

CRM AS A RAPID DEVELOPMENT PLATFORM
MICROSOFT DYNAMICS CRM 4.0

DAVID YACK

Leveraging Microsoft Dynamics CRM 4.0 as an application development platform to build real world business solutions that go beyond what you think of as “Just CRM”

CRM as a Rapid Development Platform – Microsoft Dynamics CRM 4.0

Published by

We Speak You Learn, LLC
2928 Straus Lane Ste 200
Colorado Springs, CO 80907

www.thecrmbook.com

Copyright 2008 by We Speak You Learn, LLC and David Yack

Limit of Liability/Disclaimer of Warranty: The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or no longer exist between when this work was written and when it is read.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher. Printed in the United States of America.

ISBN-13: 978-0-9815118-1-8

ISBN-10: 0-9815118-1-3

About the Author

David Yack is the CTO of Colorado Technology Consultants, a Microsoft Gold Certified Partner based in Colorado. He is a Microsoft Regional Director and a Microsoft MVP for ASP.NET. As a senior hands' on technology and business consultant with over 20 years of industry experience, David enjoys developing applications on the Microsoft platforms, specializing in large system architecture and design. David embraced .NET during the final beta days of version 1.0 and has been helping clients migrate and build new applications on the technology, as well as helping to mentor and train their staffs.

David has a special interest in **Microsoft Business Solutions** and their integration capabilities. David has been involved in all aspects of multiple **Microsoft Dynamics CRM** deployments using both the onsite and hosted model. David has been instrumental in the development of numerous projects using the Microsoft Dynamics CRM platform as vertical solutions from an ISV perspective. David has been a trainer for CRM 4.0 and has been traveling globally on behalf of Microsoft training their leading ISVs on the capabilities of the new version of the platform.

David is a frequent speaker at user group and industry events and was on the author team of two other .NET related books. He lives in Colorado Springs with his wife and two children. You can always track David down via his blog at <http://blog.davidyack.com> where he writes about his .NET adventures and <http://crm.davidyack.com> which is devoted to Microsoft Dynamics CRM.

Credits

Editor: Julie Yack

Technical Editor: Cathy Hardman

Book Designer: Julie Yack

Introduction

Years ago, the only way you could build an application was to start from scratch each time. The concept of re-use then became popular, where we had some components we could leverage and not have to re-invent the wheel each time. Today we are encouraged to use third-party vendor controls or components to do as much as we can to build solutions more quickly. The current business climate is such that they no longer have patience to wait for long application development times. They want a solution yesterday for today's problem. And did I mention they don't want to pay an arm and a leg?

Microsoft developers are given a wealth of tools to make their job easier. In fact, sometimes the number of them can overwhelm us and cause us to go into zombie mode as we try to figure out which ones to use. Using the .NET Framework and the advanced languages like C# and VB.NET, we can build amazing solutions that bring together other Microsoft components like Windows Workflow and Reporting Services. All this is great but they also all come with a big note that says "***Some Assembly May Be Required***". This means that these can each provide great value but it's left up to the developer to bring them together to build an application on top of or as I will refer to throughout this book to assemble an "Application Development Platform". Think of the Application Development Platform being like the foundation of a house, meaning it's what you build on top of.

An application development platform for our purposes is all the low-level functionality or "the plumbing" put together in one package that we could leverage time and time again. Look across the Microsoft developer division and what you see is a bunch of components, each providing great benefit and most designed to play well with other Microsoft components, but nowhere will you find a true application platform that comes pre-assembled.

Don't get me wrong, I like building infrastructure code; you know the stuff that glues together Reporting Services, Workflow Foundation, Error Handling and all the other products described by two or three letter buzz words. Today, businesses do not have the time or the patience to wait while we figure out how to put the parts together. Building that stuff is fun once every few years but not every time and with the pace of technology change, you need a large team doing that not two or three developers.

As we sit back and think about our perfect application development platform we would include things like workflow, reporting, integration with Outlook, an easily customized data model, and of course offline user support so our users can work anywhere. Oh sure the list could go on and we will explore more about what makes up a platform throughout the book but, the goal here is just to get you interested enough to read more.

In this book, we are going to explore using Microsoft Dynamics CRM 4.0 as an application development platform. I can hear your thinking “Oh great, I bought this book and it’s talking about some contact management application and this guy is going to tell me how to use it as my Swiss Army knife of application development, I could have just bought a book on Microsoft Access”. No worries, I wouldn’t do that to you. I think after you hear some of the compelling features that are hidden behind the name CRM you too will agree that Microsoft has an “Un-Named” application developer platform.

I call it the “Un-Named” application developer platform because I think calling it Microsoft Dynamics CRM 4.0 causes developers to bypass it for non CRM solutions. You know, kind of like when I give my son a lecture on growing up, at some point his eyes just glaze over. That same thing happens to developers when you mention the acronym CRM. That happens for one of two reasons. Either they don’t do CRM or they have no clue what CRM even stands for. I think as you understand a little more of the history of the product you will see how this problem was created. Originally, there was no application developer platform story and it was *just* a CRM tool. As CRM evolved, and the “plumbing” also evolved, a platform that could be used for more than just CRM solutions emerged. I thought about coming up with a crafty name we could call the plumbing or the Application Developer Platform but in the end I decided that was really Microsoft Marketing’s job and I should focus on telling you why there were two parts to the Microsoft Dynamics CRM 4.0 product. The first part is the platform; the second is the CRM functionality that Microsoft built on that platform.

To bring you up to speed a little more on the thinking and to try to help you understand we aren’t crazy, let’s look back in history. CRM 1.0 came out and it was a basic CRM-type application having all the basic features you would expect in a contact management tool to help you with your customers. Then 1.2 followed and added some international support and other minor features. Still in 1.2, it was focused on managing customer relationships and really offered no generic application developer story. CRM 2.0 was in the works but midway into the development Microsoft decided to push harder and skip 2.0 in favor of a more robust CRM 3.0 release.

It wasn't until CRM 3.0 hit the street that we started to get something we could use to build more custom applications and not just a CRM tool. CRM 3.0 allowed the creation of custom entities and relationships which gave the ability to have a declarative data model that an administrator, developer or business analyst could customize to model real business problems. This data model was also exposed as a dynamic SOA web service interface that developers could use to interact with the data model. CRM 3.0 also introduced the integration SQL Server Reporting services as the reporting engine showing the CRM team's desire to start integrating with other Microsoft "best of breed" components and not build their own. This version showed the first signs of an application development platform being born, but it still had several pieces that needed polishing and was lagging a little behind in using the current version of things, including the .NET framework.

Microsoft Dynamics CRM 4.0, which is what we will cover in this book, represents a significant advance in the platform capabilities and introduces a number of new features. These features, while they make it possible to build CRM solutions, are more than enough that developers can use it as an application development platform for line of business applications. CRM 4.0 continues the evolution on a number of the core platform features but also brings the platform current with the latest .NET framework and tools. Other key features like converting from a proprietary workflow engine to Windows Workflow Foundation again showed how the team was willing to pick up and leverage other components. Throughout the book we will continue to paint the picture of how all the core platform capabilities come together to make it easy for you to build CRM solutions as well as any type of general business application on the platform.

When I first thought about writing this book, it was early on during the development cycle when CRM 4.0 was still referred to as code name "Titan". At the time, I thought that workflow was going to be the biggest change in the release and my writing plans were centered on the use of workflow (using Windows Workflow Foundation) in CRM. As the Beta of Titan progressed it became clear that significant changes were being made, not just to workflow. As I started passing around a draft table of contents for review and feedback it became obvious some expansion was needed. Shan McArthur, who I met when he attended one of the trainings I did on the Beta for Microsoft, pointed out that you can't cover work flow without touching on plug-ins which are another way to automate business processes. Plug-ins also are also the mechanism by which workflow is integrated into the platform. Taking that to heart I went back and revisited the table of contents and expanded what I decided I would cover. At the same time, I did a lot of thinking about what was out there for building more generic business applications. In CRM 3.0, those capabilities were there but it was a bit of a stretch to call it a generic application development platform. It was then I realized that a story existed about a

developer platform that just wasn't being told. CRM 4.0 brings enough change though multi-tenant, offline, workflow, etc. the story had to be told and you had to decide for yourself if it was compelling enough to build applications upon. With that, I went back to the drawing board yet again and revised the contents to include enough to make sure developers beyond just the CRM developers would find plenty of information to leverage CRM 4.0 as a development platform.

So, the new and improved scope will cover a broader concept of what developers will find useful in developing custom solutions on Microsoft Dynamics CRM 4.0. We will not be covering the basic concepts of how to use Microsoft Dynamics CRM 4.0 from a user's perspective. Help documents and other books are available to help you understand basic operations and use – this book will focus on the developer topics.

The following are the key areas we will discuss in this book:

Understanding how to use CRM 4.0 to build line of business applications

What's new in CRM 4.0

The developer environment – how to effectively work as a developer

Client side development – a beyond the basics look at doing client side development – including advanced scripting and Silverlight integration

Web Services - web services provide a dynamic SOA interface to the platform – we will look at what's new and how to use them

Metadata API – you can now not only read the metadata but can update it using the API

Plug-ins – They used to be Callouts but now they're new and improved

Workflow – It's re-invented and a great way to automate processes

Keep reading to see a more complete list of the chapters and a synopsis of what you will learn in each.

Differences between this and the SDK

This book on Microsoft CRM 4.0 is intended to supplement the SDK documentation that is provided by Microsoft. Microsoft, in the SDK, will typically provide a detailed reference of all the classes related to the platform. This book does not attempt to provide that level of detail reference and in many cases will defer to the SDK. The SDK will also provide a very broad developer guide and samples for the platform. In this case there may be some overlap in what we cover, but typically the samples will be different and we will go into more details and provide a consultant's perspective. What do we mean by a consultant's perspective? When writing documentation for a product you must be 100% on track with the capabilities and provide guidance that is broadly usable to the majority of users. In general, you never step away from what is 100% supported and concern about real world implementations are not always the first concern. In this book, we will attempt to give the consultant's perspective and ways to look for creative solutions. If we know something is not supported we will try to highlight that fact so you can make the determination if you want to use the approach.

Who is this book for?

The target audience for this book is solution architects and developers that are building applications inside their companies. It is also for those packaging up to resell as part of an ISV solution deployed on the Dynamics CRM platform. Finally, it is for the consultant who is the road warrior helping companies customize the platform for their business needs. One thing to be clear up front you do not have to be looking to solve a CRM (Customer Relationship Management) problem to benefit from this book. In fact anyone building solutions that will help automate and manage internal business processes can benefit from its insight. Non-developers, meaning system administrators, business analysts, and even project managers might find the book useful but will have to wander past any of the real developer-centered discussions.

How to use this book?

Read it cover to cover or simply skip and jump to the areas you need more details on – there is no single correct way to use this book. You could use it online or carry around the printed book to each of your client visits. Write on the pages, fold the corners and make notes of your own ideas. If you end up building something cool, drop us a note and tell us about it.

Contents overview

Chapter 1-What’s new in CRM 4.0 – Microsoft Dynamics 4.0 has significant changes from the basic platform architecture to the tools developers use every day to create solutions. This chapter introduces the changes and sets the stage for developers to understand how to build applications.

Chapter 2- Building Line of Business Applications - Everyone thinks of CRM for managing Contacts and Accounts but Microsoft Dynamics CRM 4.0 can also be used to build line of business applications that are not just CRM. We explore how you can leverage the platform foundation to build applications without doing the “plumbing” yourself.

Chapter 3- The Developer and Team Workspace – One size certainly does not fit all when it comes to developers. This chapter, explore how developers can setup their development environments to maximize their productivity building applications. Included in this chapter are discussions on team development.

Chapter 4- Data Modeling in CRM – At the heart of any good CRM or line of business application is the data model. CRM allows developers to declaratively build the data model and work with it using all the power of the platform. Here we talk about defining the data model and the basic user inputs to manipulate data. This chapter is a must-read for a good understanding of the remaining chapters.

Chapter 5- User Experience Customization – We start with the basics and then move beyond to look at how to do client side development including leveraging some OO techniques with JavaScript (Yes, we said OO and JavaScript in the same sentence!).

Chapter 6- Client Scripting How To’s – Having a basic understanding that we provided in Chapter 5 is great but here we dig into real world examples of common tasks you will want to do with the client side capabilities.

Chapter 7- Building Alternate UI's – The built-in UI in the CRM platform is powerful, but in this chapter, we explore building alternate UI's using Silverlight.

Chapter 8- Exploring Metadata – Building on the data modeling in Chapter 4, we explore how you can programmatically access the platform metadata and use it as part of your development. New to Microsoft Dynamics CRM 4.0 is the ability to use an API to modify the metadata in addition to read access.

Chapter 9- Using the Web Services - Web services are the SOA extension of the data model and dynamically configure to implement and expose the custom data model. In this chapter, we learn the basics of working with the web services to interact with the platform.

Chapter 10- Web Services Common How-To's - Having a basic understanding that we provided in chapter 9 is great but here we dig into real world examples of common tasks you will want to do with the web services.

Chapter 11- Unified Event Framework - Building on our knowledge of the data modeling and the web services we embark to understand the Unified Event Framework (UEF) which is new to the platform in this release. The UEF sets the stage for how plug-ins and workflows are implemented into the platform. A basic understanding is essential prior to developing business logic extensions like plug-ins and workflows.

Chapter 12- Plug-in Basics - Plug-ins let you integrate your custom logic just like it was part of a platform operation. In this chapter, we explore how to build a plug-in and the different options that exist for getting it integrated to the execution pipeline.

Chapter 13- Plug-in Developer Framework - The “hello world” plug-in is easy, but what about real world plug-ins. Here we look at providing a consistent pattern for building platform plug-ins that includes a concept of a test bench for testing plug-ins outside the platform.

Chapter 14- Plug-in How To's – Take a walk on the wild side and learn what type of problems developers are solving using the plug-in capability. In this section we look at real life examples of plug-ins.

Chapter 15- Workflow Re-Energized – It's not just upgraded, it's re-invented and based on Windows Workflow Foundation. Learn about the changes to workflow and how it makes workflow in the platform a powerful tool in your arsenal for business process automation.

Chapter 16- Workflow User Interface – You used to have to be an admin to manage and build workflows. Now, sophisticated business analysts and other non-developers can build them using the web interface. Learn what is there for users so when you build custom pieces that plug into it you will know how users can leverage it.

Chapter 17- Windows Workflow Basics – You don't have to be a workflow guru to take advantage of the workflow support in the platform but knowing the basics will help you get things done quicker.

Chapter 18- Custom Workflow Activities – Building custom activities really shows off the power of the platform allowing you to build activities that users can use via the web interface when building their own workflows. These can be simple or complex, and can be built to support development of more complex workflows as we will discuss in Chapter 19.

Chapter 19- Workflow Developer Framework – As we did in Chapter 13 for plug-ins this chapter looks at how to build a reusable pattern for workflow support including the concept of a test harness for running the workflow or activities outside the platform for easy development.

Chapter 20- Workflow Code Generation – In this chapter we explore ways that you could use code generation to make workflow development more drag and drop and extend on the dynamic SOA features of the platform to workflow by code generating custom activities based on the platform data model.

Chapter 21- Workflow How To's – In this chapter we look at some real world examples of how developers and users are leveraging the workflow capabilities of the platform.

Chapter 22- Building CRM Online Solutions – This chapter focuses on the specifics of building solutions that work with CRM Online. This includes looking at some techniques to work around some of the differences that exist between CRM Online and other deployments.

Chapter 23- Leveraging Multi Currency and Language These two features are at the heart of the global nature of the platform. In this chapter, we explore how to leverage them as part of your applications.

Chapter 24- Packaging for Deployment – Ready, Fire, Aim...oh wait we need to figure out how to deploy this to a real production site. In this chapter, we discuss just that and the features of the platform that help you get your solution deployed either to an internal host or from an ISV perspective in packaging up their application.

Chapter 25- Tracking Down Problems – Nothing is worse than getting the call that something broke and not knowing where to start. In this chapter, we explore techniques to prepare for production problems and what capabilities exist in the platform to help you solve problems.

Chapter 26- Performance Tuning – In this chapter, we look at some of the techniques to squeeze out a little more from your application. We also look at some of the platform metrics provided in performance counters that are used to help identify problems.

Where do I get the Source Code?

There's no need to re-type the examples in the book unless that's what you really like to do! You can download the source code and any other supporting files for the book from the book website at www.thecrmbook.com. Make sure to get on the e-mail list to get update notices if we make any changes to the downloads.

How is the Source Code Organized?

We tried to organize all of the sample code in a way that would be easy to use. The code has been divided into two major sections Framework and Chapters.

The Framework folder contains all the code for the common libraries that you might want to use over and over again. The license agreement EULA for the source code is flexible to allow easy reuse

The Chapters folder contains sub folders that tries to group sample code based on the chapter we are covering. It's possible if an example from a chapter is from the Framework code there might not be a specific folder for that chapter.

What Software Do I Need?

In order to develop using the ideas and techniques presented in this book you will need access to a running copy of the CRM Server. The simplest way to obtain that is to download the demo VPC that Microsoft produces. Because the location of that changes from time to time here's a link that we know won't change <http://www.thecrmbook.com/demovpc/>.

In Chapter 3, we discuss different options in detail for setting up your complete development environment. However, the VPC will get you enough to get going quickly.

Some of the examples and concepts leverage capabilities that are part of Visual Studio 2008 and therefore to take advantage of them you will need access to Visual Studio 2008.

What about stuff we screwed up...You know Errata

In the process of writing, everyone sure wanted to make sure it was 100% accurate, but since nobody is perfect, we are sure you might find a few typos or other things that aren't correct. We would love to hear your feedback and will do our best to incorporate it into the next printing of the book. You can look at our website, www.thecrmbook.com for any last minute changes. Feel free to email us as well, info@thecrmbook.com if you find a typo.

3

Developer and Team Workspace

In this chapter we will look at how to prepare your individual developer workspace as well as discuss some of the things a team can do to be more productive. You might be thinking, “don’t I just download the SDK and I am ready to go?” In reality, developing applications that will run on the CRM platform is pretty straight forward compared to trying to glue all these components together yourself. The platform installation process hides much of the complexity of using all the combined products/components from you. For example, you do not need to configure all the options on Workflow or Reporting Services, this is done for you. With that said there are things that can make you and your team more productive and this is what we will be discussing in this chapter.

From an individual perspective, you need to decide what version of Visual Studio you will use and where you will be pointing for your CRM server during development. There are a few approaches and options, but most of them boil down to personal choice and definitely are not a one size fits all solution. There are also some obviously different answers to the questions depending on if you are targeting an on-premise install versus developing an application deployed on CRM Online. Another big difference is how deep are the customizations you will be

performing – if you will just be doing JavaScript or forms customizations or will you be doing deep code level plug-ins and workflow.

From a team perspective, we will explore some of the challenges you might face and some possible solutions. Teams need to handle multiple people making changes all at the same time. This will include changes to customizations, scripts, and even custom code. How to approach and solve this will involve looking at the different phases from development to production. Finally, we will discuss an approach to setting up source control and a project structure for building your solution.

Much of what we discuss in this chapter is personal choice and there are multiple techniques to accomplish the same thing. This chapter will not be dictating a single approach, but give you ideas to develop your own path. Use this chapter to build ideas on how you want to tailor your specific environment to make you and your team the most productive.

Choosing your development machine OS

One of the things you need to decide is what operating system your development machine should be running. By development machine, I am talking about the computer that you sit at all the time doing your development work, not a team server, test server or the production environment.

The reason this is important is because it will affect how you go about debugging and doing other testing during development. If you are just doing customizations and scripting the choice of operating system really does not matter. If you are doing custom work flows or plug-ins to extend business logic where you will be writing .NET code, then the choice can be more important.

The reason really comes down to debugging. Currently, since CRM requires Windows Server as the operating system to run, if you want to interactively debug while running a custom workflow or plug-in deployed then you will need to have Windows Server as your operating system. Another option is to use Remote Debugging, but from past experiences I think it takes way too much effort for the rewards.

Personally, I'm of the opinion that the tradeoffs required for running Windows Server as my day to day machine are just unacceptable. Don't get me wrong, I think Windows Server is great when run on the server. However, on my desktop I prefer to run a desktop/laptop OS. In fact,

just the thought of having Windows Server as my daily OS reminds me of doing MTS COM development years ago when that was the only way to go. I think the problem is the goals of a server OS are much different from a desktop/laptop OS especially when you start talking about doing development on a laptop.

In reality it's not so much an OS choice as a difference in philosophy on how I do my development. One of the things I do is to provide ways to do initial unit testing without needing to deploy to the CRM Server, but rely on deployment to a test server for full in platform testing. I then look for ways that I can make my local development environment more productive. For example by using a test harness for Workflow and Plug-ins I'm able to do debugging of them locally without having to have Windows Server be my desktop / laptop OS.

I will talk extensively about how to build a test harness like this later in the book in the respective workflow and plug-in chapters. I also establish a goal for myself that by the time my code works in the unit test environment I need to start thinking about production. What do I mean by that? A way too common development scenario is that developers rely on the debugger until the time that their code is deployed to production. That works great and they are pretty productive but once their code gets to production "where's the debugger?"

These developers are stuck trying to figure out what is going on in their code using a process I think is akin to walking blind-folded in a room and using the sound bouncing off the wall as a clue to where you are. I tend to like to start early thinking about how I will resolve problems in production. Like the CRM platform I have found that tracing is my friend and by building a robust tracing infrastructure and baking trace statements into my code I'm prepared for that call that we all get telling us something isn't working. Using the test harnesses, we will be discussing tracing extensively as we go through the workflow and the plug-in chapters.

So the bottom line is there is no single right OS for your development machine, it really comes down to personal choice. In fact, if you really get down to analyzing some of my reasons for not using Windows Server as the O/S, such as using test harnesses and tracing, they can all be done the same way using Windows Server. Regardless of your choice it is worth thinking about upfront.

NOTE

If you are doing work with a 64bit client O/S keep in mind that there are not currently 64bit binaries for the client add-ins. That means some of the Outlook assemblies we talk about referencing require you to force the application to 32bit.

The Virtual Machine Approach

Another common approach is to use a VM (Virtual Machine) that runs either locally or on a remote server that you access for your development. When using this approach what you run locally as your base O/S really does not matter.

The nice thing about this approach is it does give you a lot of flexibility to maximize hardware. Using this approach you can build out virtual servers that support each project or client. You can also combine virtual servers with the Multi-Tenant capabilities of CRM to really have a flexible environment to support your development. When there are multiple developers virtualization can provide each their own sandbox to work in.

Using Pre-Built VMs

Microsoft makes available a pre-built virtual server with CRM pre-installed that you can just download and go. This is simply the easiest way, other than signing up for CRM Online, to get hands on experience using CRM. Other than the time to download, you should have a CRM Server up and running in a matter of minutes. If you're reading this book in order to get an idea of what the capabilities are of the platform and you want an environment to play with, while investing little, this is the way to go. Visual Studio 2005 will be pre-installed on it, so you will also have development tools ready to use.

NOTE

Many of the examples in this book require Visual Studio 2008 to get the full advantage of the concepts. Using Visual Studio 2005 is still viable but you should understand that some things might not apply. For example, JavaScript! IntelliSense is only supported in Visual Studio 2008.

If you are committed to the platform and are doing a real world project using it, it is highly recommend building your own virtual server(s) rather than using the pre-built server image.

Building your own virtual server image will require a bigger investment (typically a few hours of time) but you will have the following key advantages over using the pre-built image.

- No data will be there, just what you input after the install – you can always download the Adventureworks database from Codeplex if you need sample data later. Microsoft also makes available import files for most of the common built-in entities that you can import directly in as needed.
 - <http://www.codeplex.com/MSFTDBProdSamples>
- You will be sure there are no customizations already done to support the demo ware on the box.
- No “bloatware”, you know what’s installed and it’s only what you need – If you need a demo version of something it will be you that installs it.
- Most important, you will understand the install process – even if it is a simplistic environment compared to what you will have in production. Knowing how the pieces fit together will help you when problems come up or more detailed knowledge is required.

The Disposable VM

In addition to using the pre-built images to get your feet wet quickly, they also are great if you need a disposable environment. Often times if you want to try a risky change or a 3rd party tool trying it first in a throw away server is always the best. You can also use the Differencing disk or Undo capabilities of Virtual Server/PC to do the same with the pre-built images or your real environment.

Virtual Server, Virtual PC or Hyper-V

All three work fine and in fact, you can share the VM easily between Virtual PC and Virtual Server environments. One thing to keep in mind if you are moving between different versions or flavors of the product, each have their own Virtual Machine Additions software to install. I have

personally seen performance of both CPU and I/O be horrible (absolutely horrible) if the VM Additions software is mismatched.

Windows Server 2008 brings a new option called Hyper-V. Hyper-V is the next generation of the virtualization technology and is more tightly integrated with the O/S. Using Hyper-V would give you more flexibility because it offers features like snap shots that can be used to save copies of your VM at a point in time for later rollback. This feature could be very helpful if you are trying something risky or experimenting and want to get back to a stable point. You can accomplish similar things using undo disks with the older version; snap shots are just easier to manage. Hyper-V also promises a number of performance improvements due to how I/O is handled. Startup and Save time for the virtual machines is noticeably faster on Hyper-V than on Microsoft's other products. Which of these flavors you pick really depends on your target OS for running it. Check the latest on the Microsoft site and I recommend using the newest version for your specific target OS.

Hardware Recommendations

There are just too many combinations and possibilities to give absolutes here but here is my rule of thumb for an all in one server environment. You will want at least 2GB of memory on the host OS. Quantity of memory and speed of hard disk makes the most difference. When possibly getting closer to 3-4GB will allow you to allocate a larger amount to your virtual machine and it will make a noticeable difference.

For laptops, I highly recommend 7200 rpm hard drives and on a desktop or server 10,000 rpm drives or faster make a big difference. Multiple drives also can provide an advantage because you can have your virtual server running on a separate drive from the host O/S.



NOTE

For your own personal developer workstation – if you do not currently have a 10,000 rpm drive but you can get your hands on one then definitely get it . Earlier this year I upgraded my desktop machines and was amazed how noticeable of a change it was. If you have to choose, then buy a slower processor and spend more on faster disks!

MVP Tip : Microsoft CRM MVP Matt Parks added that the same is true of external enclosures and 7200rpm drives. Especially when using 2.5" drives, most are 5400 RPM. He also mentioned

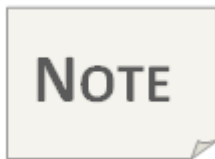
that the interface used by externals makes a difference also. He found that upgrading his to eSATA and an Express Card made a world of difference when compared to USB 2.0.

Building our own

Over the course of the next few sections we will set out to build our own VM environment from scratch. This will be a single server environment intended for development or testing.

The process here is intended to give you a basic idea of how to build the environment but will stop short of an elaborate step by step process for all the components.

The steps are intended for someone with basic system administration / network administration skills, otherwise you might need to look for more detailed instructions online where the process is not detailed in a systematic fashion.



The process here is intended to help get you going, it is not a replacement for the Microsoft CRM Implementation Guide that provides a wealth of detail on all the setup options.

Setup Windows

In VPC2007 or Virtual Server you should setup a new virtual computer and proceed to install Windows Server. The order of the various steps is important so follow them in order.

1. If you have two or more drives in your system try to place the virtual hard drive for the virtual machine on one other than your system boot disk. This will help slightly with performance. The faster the drive the better. If you are using an external drive make sure it is faster than your internal drives.
2. Allocate as much memory as you can to the VM without causing your normal machine to slow down too much. You can set it higher during the install and it will help it run faster and then just reduce it later.
3. If you haven't done an install like this before, you will use the CD/DVD menu in Virtual PC or from the Properties page in the Virtual Server to attach the machine you created to the Windows Installation ISO file or to the Physical CD drive on your host if you are

installing using physical media. The Windows Server install ISO or CD/DVD is bootable and this will allow the VM to begin the installation when started.

4. Install the O/S
5. DO NOT join this machine to your domain! We will be setting it up as a standalone domain after we install some other software.
6. Once the O/S installation is completed, connect up to the Internet and apply any updates that your O/S is eligible for.
7. Once the installation is completed, you will want to enable Remote Desktop on the virtual machine. You can find this on the My Computer -> Properties option.

Promote the Virtual Machine To a Domain Controller

The next step is to promote the server to a domain controller and install IIS.

1. You can promote the server to a domain controller using either dcpromo from the command line or using the Configure Server wizard on the Administrator menu and selecting the Domain Controller role.
 - o This will setup DNS, also make sure that you remember the password for restore.
2. Install IIS either via the Add Programs or the Add Application Server role. Make sure you enable ASP.NET.

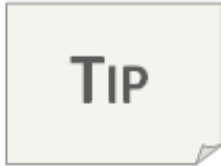
Install SQL Server

Now we need to install SQL and get it ready to handle our CRM Data.

1. Install SQL 2005 (or 2008 once it's released and supported).
2. It is best to use Standard Edition or greater.
3. Make sure you choose to install Reporting Services otherwise CRM will not pass the validation checks
4. To make it easy to work with SQL from your host, I recommend setting Mixed Mode Authentication for this install. Typically, Windows Authentication only would be a better choice if this were a production server. Using Mixed will allow you to easily connect from the host that is in a different domain.

Install Visual Studio

The next step would be to install Visual Studio if you intend to use the server for debugging using it. This is not always required but if you prefer to debug locally you will want to install it now.



Be a little more precise on the options when doing the installation of Visual Studio to your VM. There is no reason to install options that you will never use as part of your CRM development. Remember, keeping your VM hard disk size as small as possible is the goal.

MVP Tip: Microsoft MVP Guy Riddle adds that he also installs Word, Excel and Outlook so he can test other components of CRM. He also finds it useful to add Adobe Reader and a zip product. It adds a little more to the image but you will miss them if they aren't there. You can read his blog at <http://guyriddle.spaces.live.com>

Install CRM

We are now ready to install CRM.

1. Copy the CRM install file to the server and expand it into a local drive. That happens when you run the “.exe” file that you downloaded.
2. If you don't have Internet access to your VPC you will need the following downloaded before you begin. If you are using the physical media for CRM these are on the DVD but not in the download install.
 - The easiest way on these is to download each and install them prior to doing the CRM install.
 - Microsoft Application Error Reporting prerequisite can be found in the \Server\DW folder inside the CRM Server directory you expanded.
 - VC++ Redistributable is a challenge because it doesn't detect that it's already installed if it is so we have to accommodate for that. To do that we will create a \Redist folder and place the install files there so CRM Setup can handle forcing an install. Download and place the files in the following folders inside where you expanded the CRM Server install files
 - <CRMInstallFileFolder>\Redist\i386\VCRedist\vc redistrib_x86.exe

- <CRMInstallFileFolder>\Redist\i386\VCRedist\vc redistrib_x64.exe
- <CRMInstallFileFolder>\Redist\amd64\VCRedist\vc redistrib_x64.exe



NOTE

You can download the trail from the public Microsoft Downloads or you can download from MSDN as well if you are a MSDN subscriber.

Install CRM (Continued...)

3. Inside the folder where you expanded the CRM files create an ifdconfig.xml file and add the following content. (replace \$OrgName\$ and \$DomainName with the appropriate values).

The following is an example of the setup xml configuration file.

```
<CRMSetup>
  <Server>
    <ifdsettings enabled="true">
      <internalnetworkaddress>
        192.168.1.1- 255.255.255.255
      </internalnetworkaddress>
      <rootdomainscheme>http</rootdomainscheme>
      <sdkrootdomain>$OrgName$. $DomainName</sdkrootdomain>
      <webapplicationrootdomain>
        $OrgName$. $DomainName
      </webapplicationrootdomain>
    </ifdsettings>
  </Server>
</CRMSetup>
```

- The ifdconfig.xml will enable the Internet Facing Deployment capabilities. We want this enabled so we can test our applications using that as an option as well as Active Directory. You can disable and enable this at will by changing a registry setting, more on that later in this chapter.

- Now start the CRM install, running from a command prompt (Start – Run – cmd) the following command
 - Setupserver /config ifdconfig.xml.
- Once the install has completed, your server is setup. It's a really good idea to re-boot your VPC, even if you are not prompted to do this by CRM.
- Once rebooted, try to access the server and make sure it works correctly. Unless you specified a different URL or port other than the default you should be able to point your browser to <http://localhost:5555> to verify.



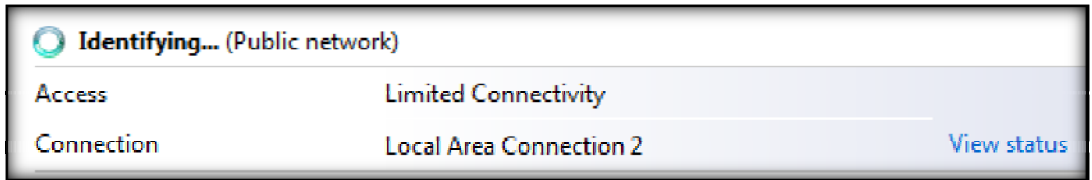
Enabling IFD is not required; however, it can be very helpful to have this available for testing. }

Install and Configure Loopback Network

Now that we have setup a standalone Windows Server, our next step is to build an isolated network. To accomplish this we are going to install a Loopback network adapter. Unlike a regular network adapter this network adapter will only allow communication between our host computer and the virtual machine.

On your Host Machine, not on the new server VM, perform the following.

1. Launch Add Hardware Wizard from Control Panel (From Windows Vista this is easier to find using the Classic Control Panel view). Click Next to start it.
2. Do not select Search for and install, instead select the second option – Install Hardware that I manually configure.
3. From the list, choose Network Adapters.
4. It should show you a list of Manufacturers, select Microsoft.
5. On the right side list, select Microsoft Loopback Adapter, and click next to allow it to install the adapter.
6. From the Network and Sharing Center select View Status on the newly added connection



- On the dialog that shows, select Properties to modify the adapter settings.
- Select the Internet Protocol Version 4 in the list, and click Properties.
- Change the IP address to 192.168.1.2 and the DNS IP to 192.168.1.1

On the New Server VM, Not on the Host perform the following

- Change the IP Address of the Server to 192.168.1.1.

In Virtual Server or Virtual PC2007 perform the following

- On the properties for the new Virtual Machine that you created there will be a Network item. Edit that and select the Loopback Adapter.

At this point you have created an isolated network between your host and the virtual machine that we are creating. This is good as it will allow you to have a sand box that can't do things like send out e-mails to external people by accident!

Prepare for Reuse

The next set of steps are not required to be done, however will make it so if you need another VPC or kill this one by accident that you have a base copy that you can reuse. It also uses a technique called differencing disks. Differencing disks are a feature of Virtual PC/Server to allow you to start with an original VHD and then the differencing disk only stores what is different from the original. That means you can setup 10 CRM VPCs each having their own differencing disk that points to our disk we built in the above steps and all that would be stored would be the delta of the data for the drive. For example, a complete copy for each would take 7-12gb, but a differencing disk might only take 1-3gb depending on use.

1. If you would like, create a bunch of accounts for testing in the Active Directory. This will allow them to be available on all copies and will save you time later.
2. Shutdown the VPC and mark your existing VHD file read only. This will be our base image (keeping a copy of it on DVD or external drive isn't a bad idea either!).
3. Create a new Virtual Hard Drive – select type of Differencing. For the parent point to your base image that we just marked read only.

4. Now create a new Virtual Machine and use the differencing VHD (Virtual HardDrive) as its hard drive – you can repeat this process for as many developments or testing environments you want! Or after you accidentally destroy your current one with too much trial software installations – no worries just build a new one!

Using, Enabling, Disabling IFD

Using the following steps, you can access your server from the Host machine using either IFD or Active Directory.

1. To access CRM via IFD from the Host OS you will need to add the IFD URL to your hosts file (c:\windows\system32\drives\etc\hosts) and map 192.168.1.1 to \$OrgName.\$DomainName (or whatever you decided to call your deployment).
2. The above step should be repeated for each new organization you create if you are using the multi tenant capabilities to have multiple organizations. Multi Tenant is the new feature in CRM 4 that allows one installation of the software host multiple organizations.
3. By default with our setup, while we enabled IFD, we set a subnet mask that will treat our host as being internal and use Windows Authentication.
4. The IfdInternalNetworkAddress registry key allows us to switch that in our environment. When the caller's IP address fits inside the network IP range and subnet mask provided we will be considered internal and get the "Windows Experience" for login. This means we will use Active Directory.
5. If we modify that value to not include our Host IP then we will get the "IFD Experience" and will see the IFD login page.

This registry key can be found at the following location:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSCRM]\IfdInternalNetworkAddress"

Set the following values depending on how you want authentication to work.

- IFD Experience = "192.168.1.1-255.255.255.255"
- Windows Experience = "192.168.1.1-255.255.255.0"

That is the short version of toggling IFD on / off. The Implementation Guide and the IFD Setup document have much greater detail on all the options that can be setup.

Wrapping up VM Building

A big Thanks to Philip Richardson from the Microsoft CRM product team. The VM install was adapted for the book with permission from a couple of blog posts by Philip on his blog at <http://www.philiprichardson.org/blog>.

In this section of the chapter, we have looked at how to build out a reusable virtual machine environment that you can use as your development environment for building numerous solutions.

Test Servers

Another topic that you should spend a few minutes on your test environment. Depending on the formality in your organization, a test environment could take a number of different forms and we will only be touching on some concepts here. Your first decision really has to do with real physical servers or virtual servers. If your goal is simply functionality testing it really is not a big deal either way if you use virtual or physical machines. If you plan to do any performance or volume testing you will want to use physical servers and avoid virtual servers.

Your next decision is one server or multiple servers, each with their own role. Typically for us, when we get to testing we go the multi server route. This means separate domain controller, separate SQL Server and finally a separate server to run CRM. New in CRM 4.0 is the ability to further separate out the roles that the CRM server performs. We typically do not separate that out in test unless we are doing performance related testing.

Testing on production

One question I am often asked right after talking about the multi tenant capabilities is can we put test and production on the same install. I always nervously answer “technically yes, but....” followed by a discussion on how just because you can, does not mean you should.

I do not think it is a good idea to mix test/production because while multi tenant gives you a good amount of isolation at the data and configuration layer, it does not protect you at the O/S layer. The purpose of testing is to prove an application is ready for production and often requires hammering on the environment to work out any quirks. In addition, often you will have hotfixes or updates that update either the O/S, its associated software or CRM itself. These type of updates are best tested in an environment with your application but isolated from the production site.

I do however think that if you have the enterprise edition where you can use multi tenant it is a good idea to have a spare organization configured on the production hardware that you can use for verification. When I say spare, I'm saying basically another organization on the same CRM installation that has a different organization name but all the same customizations as your production instance installed. By verification, I mean that you can test to see if something that is not working in your production environment also does not work in another organization. That is a great way to isolate organization issue from a hardware or an operating system related issue.

Controlling Dev/Test Server E-mails

One of the new features of CRM 4.0 is the ability to have e-mail sent/received directly via Outlook rather than requiring the centralized e-mail router and an outbound SMTP host.

When this feature is enabled, e-mail activities created by the user or on behalf of the user (e.g. via plug-in or workflow) will be set in a "pending send" status until the CRM Outlook client connects and deals with sending them out. This is a great feature and I think it will work much better than having the router for a number of different deployment scenarios.

So what's the issue with e-mails on your dev or test systems? Imagine this scenario; you have a copy of production data and you're doing testing of some enhancements. Your solution uses workflow or plug-ins etc. to generate e-mails or perhaps your testers create them as part of their testing. This is especially true if sending e-mails or automated e-mails is a key part of your solution. Prior to CRM 4.0 you could simply make sure you didn't have a viable way for the e-mail to go out, e.g. point to a bad mail server. CRM 4.0 allows you to configure the install without specification of an e-mail server. This leads you to think that no e-mails can go out if I don't have an e-mail server configured right? Wrong...

Here's how this can happen. You start testing and generate a bunch of pending e-mail activities. Then, someone says "hey let's test inside Outlook and make sure things work there too!" so they or the unsuspecting network admin installs the Outlook client. Assuming that Outlook client is connected to a valid e-mail account, soon Outlook starts and the CRM Add-in connects up to the CRM server. Surprise...all those queued up e-mails will be sent. If all you had was some test e-mail addresses in the data no problem, but what if you had some valid e-mail addresses sitting around... Usually about 5-10 minutes after this happens and panic sets in "Hey Joe, did you just e-mail a buy 1 get 5 free offer out?"

How can you avoid this? Well there are a few options. First you could just use the word of mouth approach along with hope and pray and just tell people to not turn on that option. When you create CRM Users the user is configured by default to route via the CRM Outlook client. So you could go change that option to route through a nonexistent e-mail router. You could also do a mass modification of contacts, and users etc. to invalidate the e-mail addresses. That approach is not perfect either because all it takes is someone adding a new valid e-mail address and then another user sending to it. The reality is you may actually want some e-mails to go out but want to make sure it is to a "white list" of e-mail addresses.

In a normal .NET app we control this by using a common set of code to send e-mails and then it has a white list as well as an always "CC" list so we can ensure we never have an accidental e-mail leakage. Applying this same thinking to CRM 4.0 requires a little more thinking because a user could create the e-mail via the UI or it could happen from the web service.

Import Organization / Redeploy

Another common method of handling setup of a testing environment is to use a copy of a production organization and restore it to a test environment. In CRM 3.0 this was accomplished using a redeploy process. In CRM 4.0 it has changed a little and is now part of the capabilities of Deployment Manager. Inside Deployment Manager when you right click on the organization node you can select Import Organization.

Prior to using Import Organization you would perform a database backup of the production or source CRM organization database and restore it on your test SQL Server. Once you have done that, when you enter the Import Organization process you will be able to select that database from those available on the server for importing.

The Import Organization wizard will walk you through mapping user accounts that exist in the CRM database being imported. You will have a few options on how to handle the mapping including mapping users to other users in your target domain. During these initial steps, you will also specify the new organization name. Once all that is input the process will import the data and the new organization will be ready for use.

Using this approach you can pretty quickly pull data back to the test environment. If your test environment is hooked up to a viable network you may want to keep in mind some of the discussion earlier in this chapter on Test Server e-mails.

Now that CRM import/export of customizations supports additional items like reports, workflows and roles you might find some times where redeploy/import would be used you can just move the customizations between the environments.

What version of .NET

For building any server side custom code Microsoft CRM 4.0 requires all development to be done using the .NET 3.0 Framework. This would include developing plug-ins and custom workflow activities as well as ASP.NET applications, if they will be deployed in the same address space (IIS Application Pool) as the CRM code.

If you're building a standalone ASP.NET application in a separate address space or a client application, or for that matter a standalone executable / service that will just access the CRM Web Services then you don't have the same limits on the version of .NET.

CRM 4.0 and .NET 3.5 Framework released within months of each other and as a result CRM was not fully tested with .NET 3.5 at the time of initial release. The .NET 3.5 Framework release consists of new capabilities plus a hotfix to the .NET 2.0 runtime to allow it to support the new capabilities. In fact, many of what would appear to be new runtime changes are really compiler changes related to the C# or VB.NET language.

As of the release of this book, we have done CRM related work that used .NET 3.5 on the CRM server. That said, the official guidance offered by Microsoft is to use .NET 3.0 until the .NET 3.5 testing has been completed. Keep your eye out for updated information on this either on the book site or from Microsoft.

What version of Visual Studio

Visual Studio 2005 or 2008 is required if you are doing any code level development. If you are just doing client script or form customization neither is required but can be helpful as you will see later in the book. Visual Studio 2005 works fine for developing solutions however, our recommendation is to use Visual Studio 2008 if possible. A few of the key features of Visual Studio 2008 that we believe are useful.

- For client side script development JavaScript coloring and IntelliSense is supported – in fact we leverage the IntelliSense in this book's examples to customize it with the platform meta data.

- If you're building any ASP.NET customizations you can take advantage of the improved visual design features of Visual Studio 2008 like split view between source and design view.
- Multi targeting lets you use .NET 3.0 for CRM deployed components and .NET 3.5 or .NET 2.0 for other using the same tool.
- Improved debugging support

Referencing the CRM Assemblies

Several of the assemblies provided with the SDK are essential for your CRM development. You will be referencing these often when you build your applications.

Microsoft.CRM.Sdk - This assembly contains the base classes you will use for building Plug-ins, workflows and other extensions.

Microsoft.CRM.SdkTypeProxy - This assembly gives you access to the proxy classes – this is particularly important when you are accessing the CRM services or Metadata Services as the proxy classes live in this assembly. Also included are the Request / Response classes for each built-in entity and platform message.



You will notice that this assembly also includes some of the standard entity classes that the web service exposes such as contact or account. These are provided for internal use and could be removed in the future by the product team. It is recommended you use Dynamic Entities when working with the CrmService from this assembly. More on that in Chapter 9.

If you install the client for Outlook then these assemblies will be placed in the GAC on your machine. This is also true if you are developing on a server that has the CRM Server installed. Otherwise, we typically find a common location such as a folder in a project and reference these assemblies from there so it is consistent on all developers' machines.

Microsoft.CRM.Outlook.Sdk – This assembly is provided to help you if you are building solutions to run on the client and access CRM when the user is offline. Using this assembly you are able to detect the state of the Outlook add-in and know if you are on or offline. By default this assembly is not available on the server.

Referencing the Web Services

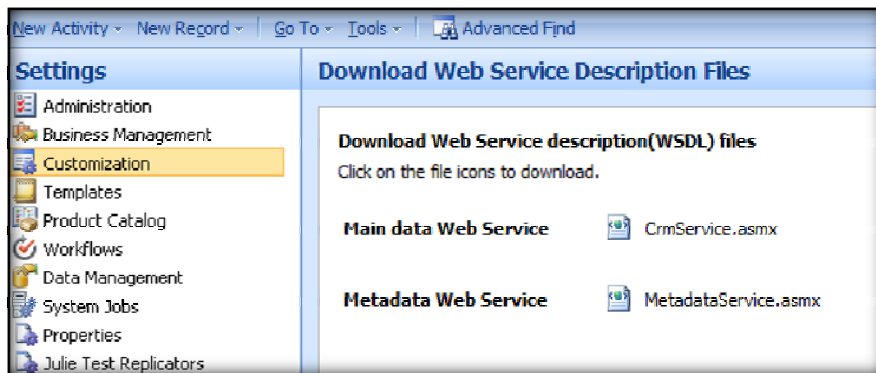
While it's possible to do all development using the type proxy classes in the CRM assemblies we just discussed and not directly referencing the CRM Web Service, you will be missing out on some of the power of the platform. One of the things that happen is as you add custom entities to the system; the CRM Web Service is dynamic and adjusts to provide an interface to your newly created custom entity. This is designed to allow you to have typed access to all the properties on your CRM custom entity.

If you are building an ISV solution or a custom solution that will work in various installations that may or may not have specific custom entities you're better accessing the CRM Services using the type proxy classes instead of referencing the Web Services. In this case you will be doing all of your work using Dynamic Entities.

One of the things you will want to do is adopt a standard name for the reference to the CRM Web Service and Metadata Service. The reason for this is simple. It just makes it more predictable and you do not have to guess what the name space will be.

Downloading the WSDL Files

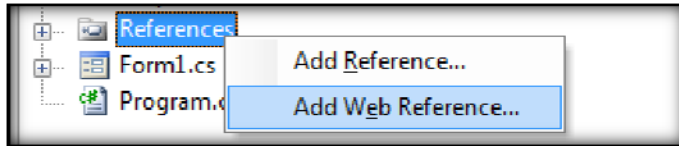
New in CRM 4.0 is the ability to easily download the WSDL files for the web services. The following figure shows that you can find those off the customization menu.



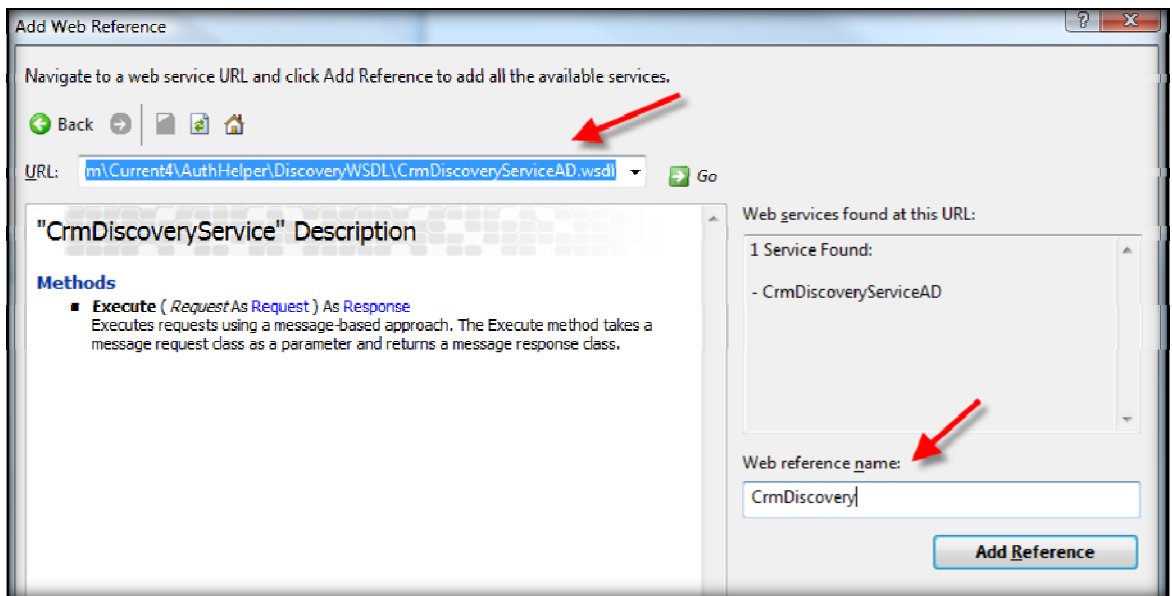
I recommend saving a copy of the WSDL to a local file ideally in your project that will be version controlled. That way you can keep a history of prior WSDL files if needed.

Referencing with Visual Studio 2005

Referencing the web services is pretty simple using Visual Studio 2005. From the project you will be referencing from select Add Web Reference as you see in the following figure.



You should then see the following dialog that will prompt you for the URL of the web service. It's not obvious but if you did store the WSDL on your local file system you can give a file path like "c:\mywsdl.wsdl" instead of a URL.



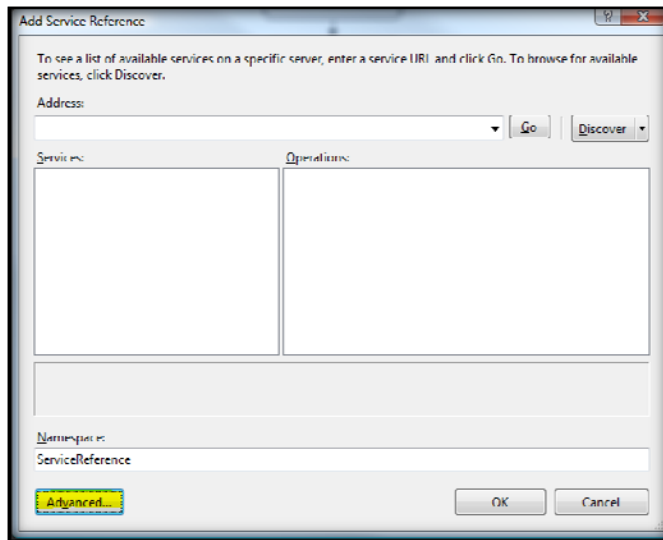
Before you click Add Reference you will want to modify the name of the web reference to be something meaningful. For example, by default it tried to name the discovery service "WebReference" . As you can see we changed that to be CrmDiscovery which simply gives a more meaningful name to reference later.

The process would be the same for doing references to the CrmService as well as the MetadataService.

Referencing Web Services with Visual Studio 2008

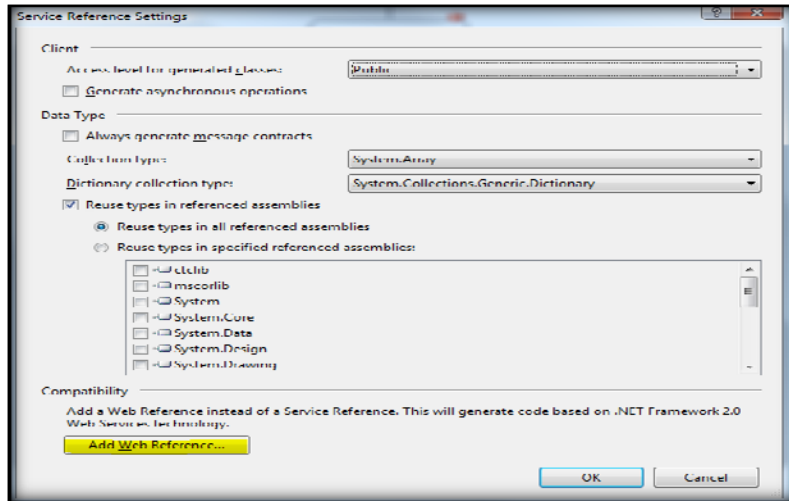
Since Visual Studio 2008 supports multi targeting of the framework version it is possible to use Visual Studio 2008 for doing CRM development. If you are doing that and using a target framework of 3.0 or below, the experience of adding a web reference for the CRM service or Metadata service are similar to what you would do in Visual Studio 2005. If you're using .NET 3.5 as the target framework when you right click on references and are looking for the "Add Web Reference" menu item you might be surprised to not find it. Instead you see a new "Add Service Reference" menu option. That's because Visual Studio 2008 introduces a lot of new features, one of which is improved tooling for managing and configuring Windows Communication Foundation (WCF). Since the CRM web services are ASP.NET ASMX services you need to handle adding the reference a little different.

The following example shows the Add Service dialog.



If you were to proceed with the above dialog, it would work and add the reference but you would likely find yourself with a set of class proxies that don't match any of the SDK documentation or examples. To add a Web Reference to the Web Service that CRM exposes you need to do a couple more steps.

When you select the advance button you will see the following..



At the bottom of the Advanced Settings dialog you will see a button for Add Web Reference. By clicking on that button you will see a familiar page that allows you to add the reference the same way you did in Visual Studio 2005.

Using a Common Project for References

You can certainly do the reference to the web service in each project that will access the CRM services; however, you might want to strongly consider creating a common project to share. Using this approach you create a new project in Visual Studio that will exist for the sole purpose of holding the web reference. Then later as you build other projects that need to reference the web service you simply reference this common project. Because web references expose as public classes you will have visibility to the CRM services.

This approach is also helpful as we talk about helper classes that the SDK provides because you can add them to this common project as well. Later in this chapter we will be discussing building a full solution file that will contain the multiple projects that you will work with in building a solution, keep in mind this approach and add that to your list of projects to create.



This common project makes a great place to also store any common logic for accessing the CRM platform and performing common processing. }

CRM helper Classes

One of the features that was enhanced in the CRM 4.0 SDK is some additional C# helper classes that make the platform web services easier to use. These helpers leverage some of the newer language features that were not available in .NET 1.1 such as partial classes. Using partial classes, the helpers augment the web services adding things like name indexed properties that are not possible to do with just a standard web reference.

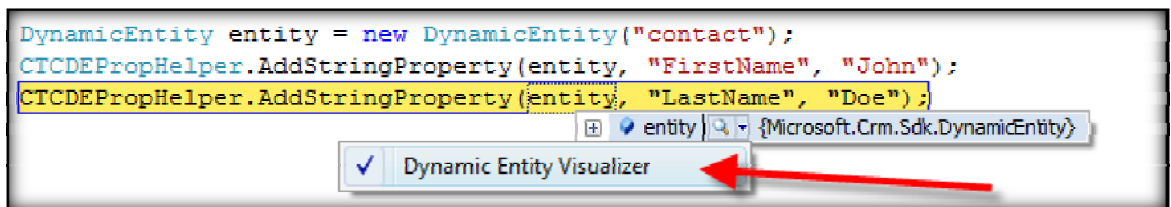
The helper classes are located in the SDK under the <sdkroot>/server/helpers/cs folder. Copy the files from that folder into your project. If you are using the common project to hold the reference as we recommended earlier in the chapter you will only need to do this once.

Once the files are copied into your project you will need to modify the namespace that is used in the files. The reason for this is for the partial classes to work, they must have the same namespace as the Web Reference you added. The Web Reference will have a namespace of <ProjectName>.<web reference name>. The web reference name is whatever you input when you added the reference for the name. So if you used CrmSdk and your project name was MyCRMServiceLib then you would change the namespaces from Microsoft.Crm.Sdk to MyCRMServiceLib.CrmSdk using replace all or replace all in files option in Visual Studio.

Installing the CRM Debug Visualizers

One of the samples that is provided in the book sample code is a Debug Visualizer specifically designed to work with some of the CRM data types like Dynamic Entities. Using this you will be more productive at debugging problems. We will explore the full capabilities in Chapter 25, but here we will focus on installing the visualizers.

In case you are not familiar with Debug Visualizers they basically are custom code you can create that interacts with the debugger and allows you to examine data. When you build a custom one you can make it understand your custom application types. The following is an example of the Dynamic Entity Visualizer and how it looks in Visual Studio.



```
DynamicEntity entity = new DynamicEntity("contact");
CTCDEPropHelper.AddStringProperty(entity, "FirstName", "John");
CTCDEPropHelper.AddStringProperty(entity, "LastName", "Doe");
```

The screenshot shows a Visual Studio window with the above code. Below the code, the variable 'entity' is selected, and the Visualizer window for 'Dynamic Entity Visualizer' is open. A red arrow points to the 'Dynamic Entity Visualizer' window.

To install the CRM Debug Visualizer you should open up the code samples from the book and look in the Assemblies folder. Copy the CRMDebugVisualizers.dll assembly to the folder for your version of Visual Studio.

Typically the following are the folders for version 2005 and 2008.

```
C:\Users\\Documents\Visual Studio 2005\Visualizers
```

```
C:\Users\\Documents\Visual Studio 2008\Visualizers
```

Once you have done that you should be able to use the Debug Visualizers. In Chapter 25 we go into detail on how to use each of the visualizers provided.



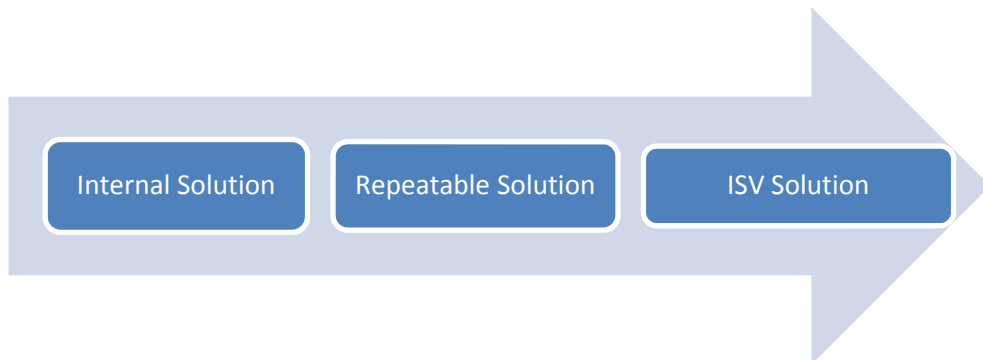
Simple debug visualizers are not that hard to create and are a great way to make debugging more productive. Creating additional ones for the CRM specific types or more complex objects you use internally can pay back quickly.



Managing and Tracking Customizations

Depending on what type of customizations you are doing to the platform, you will have to make some decisions on how to manage the changes. Version control or tracking of customizations is not a feature that CRM provides. It does provide a few hooks and places to help with the process but for the most part it's up to you to implement your own processes and procedures to track customizations.

The extent to which you formalize the tracking depends on the type of solution you are building. The following illustrates the progression from an internal solution designed for internal use to the other end of the spectrum which would be an ISV (Independent Software Vendor) solution.



Often times for internal solutions, having a more informal process is typical and that is leveraged to allow the platform to be more nimble and facilitate rapid development. As you move to the other end of ISV solutions, the need becomes more critical to have repeatable processes and a concept of a “release”. In that case being more formal about change tracking and how customizations are managed becomes critical.

While there is no single answer on the best approach here are some ideas on the various types of customizations and where you might want to store them.

Customization Type	Where / How
Data Entities, Form and Layout	Having one master server works well for this and allows a central point of change. Often times depending on team size you may want to appoint a librarian or keeper of all or certain entities. In Chapter 4, we discuss using the Description field to track a version number for the entity.
JavaScript for Form Events	Store this in a Visual Studio Team Project with version control – so you can not only track changes but also take advantage of the improved editing capabilities. We will explore more of this in the upcoming chapter on scripting. This allows for easy viewing of history and details on changes.
SiteMap/ISVConfig	Store these in the same Team Project as JavaScript but in a separate folder. This ensures consistent navigation each time deployed. You will need to refrain from making some changes in the web editor for this to work – this is not ideal for informal, but recommended for ISV solutions.
Plugins/Workflow Activities and other Code related items	In the Team Project, we will break that out more in detail later in the chapter, these should be version controlled. To allow for easy deployment, they should be in their own class libraries.

Customization Type	Where / How
Reports	Reports built with the Report Wizard are best stored as part of your master server. But those that are customized in Visual Studio should be in a report project as part of your Visual Studio projects.
Security Roles	Best kept in the master server – and moved around via import/export.
Workflows – Web Created	Best kept in a master server and moved around via import/export.

In the next section of the chapter, we will elaborate more on the Visual Studio Project concept and what project types you might use.

Setting Up a Team Project

Like most things in life, there is usually not just one right way to do things and setting up a project for working with a team is certainly no exception. The ideas presented here are just to give you an idea of how you might want to organize and manage source control in your project.

Let's first cover our requirements.

- Must be able to use Visual Studio 2005 or greater
- Must be able to support managing one or more Visual Studio Project types
- Must be able to integrate with some type of Source Control

Let's cover our nice to haves.

- Source Control structured so we can branch (create versions) as we evolve
- Have ability to automate our building of our software

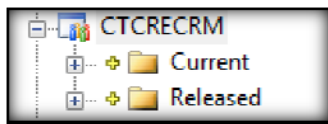
In the example we are going to walk through here, we are going to use Visual Studio 2008 and use Team Foundation Server for our source code control.

Since we are going to use Team System for our source control, we have created a new Team System Project called CTCRECRM. Do not confuse the Team System project with a Visual Studio

project, as they are separate concepts. Team System projects are designed to be a container for Source Control, Work Items (Bug / Issue Tracker), Document Repository (SharePoint), Team Builds and more. We won't go into detail about how Team System works anymore than is necessary for you to understand this project's organization.

Creating the Solution

The first thing we do after we create the Team Project is establish some folders in our source control tree to store our project files. To enable branching of our code as we do development we typically create a high-level folder that will be the container for all our project files so we can branch at that level if we want. In our example, we are creating a Current and a Released folder. Current will contain our active development baseline and Released will be our baseline that will contain our released files once we go live. Code will be promoted from the Current folder to the Released folder as we deploy to the live system. The following shows what our source control tree looks like at this point.

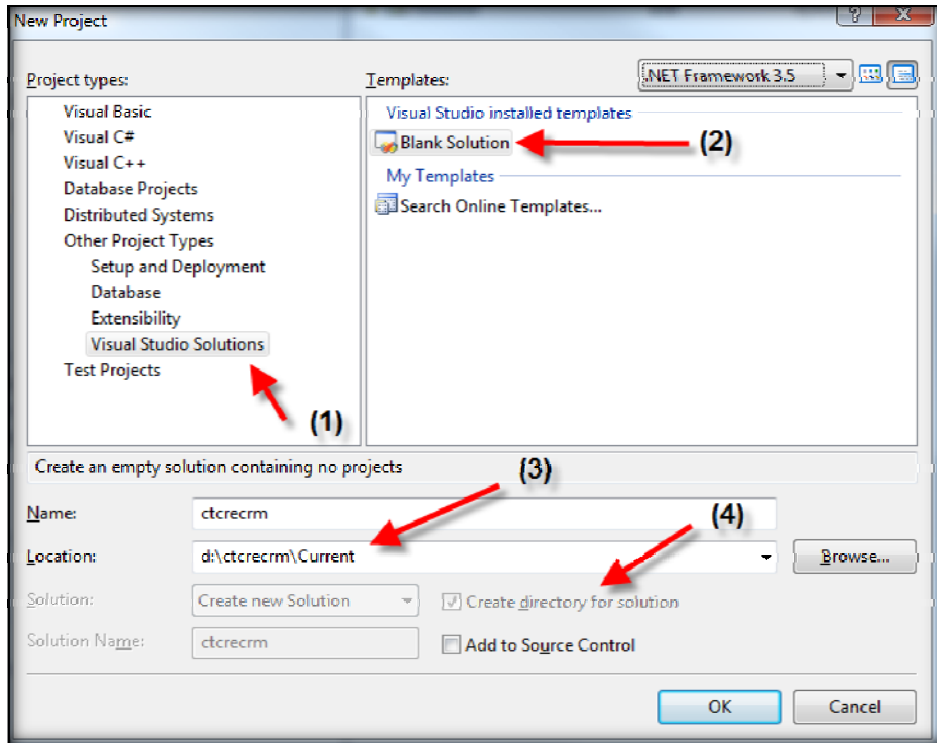


In the example, the branching folders of Current and Released are just examples. More complex projects or products can have additional folders added to allow multiple versions to be tracked.

The strategy you use for organizing your projects really depends on if you are building an internal solution or an external product. Internal solutions typically won't have to support as many active releases that are deployed out at customer sites that might need quick fixes. On the other hand, even for internal solutions it's good to give a little extra thought to how your code might be used.

You may find after some thought that certain parts of your code will be reusable by other CRM projects and should be separated out and referenced. Keep in mind it's easier to start out with a little extra organization and projects than it is to have to try to separate one project into multiple projects later. One thing to be aware of is too many projects and too much organization can be cumbersome to the developers and slow things down. The key is to find the right mix that gives you flexibility without adding too much overhead due to a complex project structure.

Next, in Visual Studio, navigate to the File -> New Project dialog and you should see the dialog we see in the figure. We are going to expand the "Other Project Types" category because that is where they have hidden the creation of blank solutions.



1. Expand Other Project Types, Select Visual Studio Solutions.
2. Select the Blank Solution, this template creates us an empty solution file because we are going to add our own projects to it.
3. For the location select the folder where you have “got latest” from source control of the Current folder.
4. Notice we cannot keep it from creating a directory for the solution. This typically drives me nuts because I really don’t want another layer deep, so I normally cheat here and let it create the folder because I don’t have a choice and then go move the solution file back to the Current folder and then I create all my projects based in the Current Folder and they become sub folders.

Next, I normally close the solution using the File -> Close Solution and I proceed to move the ctcrecrm.sln file from the new folder it created to the current folder and then delete the solution folder it created. I do this just because otherwise it creates an unnecessary layer. In reality, I want the .sln file to end up in the root of the Current folder and all my projects I create to fall nicely under the current folder in their own subfolders.

```
Directory of D:\CTCRECRM\Current
11/18/2007  01:27 PM    <DIR>          .
11/18/2007  01:27 PM    <DIR>          ..
11/18/2007  01:27 PM    <DIR>          ctcrecrm
                0 File(s)      0 bytes
                3 Dir(s)  45,098,463,232 bytes free

D:\CTCRECRM\Current>move ctcrecrm\ctcrecrm.sln ctcrecrm.sln
1 file(s) moved.

D:\CTCRECRM\Current>rd ctcrecrm /s
ctcrecrm, Are you sure (Y/N)? y

D:\CTCRECRM\Current>
```

Next, if I re-open up the ctcrecrm.sln file from the current folder I will want to add it to source control. To do that I select the File -> Source Control – Add Solution to Source Control menu option.

Adding Web Site to Store CRM Customizations

In Chapter 5, we will discuss in detail how to use a web project for storing customizations so I will not go in to great detail now. The idea here is that the web project can act as a container to store, and version control your customizations such as JavaScript files and isv.config type files. Since the web project template already offers a new item template for xml and JavaScript files it is the best choice. It also is setup if you want to create some test pages to experiment with how your JavaScript will work. To add this project you will do a Right Click on the solution file in the Solution Explorer and click Add ->New Web Site. On the dialog that comes up I recommend picking the Empty web project because we don't plan on using any of the files the other templates create. Typically, we will call this web project CRMCustomizations so it's clear what its intent is.

Adding Web Site to Store CRM Custom ASP.NET pages

If you plan to do any custom ASP.NET pages to use with your CRM site, you will want to create another web type project to hold those files. You want to keep them separate from your customizations. You can choose between a Web Site, which is just a folder, and a Web Application, which has a project file similar to Visual Studio 2003. We typically will name this project CRMWeb.

Later in the deployment chapter we will discuss how to add a Web Deployment Project where you will setup the assembly name to be more specific so you don't collide with other applications when deployed.

You also should keep in mind, that since you will likely be running in the CRM web scope, you will want to ensure your pages have a namespace added and not just depend on using the global namespace that is the default when adding new pages to a web folder project. If you use the global and anyone else deploys an application that also depended on global you will have a name conflict with the Default class existing in more than one assembly. To avoid that simply make sure your pages have a fully qualified namespace that is unique to your project.

Adding Common Class Library Project

I always end up with some common code I want to share. Every time I skip the step of creating a common library I always end up regretting it and going back and creating it and moving my common code to it. A common class library will give you a place to store reusable code and encourage creation in there instead of inline in a project file where it can't be shared. I typically recommend creating one project file that is common to the project / product you are creating and another that is intended to be used across projects / products.

Adding Plug-in Class Library Project

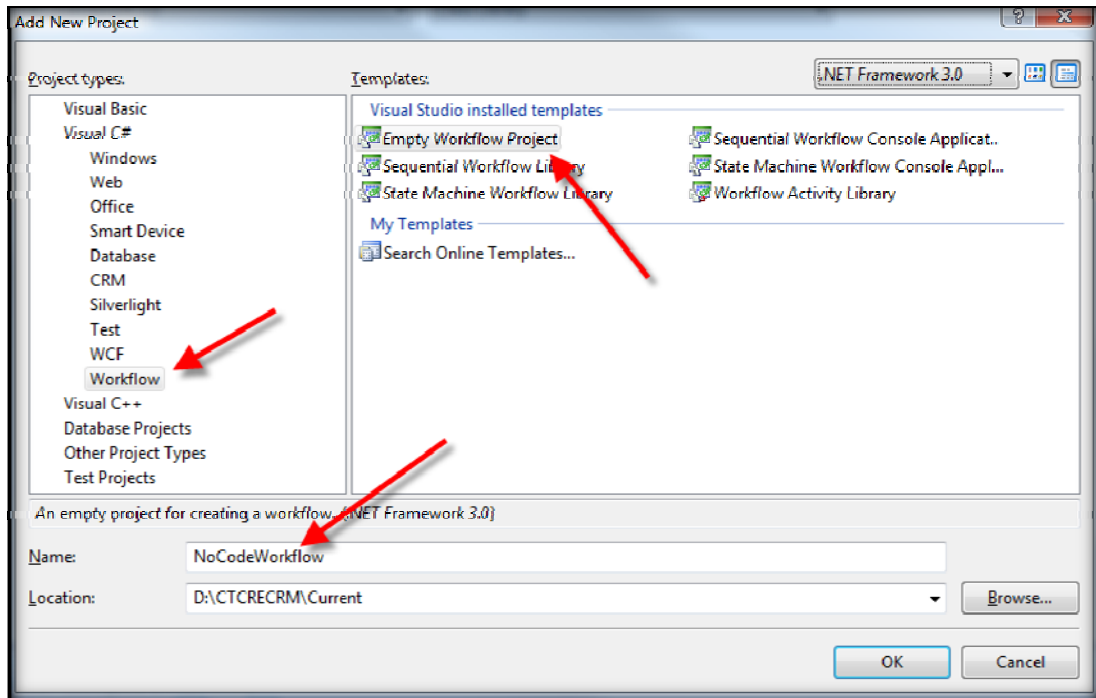
If you plan to have plug-ins as part of your project, you will want to create a separate class library project file to support them. The reason for this is typically for production deployment you will deploy this project file into the CRM Database. Therefore, there is no reason you want to have a bunch of extra code as part of this assembly that is created and registered with CRM.

From the dialog, select a new Class library. You might want to include Plugin in the name so you know what it is when you are looking at your project in solution explorer.

Adding Project for No Code Workflow

We will talk about this in more detail in the workflow chapters; however, if you plan to do workflows in Visual Studio you need to make a few decisions. If your workflows are going to be deployable to CRM Online as well as on-premise, they need to be "No Code" workflows. If your target is only on-premise, you can have workflows that have custom activities and both can have custom code as part of them.

In our example, we are going to setup both, so we will create two separate projects to be clear where we are suppose to maintain and develop each type.



As you can see above there are a few different types of templates. There is not one specifically for a No Code workflow project so the easiest is to pick the Empty Workflow Project.



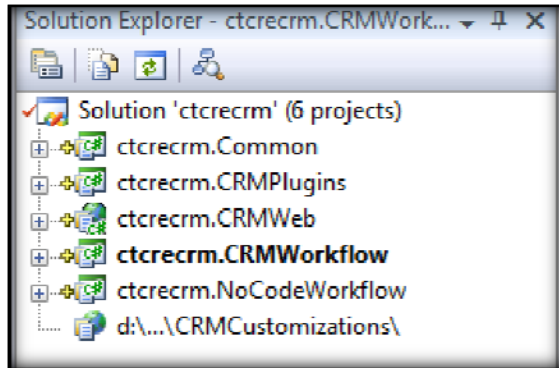
Official support for no-code workflows built in Visual Studio has not been released - the core components are in the platform just hasn't been through complete release testing. Building Custom Workflow Activities is fully supported and discussed in details in later chapters.

Adding Project for Custom Code Workflow

If you are going to be building any custom activities or workflows that include managed code, you will want a place to store them. Technically, you could store them as part of your Plug-in class library but for clarity, we think putting them in their own assembly is cleaner. For example, you might create another empty workflow project and name it CRMWorkflow. Notice I do not call it just Workflow because what happens if later you decide you want to do some non-CRM workflow as part of the solution.

Where are we?

So if we were to look now in Solution explorer we would see that we had created 6 projects to support our product or project we are working on. This gives us some good isolation so if different team members are working on different parts we are less likely to collide with their efforts. You can see from the following figure what solution explorer should look like with our projects.



Notice that while we named some of the projects initially Common or CRMPlugins in Solution Explorer they are prefixed with “ctccre.”. That’s a little trick I do to make things more readable. I want the folder to be called CRMPlugin or Common but I want the project namespace , assembly and project file to be qualified with the namespace for clarity and to avoid collisions with other projects.

Imagine if everyone named something Common and produced an assembly Common.dll how high the risk of collision is. To accomplish this after the project is added I rename the project file in Solution Explorer and I modify the project name space and assembly defaults on the project properties page before I check it in for the first time. Again, this is not required but something I do to make it cleaner and more clear what you are working with in a large solution. There are other options like leaving the name alone and modifying the name space.

Now with our projects setup, our source control established we are prepared to being development and can easily share our projects with a number of team members. In the chapters that follow, we will be discussing more about how to use each of these projects and how they are deployed to a production environment.

It is important to keep in mind that the project structure presented here represents a single idea and not a directive on how it should be done. In fact, there probably isn’t a single right way to

organize solutions and projects within them because of how much it depends on the team, the project and the organization that is building the solution. The important thing is to adopt some structure that works for you and your team.

Automating with Snippets

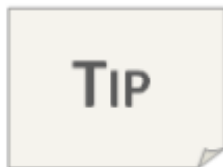
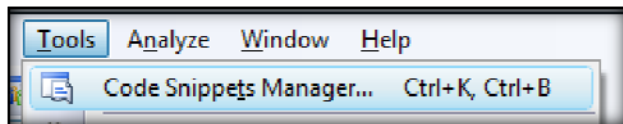
Code Snippets are a great way to package up things you do all the time so just a few keystrokes can be used to add code to your application. If you haven't heard of Snippets they were new in Visual Studio 2005 and have continued to be used to make it easier to do simple tasks.

An example of a common snippet is the property snippet. By typing prop <tab><tab> in to the Visual Studio editor while editing a source file it will walk you through creating a property in a class.

What might not be as obvious is that you can create your own snippets for your custom code. While there are not currently any CRM specific snippets included in the CRM 4.0 SDK, you can create your own library.

In our sample files, we have included a couple to get you going inside a web folder project named CRMSnippets. We have added a few examples to get you started.

Before you can use snippets you must register or import them into Visual Studio. To accomplish that use the Snippet Manager from the tools menu



You can also use the Ctrl+K, Ctrl+B key sequence to bring up the snippet manager if the menu option isn't showing. That menu option will not always show depending on your settings you chose when you first started Visual Studio.

From the Snippet Manager select Add and navigate to the sample files folder CRMSnippets and click Select Folder. You will now see the folders added.

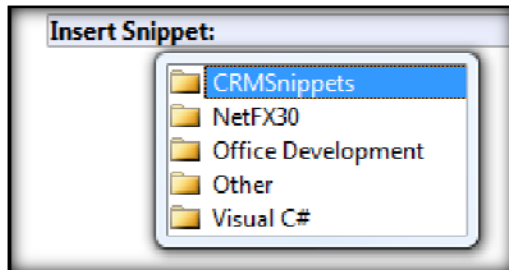
Using the Snippets

Using the snippets is easy. You can use the shortcut if one is assigned to the snippet. In our example the Auth snippet, which is designed to help you connect to CRM, we assigned a shortcut of crmauth. So when we type in the editor crmauth <tab> <tab> we get the following inserted into our source code.

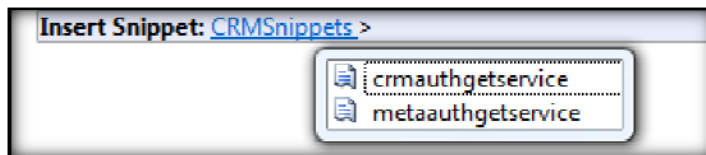
```
    CrmServiceAuthManager<CTCCRMService, CrmAuthenticationToken>  
    crmSVCMgr = new CrmServiceAuthManager<CTCCRMService, CrmAuthenticationToken>();  
    CTCCRMService crmSVC = crmSVCMgr.GetService();
```

The highlighted areas require us either to overwrite them with the correct values or tab past them to indicate the value should keep the default.

Not good at remembering the shortcut keys? Don't worry you can right click and select Insert Snippet from the menu and then you will see the following to help guide you to the CRM snippets.



Once you click CRMSnippets, all the snippets in the folder will be there to choose from as you can see in the following example.

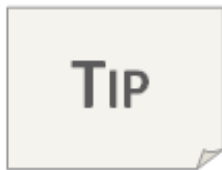


Regardless of which approach you took, shortcut or insert snippet, what is put into the source code is the same.

Building your own snippets

A complete discussion of snippet building is well beyond the scope of the book. However, we wanted to give you an example of how easy it really is to do one. As you will see in the source example of our snippet – they are just XML files.

Visual Studio comes out of the box with a bunch of Snippets and you can download a number of them from the Internet community sites to add on for common items. Similar to how Visual Studio settings work, they don't automatically share with a team – so when you start thinking about building some custom snippets you should also identify a location such as a common project to store them in.



There's not a specific Visual Studio Project Template for creating a library of Snippets – but using a simple Empty Web folder will work fine because then you get XML Intellisense as well as the ability to version control the snippet files.

Snippets are simply XML that explain to Visual Studio what to do when the snippet is invoked. At the top of the snippet is the header where you basically put the author information along with the title and description that will be used to explain how the snippet can be used.

Also in the header is the shortcut. This shortcut is what lets us do the `crmauth <tab><tab>` earlier. You will notice in the markup the actual shortcut is `crmauthgetservice` but since nothing else matched to `crmauth` it was able to resolve that for us.

As you start building your own snippets try to find names that are easy to remember but also try to keep them from conflicting with other existing snippets.

The following is an example of the xml required to build our auth helper snippet we used in our prior example.

```
<?xml version="1.0" encoding="utf-8" ?>
<CodeSnippets
xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>crmauthgetservice</Title>
      <Shortcut>crmauth</Shortcut>
      <Description>Code snippet for an automatically
implemented get of crm service</Description>
      <Author>We Speak You Learn, LLC</Author>
      <SnippetTypes>
        <SnippetType>Expansion</SnippetType>
      </SnippetTypes>
    </Header>
    <Snippet>
      <Declarations>
        <Literal>
          <ID>CrmServiceType</ID>
          <ToolTip>CRM Service type</ToolTip>
          <Default>CTCCRMService</Default>
        </Literal>
        <Literal>
          <ID>CrmAuthTokenType</ID>
          <ToolTip>Auth Token type</ToolTip>
          <Default>CrmAuthenticationToken</Default>
        </Literal>
        <Literal>
          <ID>ConnectionInfo</ID>
          <ToolTip>Optional Connection
Info</ToolTip>
          <Default></Default>
        </Literal>
      </Declarations>
      <Code Language="csharp">
        <![CDATA[CrmServiceAuthManager<$CrmServiceType$,
$CrmAuthTokenType$>
                    crmSVCMgr = new
CrmServiceAuthManager<$CrmServiceType$,
$CrmAuthTokenType$>($ConnectionInfo$);
                    $CrmServiceType$ crmSVC =
crmSVCMgr.GetService();]]>
      </Code>
    </Snippet>
  </CodeSnippet></CodeSnippets>
```

As you can tell, snippets are pretty straight forward to build. As you start thinking about how to keep everyone on the team doing things the same way, consider snippets as a technique you can leverage to help out.

Import/Export Customizations

Assuming you have one CRM server that acts as the master for customizations – import/export is one way to move them to other developers local machines , test or production. Import/Export allows you to pull the data model, the web form and view layout, workflows, reports, security roles ad extract them to a zip file. The contents of the zip file is a xml file named customizations.xml. You can export one or several entities at a time depending on the export option you choose. Export is located at Settings -> Customizations -> Export Customizations.

You can save yourself and others some time by using helpful names to indicate what is in the file that you exported. Sure you could open it up later and try to figure it out, but if it's all customizations then just put that in the name. Remember, you may end up with several of these files lying around and the more you know about its contents and creator the better.

In a dispersed team or where you have outside consultants or partners you may find yourself e-mailing and otherwise moving these files around. I have seen numerous scheme's for encoding the name of the file so you know what's in it, some type of version number or other useful bits of information. One example might be MyOrg-AllCust-1-11-08.zip. In the name I have indicated the org name, the contents and when.

Using the zip file, you can re-import that into the target CRM server. The following are some things to keep in mind about the Import process that are important.

- Import is cumulative – meaning that it will not remove entities or attributes from the target system just because they aren't in the import file.
- If you need to remove entities or attributes, you need to remove it using the web client, or the Metadata API.
- Things like Forms and Views are replaced – there's no partial on them. So if the import file removes a bunch of attributes from a form or view that another person had added the last one in wins.
- Less is better than more, meaning if you can import one or two entities that's less risky and less likely to make hard to find problems than just doing an import of all customizations.

That should get you started with Import /Export, and we will be discussing customizations again in the deployment chapter.

Wrapping it up

Taking the time to make sure that not only your individual development environment is setup for your team can save a lot of time on your project. A lot of the things discussed in this chapter can be addressed and solved using multiple approaches. Many of these approaches are personal choice and not a one size fits all solution. Use the suggestions in this chapter to help you develop your own best practices that you evolve and use on all your projects.