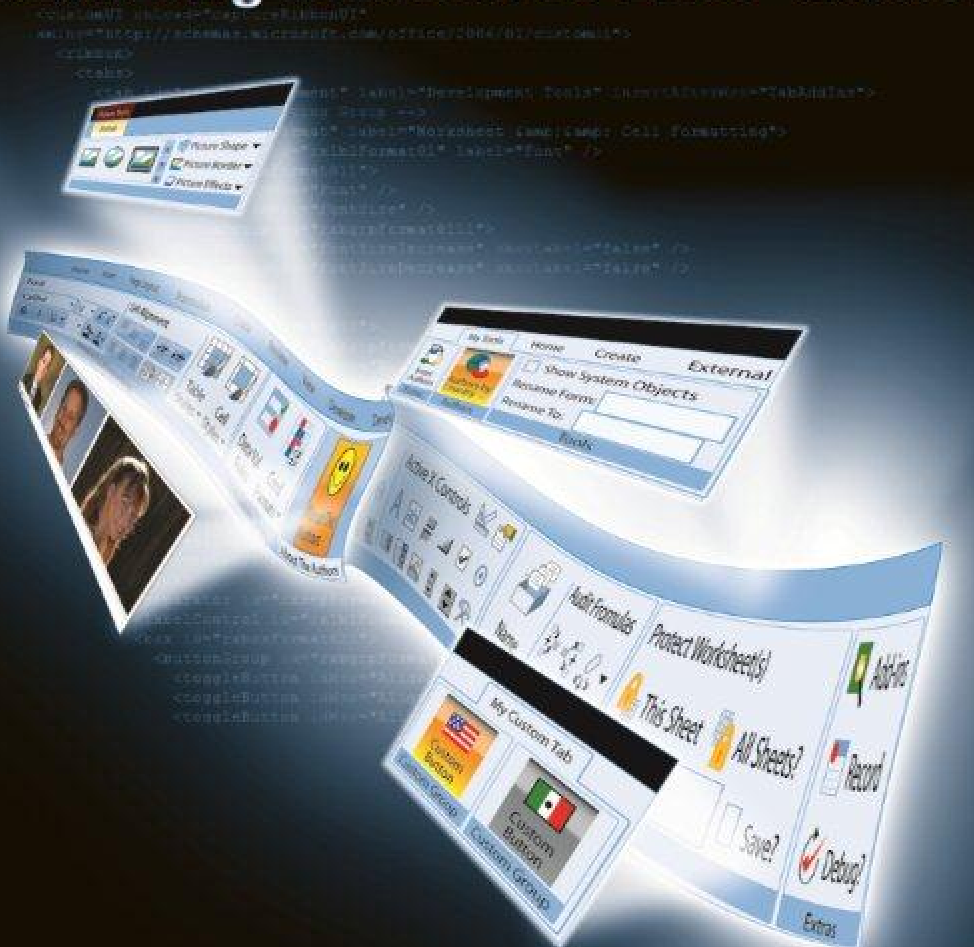
RibbonX

Customizing the Office 2007 Ribbon



Robert Martin, Ken Puls, Teresa Hennig

CHAPTER

17

Security In Microsoft Office

Reality dictates that while most programmers use their power and knowledge for good, there are also those who work with a far darker intent. Because of this, security has always been a big concern to educated Office users. Many safeguards have therefore been implemented to both complement and protect against the great amount of power that has been put at the fingertips of developers in the form of VBA.

The benefits of VBA are incredible, as it provides us with tremendous flexibility when working with a user's system, but VBA also affords the same capability to those with nefarious intent. While we can craft routines that automate entire business intelligence applications and span thousands of lines of code, it only takes a single line of correctly crafted code to render a system unusable. Not wanting to completely remove the functionality and benefits of automation, Microsoft has been left the difficult task of balancing the two sides of this coin: giving developers the access they need while protecting users from those with ill intentions. It is a difficult balance to strike, to be sure.

Fortunately, Office 2007 provides several enhancements to the security model that are targeted at both protecting the end user and making the life of the developer easier. Since every dynamic customization that we create requires using a VBA callback, it is imperative that we understand and master the concepts of security in the Office environment. This chapter discusses each of the concepts behind Office security, both old and new. Our goal is that by the end of this chapter, not only will you understand the concepts, you'll also feel comfortable and confident with the protections that they provide.

Security Prior to Office 2007

Since Office 97, Microsoft has constantly been working on and incorporating ways to improve security in the Office document field. When Office 97 was released, users were given the option to enable macros or not, and that was pretty much the extent of the choices. Because many users wanted macros for legitimate functionality, they would turn off the Macro Security flag, thereby leaving their computers vulnerable to a proliferation of macro viruses that made their way around the world on the backs of e-mails.

In Office 2000, we saw the introduction of *digital certificates,* which enable users to actually sign their code. As this tool is still valid for security today, it is discussed in much more detail later in the chapter; but the basic premise of the digital certificate is that if the code is modified on another machine, then the signature is discarded. If a specific signature is trusted on the user's machine, then the code will run without notification as long as the digital signature is intact.

Along with the digital certificate were three complementary settings of great importance. While these settings have been slightly modified in Office 2007, from Office 2000 through 2003, the user was able to set three security modes:

- High would grant execution rights only to code signed with a trusted certificate, and disable all others.
- Medium would grant execution rights to code signed by a trusted certificate and prompt the user for acceptance of any unsigned documents.
- Low would enable all macros, signed or not, to run without any notification.

Office XP (also known as Office 2002) added yet another wrinkle to the security model, which is still present in Office 2007. By default, each Office installation disables access to programmatically manipulate the Visual Basic Editor itself, as well as modules and userforms that may hold VBA code. While this can cause frustration for programmers who need to set options related to these components, it does add a level of protection by locking down some key areas of the computer system so that malicious code will not have the ability to run amok.

Office 2003 added one additional element to the security model, which was the ability to "Trust all installed add-ins and templates" as if they were digitally signed. This may seem like a minor enhancement, but was an important one for developers, as it allowed them to push out add-ins and global templates without the need to walk the user through installing a self-created or commercially purchased digital certificate. This setting was modified in Office 2007, as you'll see later in this chapter.

Macro-Enabled and Macro-Free File Formats

The first of Office 2007's security enhancements was specifically targeted to help the end user. Since Microsoft revamped the file formats for Word and Excel documents to use the openXML format, they were also able to split files into two distinct subformats: macro-enabled (`xlsm`, `docm`) files and macro-free (`xlsx`, `docx`) files. These two main file formats also have distinct differences in their icons, as shown in Figure 17-1. There are additional file types for each application, but those are not discussed in this chapter.

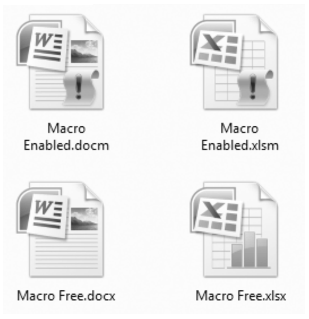


Figure 17-1: Word and Excel file icons

Notice that both of the macro-enabled file formats now bear an exclamation mark, signifying that they (most likely) hold macro code. This distinction provides a very obvious cue to the user that there may be more to the macro-enabled files than meets the eye. A user can take solace in the fact that if the file is saved in a `docx` or `xlsx` file format, it cannot contain any macros, as all VBA code will be stripped from the file when it is saved.

NOTE When working with Word and Excel, keep in mind that macros and VBA are synonymous. Unlike Access macros, all Word and Excel macros are written in VBA.

NOTE Remember that files stored in a binary format, despite the fact that they don't have separate file formats for a macro-enabled and macro-free distinction, can still hold VBA code. The binary file format is used by Office 97–2003 and all Access files.

While it's nice that there is a visual cue to differentiate between files that may hold code and those that don't, we all know that many users will either be ignorant of the symbol and its meaning, or will become so conditioned to cautions that they ignore this warning. This is why the file format should be considered only the first line of defense.

The Trust Center

In addition to the file structure split, there are also other enhancements to the Office 2007 core that affect both end users and developers alike.

Microsoft regrouped all of the security settings and put them into a central place called the "Trust Center." This new collection provides us with a central location for accessing and managing all the security settings related to how Office reacts to files that use potentially dangerous controls. This is important for developers; it is both a "one-stop shop" to configure the end users machine, and it enables developers to configure a specific setting in order to make their own development life a bit easier.

Throughout this section are many references to digital certificates and digital signatures, as many of the security settings reference this particular tool. The full discussion of digital certificates, including their creation, is covered later in this chapter.

The Trust Center can be accessed by clicking the Office button, selecting the application's Options button, and choosing Trust Center from the list on the left. Upon doing so, you are greeted by a screen containing some hyperlinks for learning more about security and privacy. The part that is of most interest to us is the Trust Center Settings. Go ahead and click that button, and you'll be taken to the interface shown in Figure 17-2.

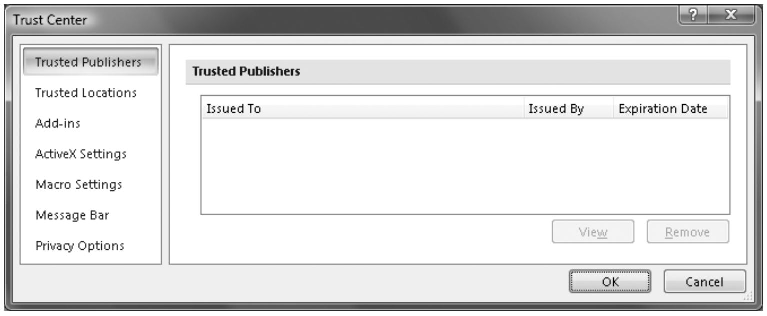


Figure 17-2: The Trust Center in Word

It appears that there are a lot of options here, so let's take a brief look at each one.

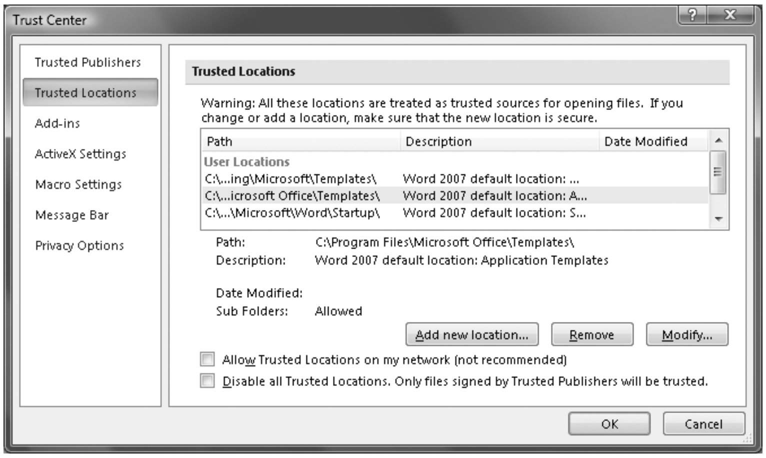
NOTE **Excel actually has one additional item in the Trust Center: External Content. As this is a setting specific to Excel functionality, it is not covered in this book.**

Trusted Publishers

The Trusted Publishers tab, highlighted in Figure 17-2, gives you a list of all of the digital certificates that you have trusted on your system. It is quite common for this list to be empty, but you may see items here if you have ever installed an add-in. Adobe's PDF Writer is a commonly used add-in that will install a digital certificate. Later in this chapter you will learn how certificates are added to this list (see the section "Trusting Digital Certificates").

Trusted Locations

Many of the options that are accessible in the Trust Center were available in prior versions of Office, albeit with some minor changes. There are also a few new features, and from a developer's perspective, the biggest enhancement is most likely the ability to trust locations. Both developers and users will have an instant affinity for the convenience that this setting affords. The interface for establishing Trusted Locations is shown in Figure 17-3.


Trust Center

- Trusted Publishers
- Trusted Locations
- Add-ins
- ActiveX Settings
- Macro Settings
- Message Bar
- Privacy Options

Trusted Locations

Warning: All these locations are treated as trusted sources for opening files. If you change or add a location, make sure that the new location is secure.

Path	Description	Date Modified
User Locations		
C:\...ing\Microsoft\Templates\	Word 2007 default location: ...	
C:\...icrosoft Office\Templates\	Word 2007 default location: A...	
C:\...\Microsoft\Word\Startup\	Word 2007 default location: S...	

Path: C:\Program Files\Microsoft Office\Templates\
Description: Word 2007 default location: Application Templates

Date Modified:
Sub Folders: Allowed

Add new location... Remove Modify...

☐ Allow Trusted Locations on my network (not recommended)
☐ Disable all Trusted Locations. Only files signed by Trusted Publishers will be trusted.

OK Cancel

Figure 17-3: The default Trusted Locations tab in Word

To put it simply, the Trusted Locations tab is an interface that enables you to designate certain folders as "safe." This is a fantastic concept and one of the best enhancements to the Office security model for developers.

Trusting your development folders means that you can avoid all security messages as you load and unload files, without having to go through the extra steps of adding a digital certificate to each project. For those of us who create test files on a regular basis, this can truly save some time. It also means that we only actually need to worry about handling the files that we will distribute to others at the end of the development process.

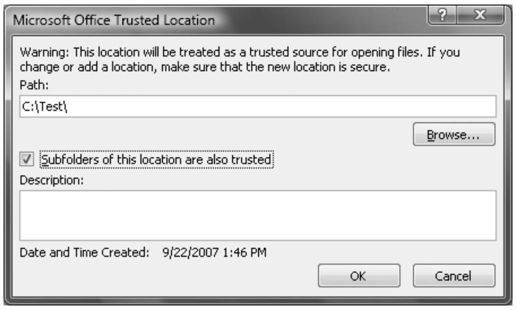
The basic theory behind the Trusted Locations model is, of course, that you will only store in these trusted folders files that you are absolutely certain contain safe code. The instant that you violate that idea, you expose your system to any malicious code that resides in the file, and may as well be running without macro security.

CAUTION **Any file in a trusted folder has full rights to run on the user's system, including VBA macros, data connections, and ActiveX controls. Make sure that you only trust folders that you have control over and know are safe! It is strongly recommended that you *not* trust a location that would be a target of automatic downloads, such as your Documents folder or desktop.**

NOTE **As a means of protecting users from themselves and from malicious script, Microsoft has prevented certain locations from becoming a Trusted Location. The folders include the root of the C:\ drive, as well as the Temporary Internet Files folder.**

Adding, Modifying, or Removing Trusted Locations

To add, modify, or remove folders from the Trusted Locations tab, simply click the appropriate button and follow the prompts. Be aware that when you add or modify a folder, you will see a checkbox asking if you'd like to trust the subfolders as well, as shown in Figure 17-4.



Warning: This location will be treated as a trusted source for opening files. If you change or add a location, make sure that the new location is secure.

Path: C:\Test\

☑ Subfolders of this location are also trusted

Description:

Date and Time Created: 9/22/2007 1:46 PM

Figure 17-4: Trusting a new location, including its subfolders

Why not try giving it a test? Create a new folder on your C:\ drive (or use an existing folder if you prefer), and add it to your trusted folders. Create a new Word or Excel file with the following code in a standard module:

```
Sub Test()
    MsgBox "Hello World"
End Sub
```

Save the file (in the macro-enabled format) on your desktop. Close and reopen it; note that you are given the warning about macros in the file.

NOTE **If you do not get a macro warning, check the Macro Settings tab of the Trust Center and make sure that you have selected "Disable all macros with notification." You'll learn more about the Macro Settings tab momentarily.**

Now close the file and move it into the folder that you trusted. Upon reopening the file, no macro warnings will be present!

NOTE **Trusted locations are application specific. What you set for Excel needs to be set again in Word or Access if you wish to trust the location in all applications.**

Trusting Network Locations

There are two other settings on the Trusted Locations tab, the first of which is Allow Trusted Locations on my network (not recommended). As you'd expect, checking this option allows the user to set trusted locations on a network drive.

CAUTION Chances are fairly good that you do not have control over what is or is not placed in folders elsewhere on the network, so trusting network locations can present a serious security risk!

NOTE You can trust locations on a USB thumb drive, but not at the root level. Instead, you must trust a folder on the USB drive. While this can be very handy if you like to carry code from one location to another, it also exposes you to the risk that someone else will have the same folder hierarchy that you have trusted. If that happens, their folder will also be trusted and they will be able to run code unguarded on your system.

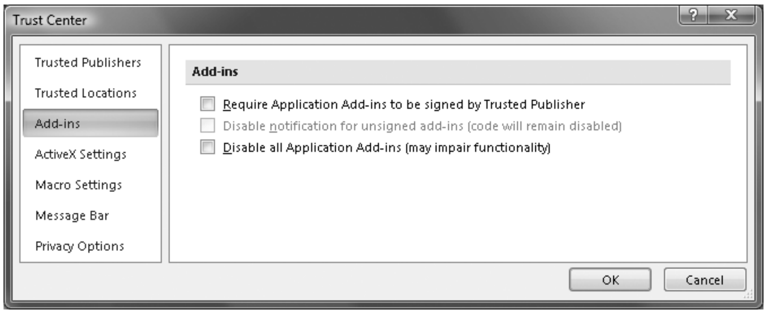
Disabling Trusted Locations

If you are a systems administrator or a user who is concerned about malicious code, then this setting is for you. You can check the box to "Disable all Trusted Locations. Only files signed by Trusted Publishers will be trusted" and stop any project dead in its tracks unless it has a digital signature. (Experiences will vary depending on the settings on the Macro Settings and ActiveX tabs.)

Add-ins

The security settings on the Add-ins tab are specifically designed for treatment of the Add-in file formats that you learned to build in Chapter 16 (including Word's global templates). Unlike the Office 2003 model, which required you to check a box in the security settings in order to trust all installed add-ins and templates, in Office 2007 these files are trusted by default. After all, it typically takes an intentional act to incorporate an add-in. However, there are a few options that allow you to override this setting, as shown in Figure 17-5.

In looking at this pane, it appears that you only have three options: all add-ins will run (the default), only add-ins signed by a trusted publisher will run, or no add-ins will run. However, as you'll soon learn, there are additional settings that provide more flexibility, particularly with the first option.



Add-ins

☐ Require Application Add-ins to be signed by Trusted Publisher
☐ Disable notification for unsigned add-ins (code will remain disabled)
☐ Disable all Application Add-ins (may impair functionality)

Figure 17-5: The Add-ins tab of the Trust Center

Requiring Add-ins to Be Signed

The first checkbox on this tab will prevent any add-in from executing code unless it is signed with a trusted digital certificate. If you select this option, then each add-in is checked for a trusted signature at load time. If a trusted signature is not present, then the add-in is not loaded, and a notification is displayed to the user.

NOTE The user may still enable the unsigned add-in(s) based on your choice for the next setting.

A word of caution should be issued here as well: Excel, in particular, gets a lot of its additional functionality through add-ins, including the Analysis Toolpak add-in, which is (finally) installed by default. This brings up a very important point: The term "add-in" includes tools that you create, as well as those created by Microsoft and third parties. Note also that even Microsoft is not a trusted publisher by default, so if this checkbox is set, users will be prompted with the message shown in Figure 17-6 upon restarting Excel. Of course, this add-in file is perfectly safe, but the message could be rather disconcerting and irritating to end users!

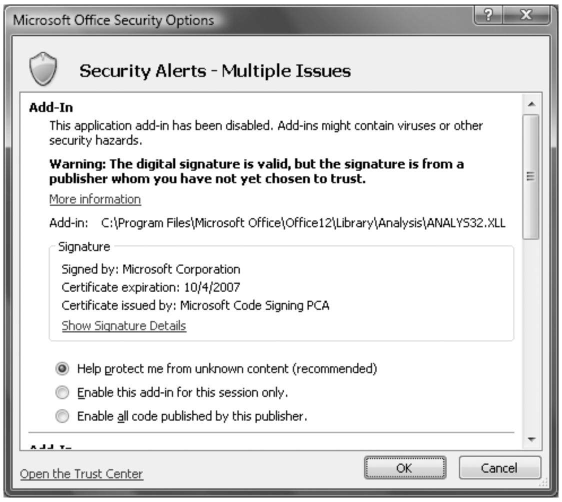
Naturally, checking "Enable all code published by this publisher" will trust Microsoft permanently and avoid this issue in the future.

The message for add-ins with no digital signature is quite similar. It includes the first two options shown in Figure 17-6, but does not offer the option to enable all code published by the publisher. The reason for this should be obvious, as no publisher is associated with the file.

Disabling Notification for Unsigned Add-ins

The "Disable notification for unsigned add-ins" checkbox only becomes active if the Trust Center is set to require add-ins to have signatures. By activating this checkbox, users will not receive a notification when an unsigned add-in is stopped.

That means users will not even receive a notification to give them the option of trusting an unsigned add-in. However, add-ins that are digitally signed but have not yet been trusted on the system will still generate a security notification.


Microsoft Office Security Options

Security Alerts - Multiple Issues

Add-In

This application add-in has been disabled. Add-ins might contain viruses or other security hazards.

Warning: The digital signature is valid, but the signature is from a publisher whom you have not yet chosen to trust.

More information

Add-in: C:\Program Files\Microsoft Office\Office12\Library\Analysis\ANALYS32.XLL

Signature

Signed by: Microsoft Corporation
Certificate expiration: 10/4/2007
Certificate issued by: Microsoft Code Signing PCA
Show Signature Details

[✓ Help protect me from unknown content (recommended)]
Enable this add-in for this session only.
Enable all code published by this publisher.

Figure 17-6: Security settings triggered by allowing only signed add-ins

Disabling All Add-ins

The "Disable all Application Add-ins" setting rightfully carries a warning that you may lose functionality. This setting should only be used in the tightest of security environments.

NOTE As a very interesting point, note that while setting this checkbox will disable all the VBA code in an add-in from running, the add-in still seems to load. This is demonstrated by the fact that disabling all add-ins does *not* block any RibbonX customizations that exist in those files. In other words, your add-in will still load and create new tabs and groups, and it will still move icons around. In addition, if all of your customizations were based on built-in controls alone, they would still function as designed.

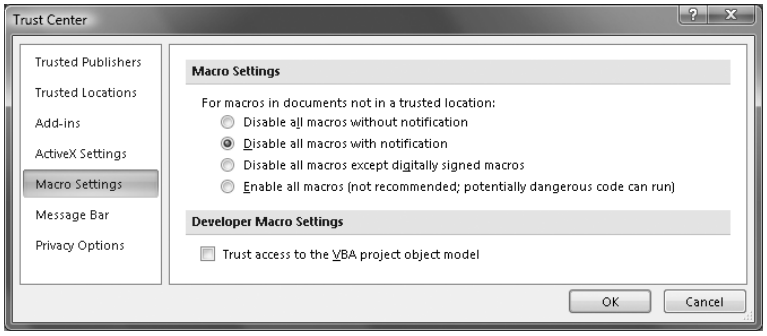
ActiveX Settings

This section of the Trust Center (not available in Access) is also new in Office 2007. It establishes how ActiveX controls will be treated from a security standpoint. As ActiveX controls are outside the scope of this book, these settings are not explored. However, it might be reassuring to know that the settings contain similar notations and guidance about the options and their effects.

NOTE For those who have never used ActiveX controls, these are the controls that are added to documents and workbooks. They are found on the Developer tab.

Macro Settings

The macro security settings are slightly different from those offered in prior versions of Office. This was to align them with the new Trusted Locations concept. The available settings are shown in Figure 17-7.



Macro Settings

For macros in documents not in a trusted location:

- ○ Disable all macros without notification
- ● Disable all macros with notification
- ○ Disable all macros except digitally signed macros
- ○ Enable all macros (not recommended; potentially dangerous code can run)

Developer Macro Settings

- ☐ Trust access to the VBA project object model

Figure 17-7: Macro settings in Office 2007

As you will immediately notice, the settings in the first section of this window (under the heading Macro Settings) are only applicable to files that are not in trusted folders. For reference, trusted folders run with the effective permission of "Enable all macros."

You'll also notice that the option most relevant to developers conveniently stands out with its own heading. That's because it is critical to trust the VBA project object model if you plan to use VBA to manipulate VBE components or code. The following sections explain these setting options.

Setting Macro Options

While the most secure setting is to automatically disable all macros without even notifying the user, this obviously obliterates the benefits of VBA code. As a person who would read this book, it is highly unlikely that you would want to do that.

The default setting for macro security is to disable all macros with notification. This is probably the nicest way to strike a balance between security and functionality, as users have the option to enable macros if they'd like.

The option to "Disable all macros except digitally signed" does exactly that, without any notification. If this setting is chosen, users do not have the option to enable the macro content for an unsigned macro.

NOTE If a file is in a trusted location, the preceding setting is ignored when the file is opened. Therefore, it opens with macros enabled and without prompting, as we would expect with any file in a trusted location.

Of course, “Enable all macros” is the most wide open setting, as it allows everything and anything to run. This setting is rarely recommended. A much better approach is to place applicable files in a trusted location, rather than effectively using a blanket switch to turn security off.

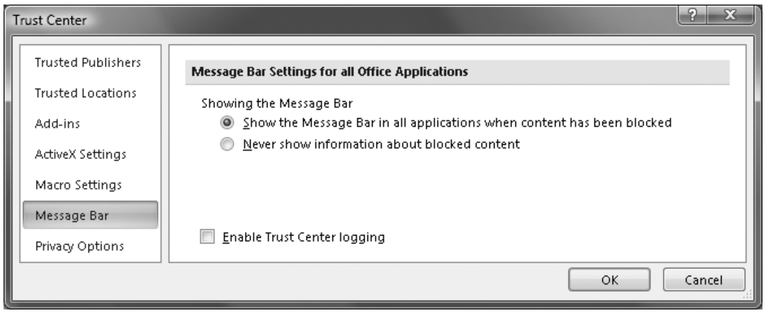
Trusting VBA Project Access

The last setting on this pane enables you to trust access to the VBA Project object model. In short, this setting allows you to successfully run code that can manipulate the VBA project components and structure, instead of just targeting the layer of the file that is seen in the user interface. This type of access allows code that can actually write, modify, or delete code, meaning that the code itself could even add or remove entire code modules.

Based on this simple explanation alone, you can see the power that someone would have if you allow and trust access to the VBA project. As a general rule, you would most likely never want to set this on a user’s workstation. This is a global setting and it affects files both inside and outside the trusted location folders.

Message Bar

The message bar settings, shown in Figure 17-8, control how the application reacts when macro content has been disabled.



Message Bar Settings for all Office Applications

Showing the Message Bar

- [✓ Show the Message Bar in all applications when content has been blocked]
- Never show information about blocked content

☐ Enable Trust Center logging

Figure 17-8: Message bar settings

By default, the message bar is shown when content is blocked, and appears as shown in Figure 17-9. Changing the setting to “Never show . . .” will, of course, stop the system from prompting the user when macro content has been disabled.

Removing this message also removes the convenient option that allows the user to enable macros associated with the file.

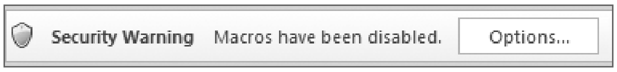

Security Warning Macros have been disabled.

Figure 17-9: Message bar warning of disabled content

NOTE The message bar is used to display warnings about what could be malicious data connections, as well as code. These warnings can also be triggered in Word when using the mail-merge feature.

Privacy Options

As we are primarily concerned with exploring security, the privacy options are not covered in this book. This tab provides relatively clear options and guidance to help users customize their settings.

Digital Certificates

As mentioned at the outset of this chapter, digital certificates are an integral part of deploying a macro-enabled solution. In the past, digital certification used to be the only real security method for trusting files, but now this process is most applicable to files that lie outside the Trust Center's trusted folders.

While you may be able to avoid the digital certificate process if you are developing files only for your own use, their value quickly becomes apparent when you deploy applications to other users. Just imagine having to tell each user to set up the appropriate trusted folders for your files! Not only does this expose the client to unacceptable risk (as someone could dump a malicious file in there as well), but it appears unprofessional. Using a digital certificate can avoid these types of issues.

How Digital Certificates Work

To keep things simple, let's run through a scenario. Consider a developer who wants to secure his code. He goes to the "locksmith" and asks for two keys.

The first key, known as the *private key,* is a master key and is unique. The developer knows that the locksmith will never issue the same private key to anyone else, and that the locksmith has recorded his name next to it in a log book. Using this key will lock the code in the file and let everyone know that he wrote and secured it.

The second key, known as the *public key,* is almost identical to the first, but has a few less teeth. This key will unlock the code in the file so that the code is readable but not editable. Unlike the private key, which the locksmith marked with a "Do Not Copy" stamp, this key comes in sets that the developer will freely hand out to anyone who wants to use his work.

The developer takes his private key and turns it in the lock, thereby setting the security on the code in his application. It is this process that is known as digitally signing a file. He then sends the file out to his clients, along with a copy of the public key.

When the digitally signed file is opened by the client, the system recognizes that there is a lock on the file, and matches it with the appropriate key. The system looks up the developer in its list of trustworthy developers, and not finding him there, takes the actions dictated by the Macro Security Settings tab of the Trust Center. Assuming that the settings specify prompting the user, the system then asks the user whether they would like to run the code, and then offers the option to trust the developer permanently.

Assuming that the user agrees to trust the developer, the system adds his name to the list of trusted developers, uses the key to unlock the code, and puts the key in safekeeping until the next time. Thereafter, any file that is signed with the identical signature will be unlocked with the stored public key and the code will be allowed to execute. Keep in mind that several things had to happen to make this possible, including the user specifically stating that they trust the developer. It also requires that the code is still digitally signed. We're about to discuss some nuances to that.

NOTE While the digital certificate proves who signed the code, it makes absolutely no guarantee that the code within the file is safe. It is completely up to users if they want to trust the issuer of the certificate on their system.

Where the digital certificate proves its worth is in the mandate that the private key must sign the code. Any modifications made to code in a digitally signed file forces a validation of the digital certificate. Failing to get validated, the code in the file will no longer have a digital certificate.

Assume that in the preceding case, one of the developer's clients decided to modify the code. As they begin to make modifications, the lock pops open completely. Unlike most padlocks, however, which can be relocked simply by closing the hasp, this lock requires the private key to relock the file. Since the private key resides on the developer's computer, and not on the client's, the file cannot be relocked and the digital certificate evaporates. Unfortunately for the client, the developer's locksmith will not issue him a copy of the private key, so he is left with an unsigned file. Any attempts to open the file in future will then be treated as unsigned code and reacted to with the security permissions set in the Trust Center's Macro Settings area.

As you can see, the digital certificate offers assurance to both the developer and the client. The client can be assured that the code was delivered as intended by the developer, and the developer can be assured that the client has not changed the code in any way (or, if the code has been changed, it no longer carries the developer's digital certificate). This can be quite useful for the developer from a liability standpoint if any destruction is caused and blamed on his file. If the digital certificate is gone, it indicates someone has tampered with the code.

NOTE While the preceding anecdote explains how a digital certificate works, a far more technical explanation can be found on Microsoft's Technet site at the following URL: `www.microsoft.com/technet/security/guidance/cryptographyetc/certs.mspx`.

Acquiring a Digital Certificate

While digital code signing certificates can be purchased from a third-party vendor, they can also be created without cost using a self-certification program that is installed with Microsoft Office: `SELFCERT.exe`.

At first glance, you might think it is a no-brainer to simply use `SELFCERT.exe` to create a free certificate, rather than pay for a commercial version. As with most things, however, each approach has both benefits and drawbacks, the biggest of which are cost, verification, and acceptance.

As mentioned, the self-certification route carries a significant cost advantage over purchasing a digital certificate from a third-party issuer. Commercially purchased certificates can range in cost from $200–$500 on a yearly basis, depending upon the vendor that you choose.

Conversely, anyone can generate a certificate with the self-certification utility, so what does it really tell you about the developer? The signature could be published under any name, real or fictitious, and while it can't necessarily be forged, it could certainly be masquerading under someone else's name.

Part of what you pay for when you get a certificate from a commercial outfit is third-party entity identification. The cost of the certificate helps cover the staff time and resource costs for verifying the existence of the person or company who has requested the certificate. As the entire point of having a certificate is to add security, it only makes sense that we should also try to show that the developer actually exists!

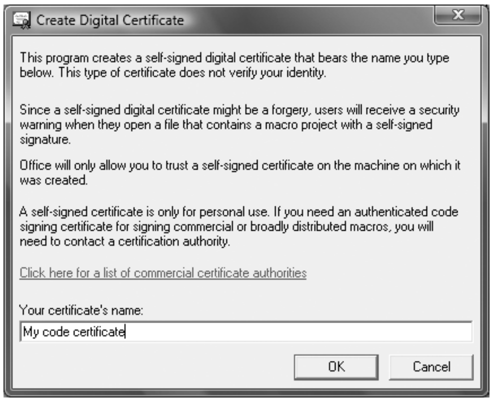
A variety of third-party certificate authorities are currently doing business on the Internet. Microsoft's MSDN website carries a list of such businesses: `http://msdn2.microsoft.com/en-us/library/ms995347.aspx`.

TIP If you are interested in purchasing a digital certificate from one of the commercial certificate authorities, make sure that you purchase a *code signing certificate,* rather than one of the other kinds.

Using SELFCERT.exe to Create a Digital Signature

Because many of you will at least want to experiment with the self-certification tool, we will now look at the process.

To launch the self-certification program, go to the Windows Menu (or Start Menu in Windows XP) and choose All Programs ➪ Microsoft Office ➪ Microsoft Office Tools ➪ Digital Certificate for VBA Projects. This will launch the program, displaying the dialog shown in Figure 17-10.


Create Digital Certificate

This program creates a self-signed digital certificate that bears the name you type below. This type of certificate does not verify your identity.

Since a self-signed digital certificate might be a forgery, users will receive a security warning when they open a file that contains a macro project with a self-signed signature.

Office will only allow you to trust a self-signed certificate on the machine on which it was created.

A self-signed certificate is only for personal use. If you need an authenticated code signing certificate for signing commercial or broadly distributed macros, you will need to contact a certification authority.

Click here for a list of commercial certificate authorities

Your certificate's name: My code certificate

Figure 17-10: Creating a digital certificate with SELFCERT.exe

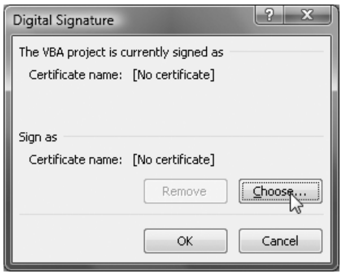
NOTE **Despite the warnings shown in Figure 17-10, you can actually trust the public key of a self-signed certificate on another computer. This is of critical importance to developers who want deploy solutions.**

To create your own digital certificate, all you need to do is enter a name and press OK. You will instantly receive notification that your certificate has been created.

Adding a Digital Certificate to a Project

After you have acquired a code signing certificate, either via self-certification or through a commercial certificate authority, you need to assign it to your project. Fortunately, this is also very easy to do.

Open your favorite macro-laden file (all applications use the same process) and enter the VBE. From the Tools menu, choose Digital Signature. You will find yourself at the screen shown in Figure 17-11.


Digital Signature

The VBA project is currently signed as

Certificate name: [No certificate]

Sign as

Certificate name: [No certificate]

Figure 17-11: Selecting a digital signature

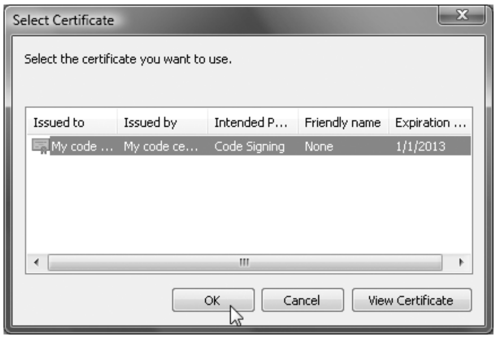
Notice that at this point, there is neither a signature associated with the project, nor a default certificate showing in the Sign As field.

NOTE **After you have signed your first project, the Sign As area will be pre-filled with the name of the last certificate you applied. Clicking OK at that point will apply the certificate in the Sign As field to your active project.**

Click the Choose button to be shown a list of the existing digital certificates that are available for signing the code (see Figure 17-12).

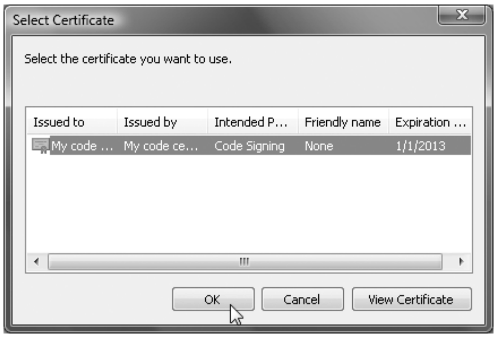
As you can see, the certificate that you just created, My Code Certificate, is on the list. With that selected, click OK, and you will be returned to the digital certificate interface. At this point, the project is signed and the certificate name is listed in the Sign As area for easy application to other projects, as shown in Figure 17-13.

Upon clicking OK and saving the file, the project is signed and ready for distribution.



Issued to	Issued by	Intended P...	Friendly name	Expiration ...
My code ...	My code ce...	Code Signing	None	1/1/2013

Figure 17-12: Selecting a digital certificate



Issued to	Issued by	Intended P...	Friendly name	Expiration ...
My code ...	My code ce...	Code Signing	None	1/1/2013

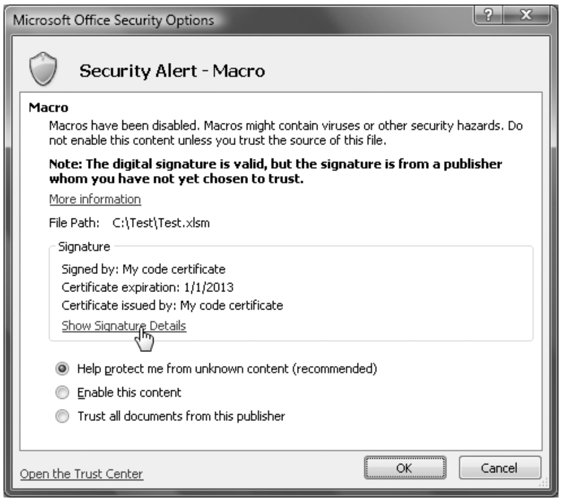
Figure 17-13: The digital certificate interface, showing the project is signed

Trusting a Digital Certificate on Another Machine

The process of trusting a digital certificate on a client's machine can seem cumbersome. Remember that this is in the end user's best interest, as it avoids unnecessary exposure to risks; and once your signature has been trusted, users will not have to repeat the process with future signed applications.

To begin the process, users must open a file that contains your digital signature. When prompted that macros are disabled, they click the Options box to be taken to the screen shown in Figure 17-14.

NOTE **If no macro warning is seen, check whether the Trust Center's Macro Settings are set to disable macros with notification. You may also want to check whether the file has been placed in a trusted location.**



Microsoft Office Security Options

Security Alert - Macro

Macro

Macros have been disabled. Macros might contain viruses or other security hazards. Do not enable this content unless you trust the source of this file.

Note: The digital signature is valid, but the signature is from a publisher whom you have not yet chosen to trust.

More information

File Path: C:\Test\Test.xlsm

Signature

Signed by: My code certificate
Certificate expiration: 1/1/2013
Certificate issued by: My code certificate
Show Signature Details

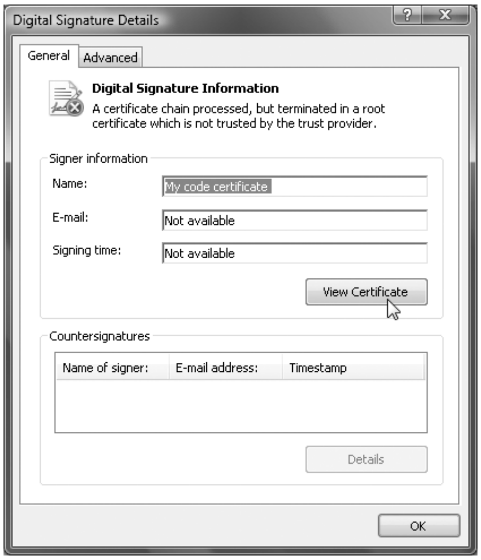
[✓ Help protect me from unknown content (recommended)]
Enable this content
Trust all documents from this publisher

Open the Trust Center

Figure 17-14: Macro security alert details

NOTE Figure 17-14 shows another difference between self-certified code and code signed by a commercial certificate. In Figure 17-6, where the code was signed by a Microsoft certificate, users had the option to trust it immediately; however, a self-signed certificate requires additional steps.

At this point, the user must click the Show Signature Details link to be taken to the details of the digital signature file, as shown in Figure 17-15.



Digital Signature Details

General | Advanced

Digital Signature Information

A certificate chain processed, but terminated in a root certificate which is not trusted by the trust provider.

Signer information

Name:	My code certificate
E-mail:	Not available
Signing time:	Not available

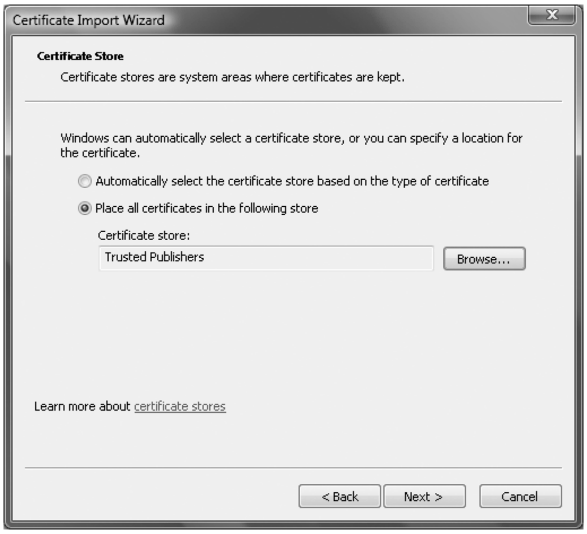
Countersignatures

Name of signer:	E-mail address:	Timestamp

Figure 17-15: Digital Signature Details

Next, they need to click View Certificate, and then click Install Certificate. This will launch the Certificate Import Wizard.

At the Certificate Import Wizard's welcome screen, the user must click Next to be taken to the Certificate Store page. After selecting the "Place all certificates in the following store" radio button, the user would click Browse, and select Trusted Publishers from the list. The Certificate Store options would then appear, as shown in Figure 17-16.



Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store: Trusted Publishers

Learn more about certificate stores

Figure 17-16: Certificate Store settings

The user then clicks Next, and then Finish, to complete the process of importing the digital certificate, finally clicking OK when notified of success. They need to keep clicking OK until they are returned to the original Macro Security warning message.

Finally, the user should click the "Enable this content" button and again click OK to finish loading the file.

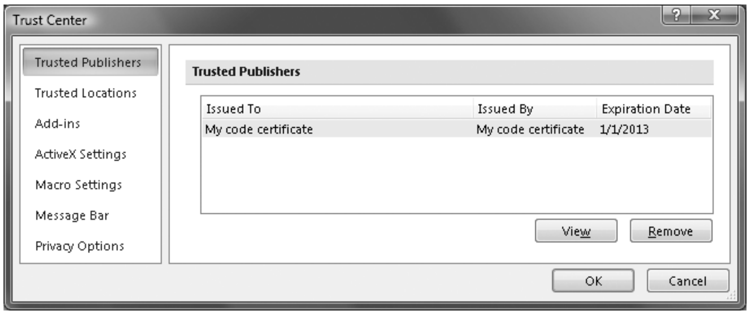
This may seem like a lot of work, but try closing and reopening the file. Not a single warning in sight! Even better, if you sign another file and send it to this same PC, it will open like a breeze; the user won't need to repeat this lengthy process.

As a final comment on certificate installation, it is also worth noting that the trusted certificate now shows on the Trusted Publishers tab of the Trust Center, as shown in Figure 17-17.

Therefore, just by self-certifying, you can have your certificate listed with the likes of Microsoft, Adobe, and other major entities.

Deleting a Digital Certificate from Your Machine

While it is fairly straightforward to remove a digital certificate from the Trusted Publishers store (highlight it and click Remove in the Trust Center) or to remove a digital signature from a project (click Remove in the screen where you would apply it), what would you do if you actually wanted to delete the private key for some reason, effectively destroying the certificate?

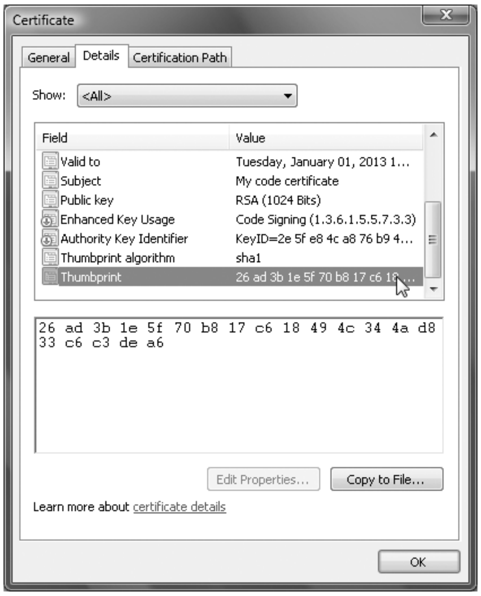


Trusted Publishers

Issued To	Issued By	Expiration Date
My code certificate	My code certificate	1/1/2013

Figure 17-17: The digital certificate listed in the Trust Center

One way to remove the certificate and its private key is to locate the certificate file on your computer and delete it. This can be done by going into the VBE and choosing Tools ➪ Digital Signatures ➪ Choose. Select the certificate that you wish to delete and click View Certificate ➪ Details. Scroll down the list of items until you find Thumbprint, as shown in Figure 17-18.



Field	Value
Valid to	Tuesday, January 01, 2013 1...
Subject	My code certificate
Public key	RSA (1024 Bits)
Enhanced Key Usage	Code Signing (1.3.6.1.5.5.7.3.3)
Authority Key Identifier	KeyID=2e 5f e8 4c a8 76 b9 4...
Thumbprint algorithm	sha1
Thumbprint	26 ad 3b 1e 5f 70 b8 17 c6 18...

```
26 ad 3b 1e 5f 70 b8 17 c6 18 49 4c 34 4a d8
33 c6 c3 de a6
```

Figure 17-18: Locating a digital signature's thumbprint

Now, based on your version of Windows, browse your computer to locate the following folder:

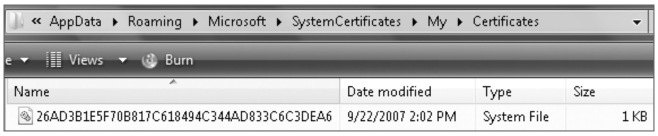
- Windows Vista:

  ```
  C:\Users\UserName\AppData\Roaming\Microsoft\SystemCertficates\My\
  Certificates
  ```

- Windows XP:

```
C:\Documents and Settings\UserName\Application
Data\Microsoft\SystemCertificates\My\Certificates
```

As shown in Figure 17-19, there is a file with a very similar name. In fact, the filename is the same as the thumbprint value except for the spaces!



« AppData ▸ Roaming ▸ Microsoft ▸ SystemCertificates ▸ My ▸ Certificates

Name	Date modified	Type	Size
26AD3B1E5F70B817C618494C344AD833C6C3DEA6	9/22/2007 2:02 PM	System File	1 KB

Figure 17-19: The digital signature file in the Certificates folder

The similarity between the file name and the thumbprint name is obviously no coincidence. This is the private key for the digital signature that you want to remove. To do so, first close all the digital signature windows in the VBE. Then, simply delete the file from the Windows Explorer window.

CAUTION **Before you remove a private key from your machine, be absolutely sure that you have the correct one, and that you really want to do this. Once the key has been deleted, *it can never be re-created,* as the keys have randomly generated hidden components. Creating a new key with the same name will not create a key identical to the one you deleted!**

In addition, keep in mind that *other* files may also rely on this key. In fact, all files that are certified by this developer (or other entity) will likely rely on that specific private key.

TIP **The ability to delete a certificate can come in handy if you need documentation to instruct end users about how to install a certificate. Simply create a (bogus) signature, apply it to a file, and then delete the certificate as explained. Upon launching the file, you will be prompted to install the unknown certificate. That way, you can go through the process and can take as many screen shots as you need!**

Conclusion

Microsoft has expended a fair amount of effort to improve the security features in Office 2007 for the benefit of both end users and developers.

For end users, splitting the file formats between macro-free versions and macro-enabled versions provides an obvious clue as to what kind of file they may be dealing with. The Trust Center also allows more security configuration options than were offered in prior versions, such as how to manage macro-laden files and ActiveX controls.

The major changes in Office 2007's security model, however, truly benefit developers. In the past, we could only avoid the nagging macro prompts by turning off our macro security settings or by digitally signing every project we started. The creation of the Trusted Folder in Office 2007 makes it easy to eliminate these annoying hassles. With the ability to designate entire directories as safe havens, developers can create and test files locally without needing to worry about the security settings.

After completing the local development and testing process, developers still have the ability to digitally sign the code before deployment. Using digital signatures, even self-signed certificates, is encouraged; and with the proper settings, end users can configure their machines to run trusted code without taking a carte blanche approach and disabling everything.