the Component Object Model (COM) standard could now imbed, or *host*, VBA and benefit from the same level of customization and integration that previously was available only in Office products.

Once software developers licensed VBA from Microsoft, application components written to the COM standard could then be controlled through VBA. Perhaps the greatest benefit of developing applications to the COM standard is that applications like Dynamics and Visio can use COM as a platform to integrate and extend their software with Microsoft Office applications, as well as other applications that host VBA:

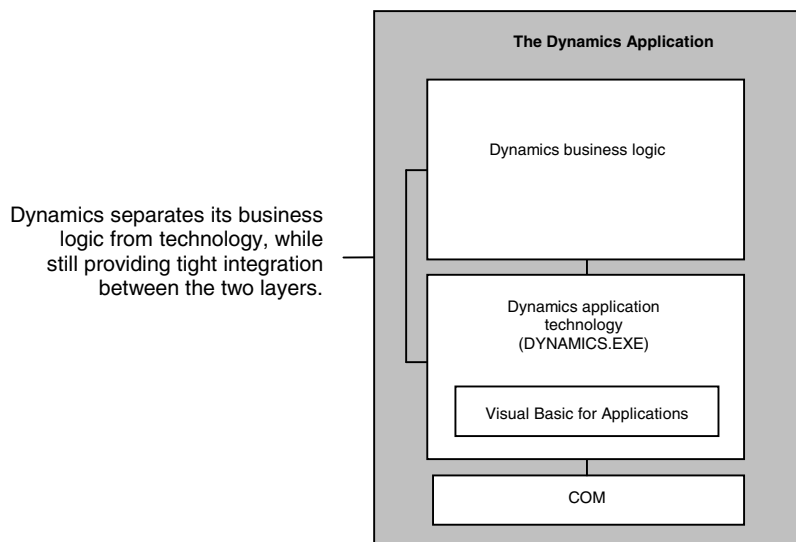


Benefits of the Dynamics architecture

Based upon its flexibility and wide acceptance, VBA was chosen by Great Plains as its customization tool for Dynamics. However, to host VBA, the VBA environment not only needed to become “part of” Dynamics, but support for the COM standard needed to be built.

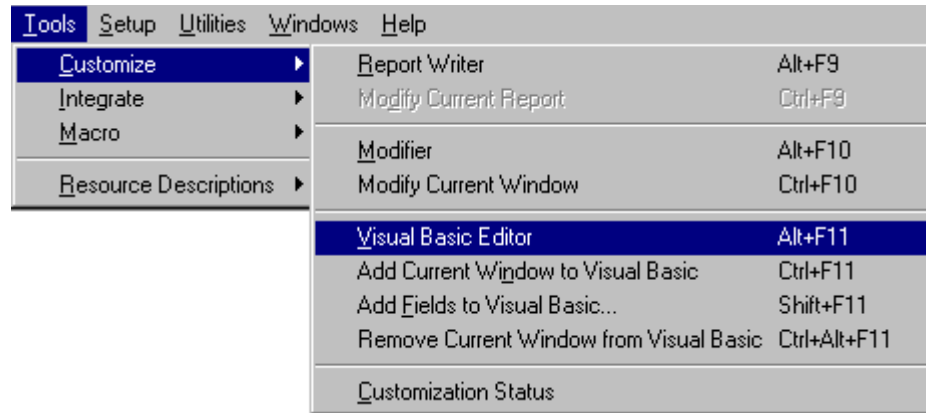
For many software packages, the integration of VBA would be exceedingly difficult. The technology supported for the application and the application’s business logic are assembled, compiled and integrated as one component. When new technology such as VBA is added to the application, it then has the potential to impact not only the technical portion of the application, but also the business logic. Therefore, the introduction of new technology is a lengthy and potentially risky proposition.

However, Dynamics separates its application technology (including support for OLE, COM, ODBC and Pervasive.SQL and SQL database APIs) from its business logic (including Dynamics windows, posting and transaction processing). This division is significant because implementing new technology impacts only one portion of the product, and allows for rapid adoption of new tools and standards such as VBA and COM:



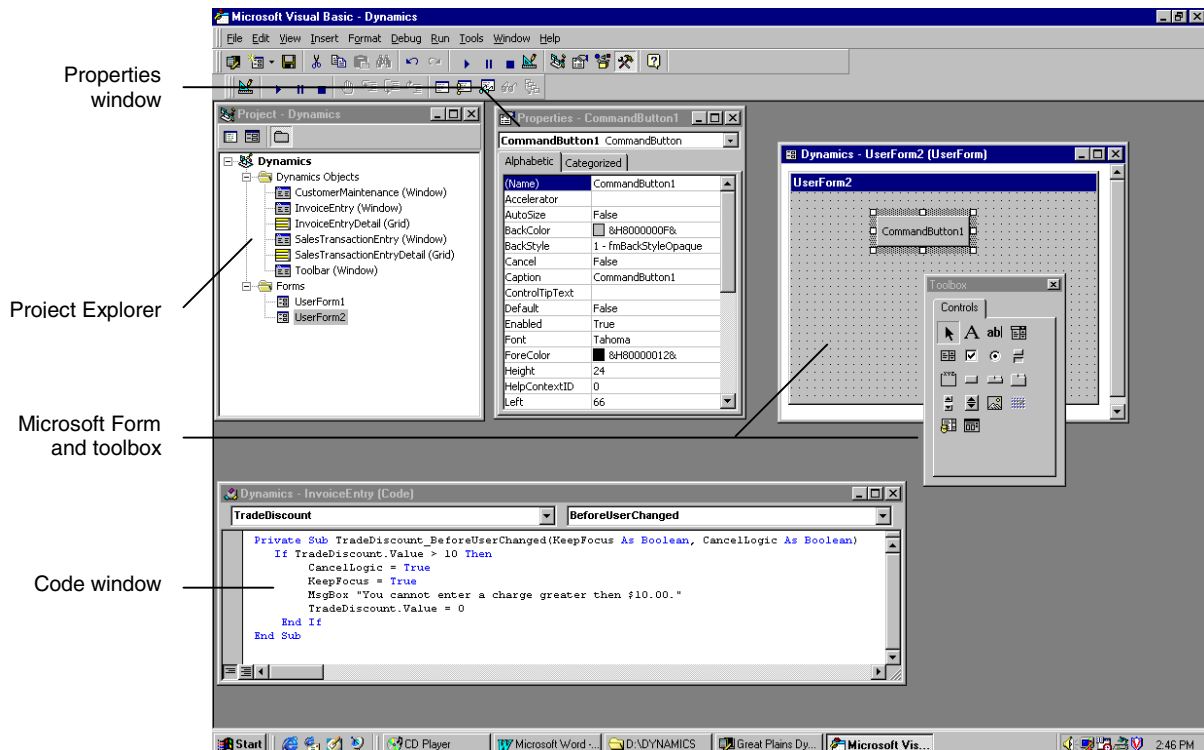
Dynamics VBA Overview

Visual Basic for Applications includes the Visual Basic language engine (programming language and compiler), as well as a complete integrated development environment (IDE) that includes an editor, Project Window, Properties Window, and debugging tools. Dynamics includes Visual Basic for Applications with the Modifier. When users register the Modifier, the Visual Basic Editor becomes accessible from the Tools menu:



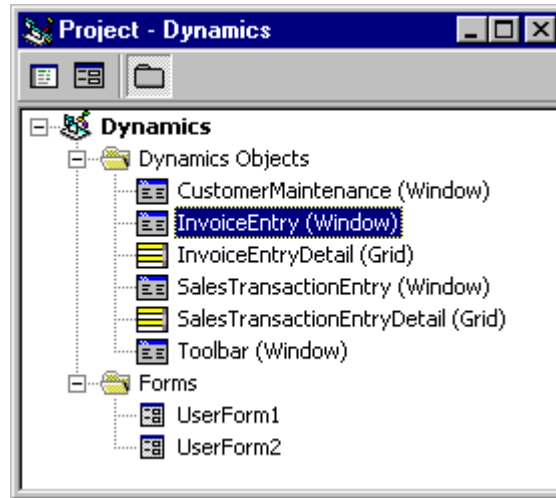
The VBA customization environment

The VBA customization environment uses a number of on-screen tools that help users build and manage Dynamics customizations. The tools available in this environment are identical between host applications, so the Microsoft customization environment in Microsoft Excel or Word will be exactly the same as in Dynamics:



The Dynamics project

The Visual Basic Editor stores Dynamics customizations in a project file. Once added to your project, you can reference and manipulate these objects in VBA using predefined methods and properties. In addition, the Visual Basic Editor lets you add Microsoft Forms, user-defined procedures and user-defined class modules to your Dynamics project:

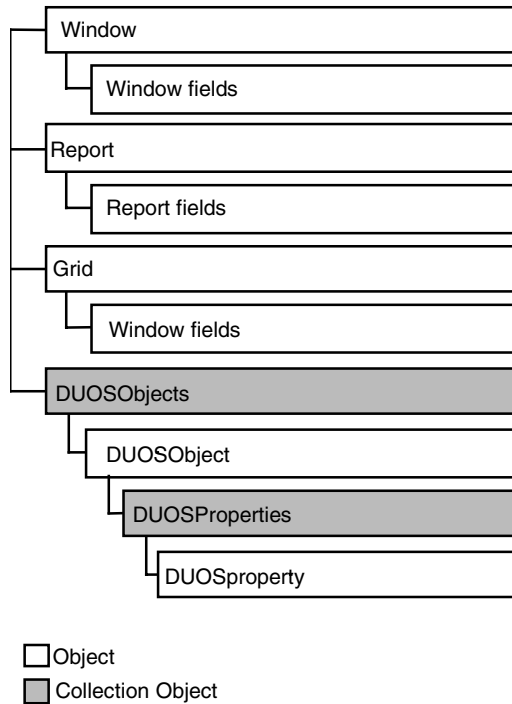


Dynamics allows you to selectively add the window and report objects you want to use with VBA. Dynamics supports “object addition” mode during runtime, so, when a window is displayed, you can use Dynamics menu commands to add the window, then point-and-click the window’s fields, buttons and controls you want to reference in VBA.

The ability to selectively choose only the objects you integrate with will significantly aid your ability to upgrade customizations from release to release, a key requirement for anyone customizing their business management system.

The Dynamics object model

VBA allows developers to work with application components (such as documents, charts, drawings, etc.) through the application's *object model*. The application objects in this model must be written to Microsoft's COM standard, and allow developers control over those objects in a VBA environment. Dynamics exposes its application components to VBA through a set of programmable objects. These objects include windows, reports, grids and Dynamic User Object Store (DUOS) objects:



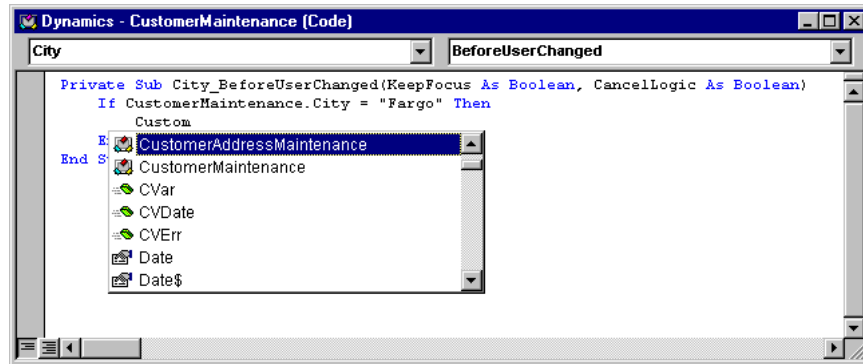
Like object models available for other applications that host VBA, the organization of the Dynamics object model is important, since you must navigate through the object model within VBA code to access lower-level objects. For example, to access a window field, you must first indicate the field's window object:

```
'Set a field on a window  
CustomerMaintenance.CustomerID.Value = "AARONFIT0001"
```

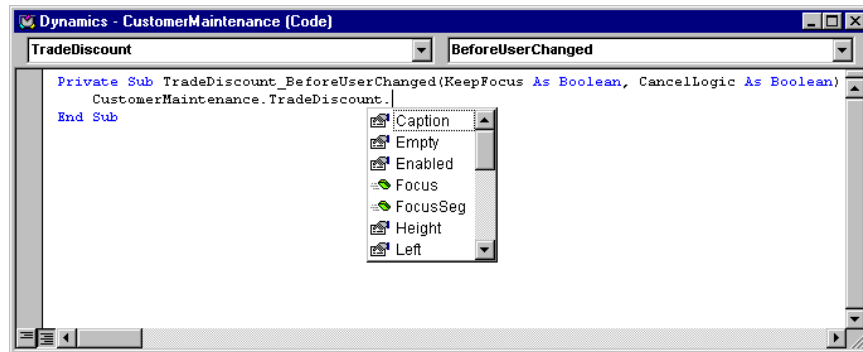
Programming Dynamics using VBA

Programming assistance

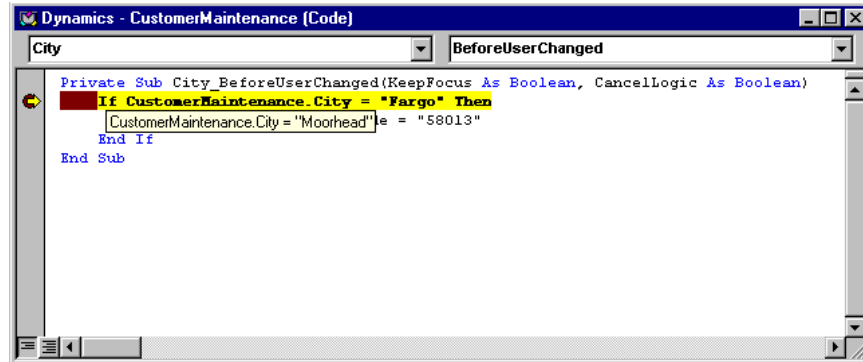
There are several tools built into VBA that aid in programming Dynamics objects. With VBA's IntelliSense feature, VBA prompts the user through the process of writing code. The following example shows the sequence for writing a line of code to set the value of the zip code field based on the value entered in the City field. The user can simply start typing the window name (CustomerMaintenance), and VBA prompts the user to auto-complete the name of the window:



VBA then prompts the user with a list of the fields available for the CustomerMaintenance window, as well as a list of the methods and properties for the CustomerMaintenance window:



Additional tools like the built-in debugger allow users to evaluate when and how VBA code is executing. By adding a breakpoint to VBA code, the user can simply position the mouse over the expression and display the current contents of an expression in a Tooltips box.



Dynamics methods, properties and events

Each object in the Dynamics object model has defined *methods* and *properties* you can use in VBA code to manipulate the behavior of the object. Methods and properties for Dynamics objects are declared using standard VBA *object.method* and *object.property* syntax. Most objects also have specific *events* that specify when associated VBA code executes for the object.

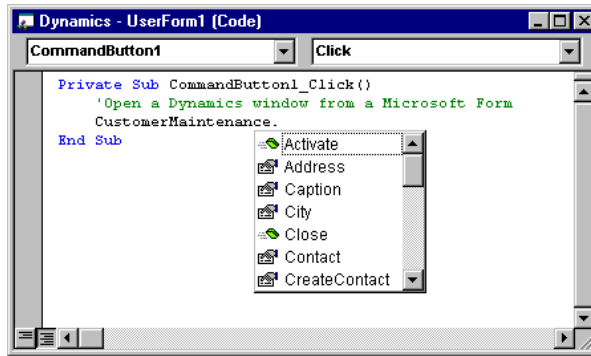
Methods are the actions you can perform for a given object, such as when you *open* a window or *hide* a field. For example, the Dynamics field object uses the **Focus** method to move the focus to a different field in the window:

```
`Move the focus to the salesperson field  
SalespersonID.Focus
```

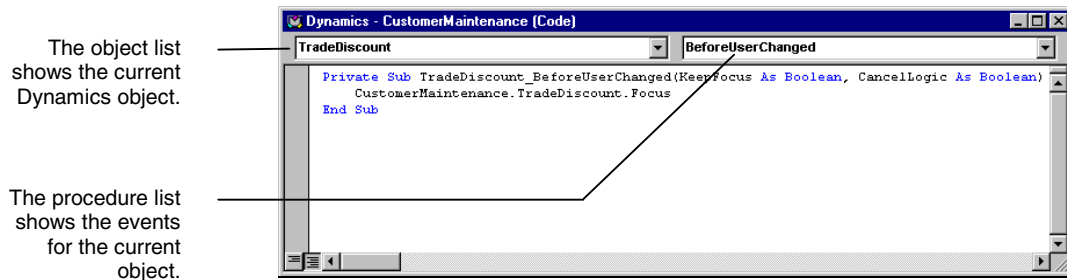
Properties are characteristics of a given object, such as the *name* of a window or the *value* of a field. You can retrieve or set an object's property. For example, you can use the Dynamics **Caption** property to change the name of a Dynamics window or field:

```
`Change the prompt for a field  
CustomerMaintenance.CustomerID.Caption = "Patient ID"  
  
`Change the title of a window  
CustomerMaintenance.Caption = "Patient Maintenance"
```

Once you've added an object to a project and use it in VBA code, IntelliSense provides immediate access to a list of all methods and properties for a given object. Notice that this drop down help capability provided by Microsoft's IntelliSense technology is fully aware of the Dynamics object model. The following illustration shows the methods and properties for the CustomerMaintenance window object. Any of the CustomerMaintenance fields added to the project appear in the list, as well:



Events specify when VBA code executes. A code procedure associated with an event executes at certain times for a specific object, such as when the user changes the value of a field, clicks a button, prints a report, or opens a window. Event procedures are predefined for each object and can be viewed in the VBA Code window:



For example, **BeforeOpen** is an event for a window object. This event executes its associated VBA code when the specified window opens. In the following example, the **BeforeOpen** event for the Receivables Transaction Entry window sets field values before the window is visible:

```
Private Sub Window_BeforeOpen()
    SortBy.Value = 3 'Set the sort list to "Date"
    DocumentType = 7 'Set the document type to "Returns"
End Sub
```

Customizing Dynamics

Customizing windows

You can execute VBA code whenever a window opens, closes, or activates. Additionally, you can specify whether the code should execute before or after any Dynamics code that executes for the same event. In addition to windows, Dynamics also exposes the fields contained on the selected windows. For window fields, you can execute VBA code when the user enters the field, changes the value of the field, or exits the field. Some of the most common window and window field customizations include:

- ◆ **Provide field defaulting** – You can default field values when a window opens, or default a field's value when the value of another field changes. Strict enforcement of business integrity is maintained at all times by Dynamics, and you cannot set fields to values that don't meet Dynamics field-level verifications of data. Field defaulting can also be performed on a user basis, allowing you to customize Dynamics windows based on how each user works with that window.
- ◆ **Provide field-level security** – You can programmatically lock access to given fields, disable buttons or hide information in the window based on the current user and company. For instance, using a VBA user form, you can require the user to enter a password when they enter a transaction amount greater than a predefined limit. Unless the user enters the correct password, or enters a smaller transaction amount, you can disable the Save and Post buttons in the window.
- ◆ **Enforce business rules** – You can programmatically enforce business rules using VBA. For example, after a user enters the discount percent, you can check the value and stop processing if the discount percent exceeds a predefined limit. In addition, built-in VBA support for message boxes gives you control over communicating with the user.
- ◆ **Show additional information** – Using the Dynamics Modifier, you can add more data entry fields to a Dynamics window. Using the Dynamic User Object Store (DUOS) or another data store mechanism (such as a Microsoft Access database), you can store or retrieve data for these fields and display it in the window.

Working with reports

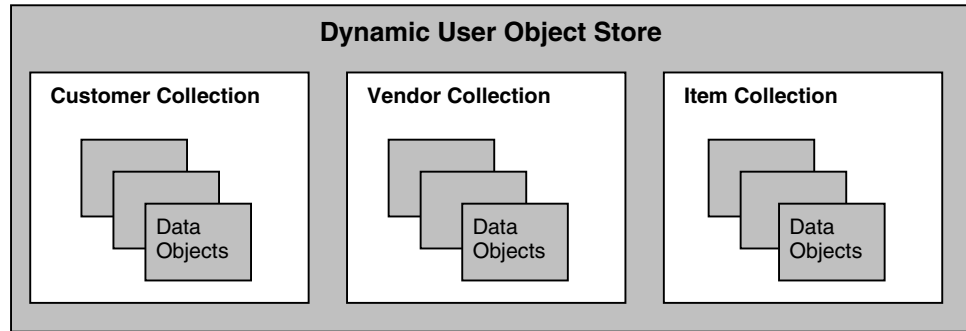
You can execute VBA code when a report starts, when a selected band prints (such as the report's header, body or footer) or when the report ends. Some of the ways in which you can use VBA with Dynamics include:

- ◆ **Export record sets using Automation** – Since reports can contain several records from across several tables in Dynamics, you can use VBA to select fields from a report, such as a customer name and transaction total field, and export that data into Excel or any external application using *Automation* (formerly referred to as OLE Automation). As each customer name and transaction total prints, Automation allows you to add that information directly to cells in the Excel spreadsheet or any external application. Data can also be pumped out of Dynamics into an external application, such as an Oracle database using an ODBC connection in VBA and the query processing power of the reporting engine.
- ◆ **Show additional information** – If you store additional data using the Dynamic User Object Store (DUOS), Microsoft Access or another COM-enabled database, you can create new fields on a Dynamics report, and set those fields with values stored in the database.

Leveraging Dynamics

Working with the Dynamic User Object Store (DUOS)

The Dynamic User Object Store (DUOS) lets you use VBA to create, store and retrieve user-definable data using a predefined set of Dynamics collections and objects. You can use the DUOS to extend Dynamics by storing data entered in new fields you add to a window (using the Modifier), or fields entered in a Microsoft Form. You can then retrieve a data object from the DUOS and display it in a Microsoft Form or in a Dynamics window.



The DUOS is like a data “bag”: whatever you place in the DUOS, you can pull back out when you need to. Using VBA, you can create as many collections as you want, such as customer, item, or vendor collections, and store additional information (data objects) in each collection. For instance, you may want to track a customer’s internet address, then retrieve that information and display it on the Customer Maintenance window when the user displays that customer record. DUOS collections and data objects are created and maintained specific to the current Dynamics company, so you can create custom collections and objects for each company you’re using.

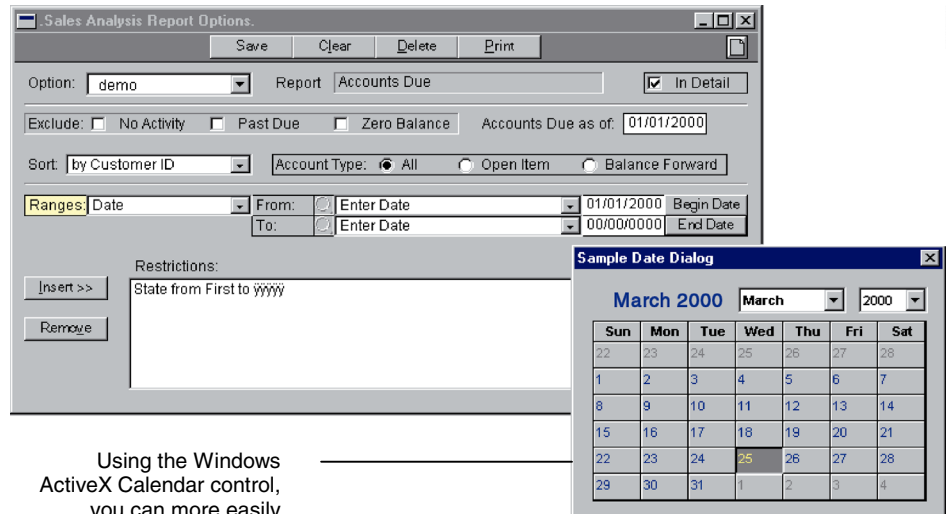
In addition to the DUOS, other options for accessing your Dynamics data are available, including:

- ◆ **ActiveX Data Objects (ADO)** – When combined with commercially available Btrieve or SQL Server ODBC drivers, ADO provides direct access to Dynamics tables.
- ◆ **Microsoft Access** – Since Access is an Automation server, you can extend Dynamics windows or Microsoft user forms by storing and retrieving data directly from an Access database.

Using ActiveX controls

Since VBA supports integration with ActiveX controls, you can tightly integrate and extend Dynamics using either commercially available ActiveX controls, or custom ActiveX controls built using Visual Basic. ActiveX controls are compact, reusable software components, and represent the most widely used, most versatile component format available for the Windows platform.

An example of an ActiveX control is the Windows Calendar control. By adding this ActiveX control to a VBA project, you can quickly and easily enhance a Dynamics window. In the following example, buttons added to the window using the Modifier invoke VBA code that displays the Calendar control. When the user selects a date, the Calendar control returns the date to a Dynamics date field:



Using the Windows ActiveX Calendar control, you can more easily provide date ranges for a report.

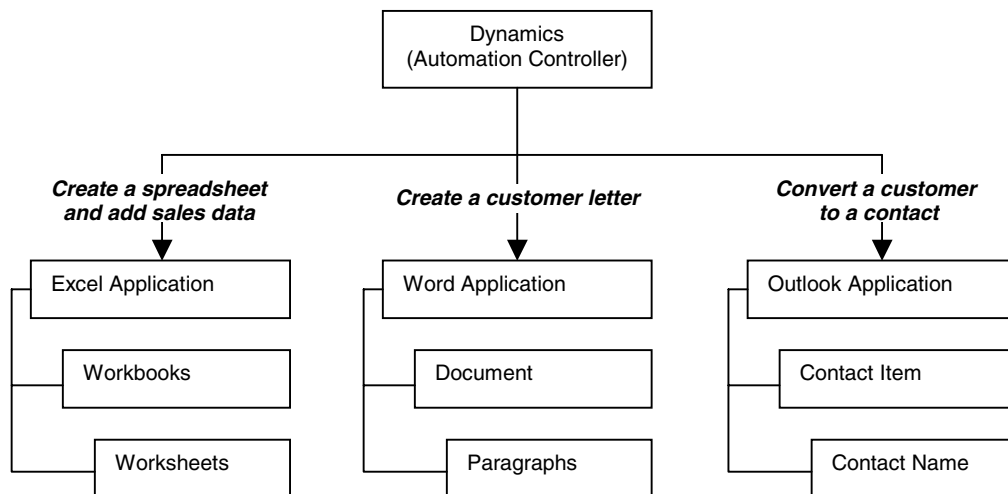
Because there are more than a thousand reusable ActiveX controls already available, you have a wide choice over the ActiveX controls that fit your customization needs. Several hundred software makers currently create ActiveX controls, including companies such as Borland, Oracle, and Sybase/Powersoft.

Integrating Dynamics

Using Automation with Dynamics

Dynamics can integrate and share data with any application that supports *Automation*, enabling you to integrate objects in the Dynamics object model (windows, fields, reports) with objects in another application's object model. Automation is extremely powerful for integrating Dynamics with other desktop and productivity applications in a way that appears seamless to the user.

There are always at least two applications involved in Automation; an *Automation controller* controls the actions between the two applications, and the *Automation server* responds to the Automation controller. Dynamics is an Automation controller and can create and manipulate objects in Automation servers like Microsoft Outlook, Excel and Word:



For example, Microsoft Outlook for Microsoft Office supports Automation. When you declare references to the Microsoft Outlook object model in your Dynamics VBA project, you can create instances, or *instantiate*, Outlook objects such as a contact item or a task item. By using the events available to the Dynamics window and window field objects, you could add a button to the Customer Maintenance window that, when clicked, would add the current customer to Outlook. In addition, you could place VBA code on a Dynamics customer report that exports your top 10 customers to Excel and automatically generates a graph, or use the same report to generate a mail merge document using Word.

Since Dynamics is an Automation controller, your Dynamics VBA code controls everything that the other application does. An Automation operation can be completely invisible to the user if you want. For example, Excel has a set of powerful built-in mathematical functions. You could use these functions to perform math calculations using Dynamics data, without the end-user ever seeing Excel.

Paradigm Shift: Integration Now

In a recent copy of *Datamation*, David E.Y. Sarna and George J. Febish noted the following in an article titled “Paradigm Shift: Integration now”:

“Finally, these systems interact with Word to document the changes and with PowerPoint to build a presentation for management. The mundane common tasks are automated via VBA and the COM interface of each product. The programmers concentrate on what they’re presenting, not on where things are. This brings groupware to a new height.

What is really amazing about the previous paragraph is that it all exists today—off the shelf. Microsoft showed us a demo of just such a system using the *Great Plains Dynamics* accounting system, Visio’s *Visio Professional* business graphics tool, and *Microsoft Office*. We’re not usually awed by demonstrations, but this one hit us hard. A paradigm shift had occurred, and we had missed it.

What’s amazing is that one of our industry’s biggest problems was solved using the most popular development language in history and no one noticed. Quiet revolutions are usually the ones you look back on and ask, ‘How did it happen so fast without me knowing about it?’”

Conclusion

Microsoft Visual Basic for Applications allows users to leverage, customize and integrate business applications to deliver maximum benefit to the organization. Great Plains Dynamics is the ONLY business management solution to deliver the power of Microsoft VBA to its customers. Released in August 1997, the Dynamics Modifier with VBA has over 4,000 installations, and all Great Plains’ top development and value added reseller partners are skilled in using VBA to fill customer requests, fitting Dynamics to their business processes and adding capabilities as they become important.

Find out how Dynamics and VBA can help you reach your business management goals. Contact your local Great Plains Partner, or order your Dynamics demonstration CD from Great Plains by calling 800-456-0025, press 2.

Copyright © Great Plains Software, Inc. All rights reserved. Great Plains eEnterprise is a registered trademark of Great Plains Software, Inc. All other product and company names may be trademarks of their respective owners. No part of this publications may be duplicated by any means without permission from Great Plains Software, Inc. Printed in the USA.



US and Canada 800-456-0025
International 701-281-0555
E-mail: info@greatplains.com
www.greatplains.com