Outlook Security Manager

.NET Edition Developer's Guide

VCL Edition Developer's Guide

ActiveX Edition Developer's Guide

Last revised at **17-Dec-04**

Document version: **1.2** Product Version: **1.0**

Contents

Introduction

Terminology

In this document, on our website, and in all our text files we use the following terms.

- When we say “protected Outlook objects” we mean Outlook objects (COM interfaces), their properties or methods that being called fire security warnings.
- When we say “protected CDO objects” we mean CDO objects (interfaces), their properties or methods that being called fire security warnings.
- When we say “protected Simple MAPI calls” we mean Simple MAPI functions that being called fire security warnings.
- When we say “Outlook Security” we mean the Outlook mechanisms that prevent a developer from using Outlook objects, CDO objects or Simple MAPI functions programmatically. Microsoft started this feature with Outlook E-mail Security Update for Outlook 2000. The complete description of interoperability issues with Outlook Security you can find at http://support.microsoft.com/?kbid=264128.

What is Outlook Security Manager

Outlook Security Manager is an in-process COM object that allows a developer to avoid Outlook Security when he uses protected Outlook objects, protected CDO objects or protected Simple MAPI functions.

Outlook Security Manager makes possible to avoid Outlook Security via only three properties that disable security warnings for Outlook Object Model (OOM), Collaboration Data Objects (CDO) and Simple MAPI. Outlook Security Manager is delivered with three editions, .NET, VCL and ActiveX, each of which takes into account a platform's peculiarities.

Compatibility

Outlook Security Manager for .**NET**, **VCL** and **ActiveX** platforms directly supports C#, Managed C++, Visual Basic .NET, J#, Delphi for .NET, Delphi 5 - 7, Visual Basic 6, C++ MFC/ATL, VBA, Outlook VBScript etc.

Outlook Security Manager is fully compatible with **Outlook 2000 with E-mail Security Patch and higher**.

Examples

To aid you in learning how to use Outlook Security Manager, several examples are included in the installation package. All these examples can be found in the QDemo subfolder of the Outlook Security Manager folder.

Compatibility with Add-in Express

Outlook Security Manager is fully compatible with Add-in Express .NET Edition and Add-in Express VCL Edition.

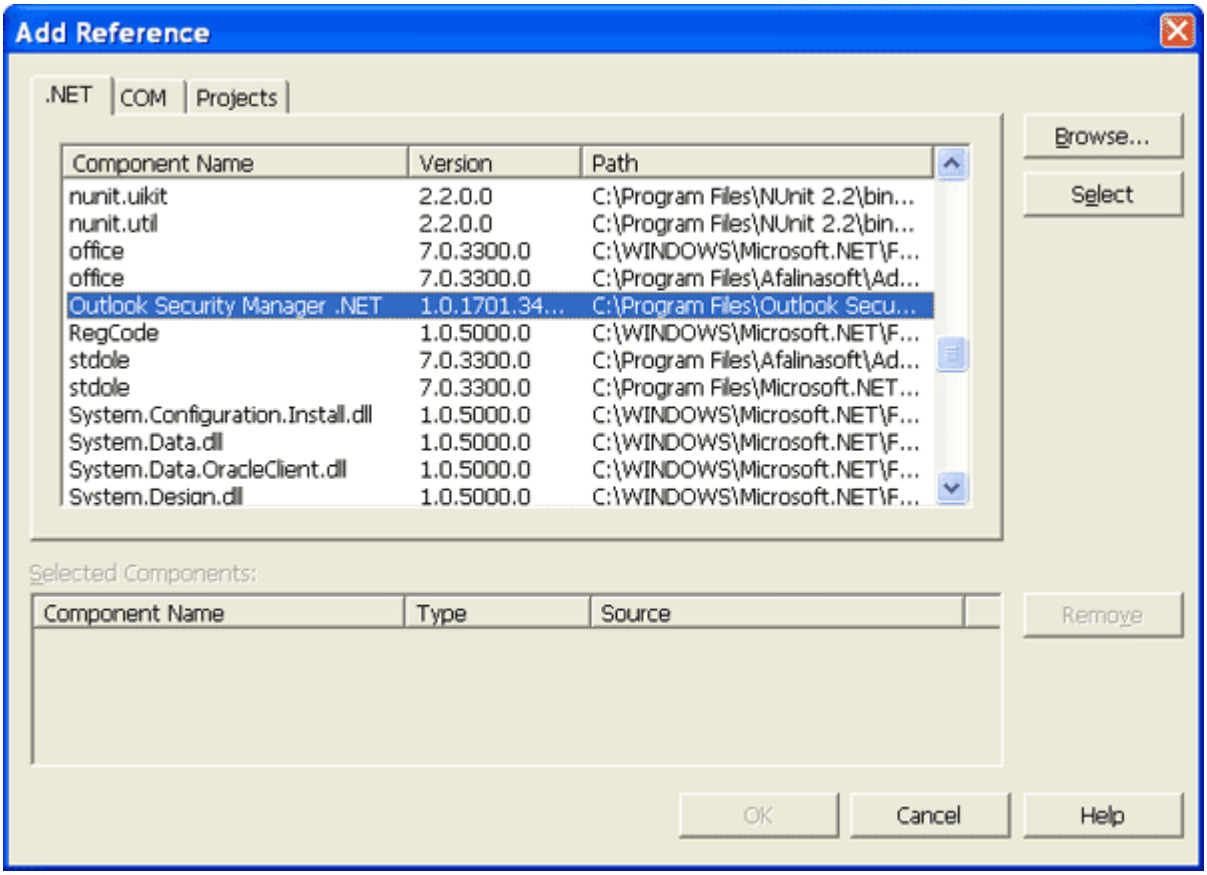
Technical Support

Please email any questions or comments to support@add-in-express.com. But first, please always check Add-in Express Web-site and Add-in Express Forums where you will find the latest news, FAQ, and the tips and tricks section.

.NET Edition Developer's Guide

The examples described below are developed in C# on Visual Studio 2003 but you can use Outlook Security Manager in all .NET compatible programming languages.

To add a reference to Outlook Security Manager into your project just use the Add Reference dialog box and select the "Outlook Security Manager .NET" assembly.



To create an instance of Outlook Security Manager you can use the following code:

```
private AddinExpress.Outlook.SecurityManager SecurityManager;
this.SecurityManager = new AddinExpress.Outlook.SecurityManager();
```

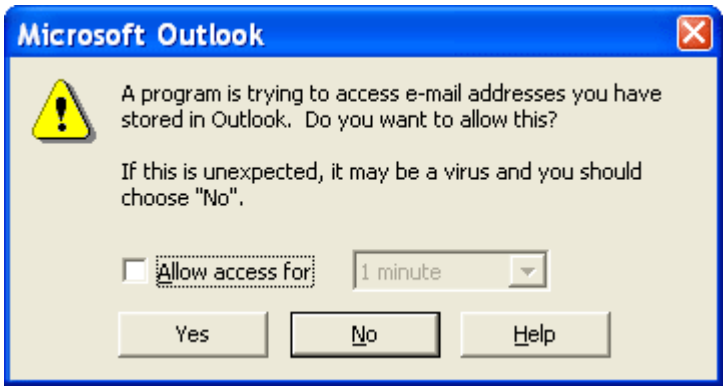
Example #1 – Outlook Add-ins

The first example is an Outlook COM add-in based on Add-in Express. You can find it in the QDemo\COMAddIn subfolder of the "Outlook Security Manager .NET" folder. Please note all the examples use Outlook 2000 PIA.

Microsoft assumes that COM add-ins can be trusted and this completely solves the security problem. Aha! We have tried many times to make sure this is true; but trusted add-ins work in Outlook 2003 only. That is why this example will be useful for you if you develop add-ins that support Outlook 2000 and 2002 as well as Outlook 2003. Please note that all described below is true for Outlook 2000 and 2002.

This add-in adds to the Explorer window a new command bar with three buttons. The first button is a switcher that disables or enables Outlook Security, the second button shows Outlook contacts, and the third shows the details of a selected message.

When Outlook Security is enabled (by default) you get a security message when click the contact or details buttons. If you disable Outlook Security the alert message doesn't appear.



How this works. Open the add-in solution in Visual Studio IDE and then open the add-in module. You can find the SecurityManager component, an instance of the OlSecurityManager class. In the DoEnumContatcs and DoGetInfo methods of the add-in module you can see the following lines:

```
if (btnOnOff.State == AddinExpress.MSO.ADXMsoButtonState.adxMsoButtonDown)
  SecurityManager.DisableOOMWarnings = true;
try
{
  …some actions with protected Outlook objects…
}
finally
{
if (btnOnOff.State == AddinExpress.MSO.ADXMsoButtonState.adxMsoButtonDown)
  SecurityManager.DisableOOMWarnings = false;
}
```

The first "if" sentence disables Outlook Security if the security button is down. The "finally" section enables Outlook Security. Between these lines there is a code that uses protected Outlook objects.

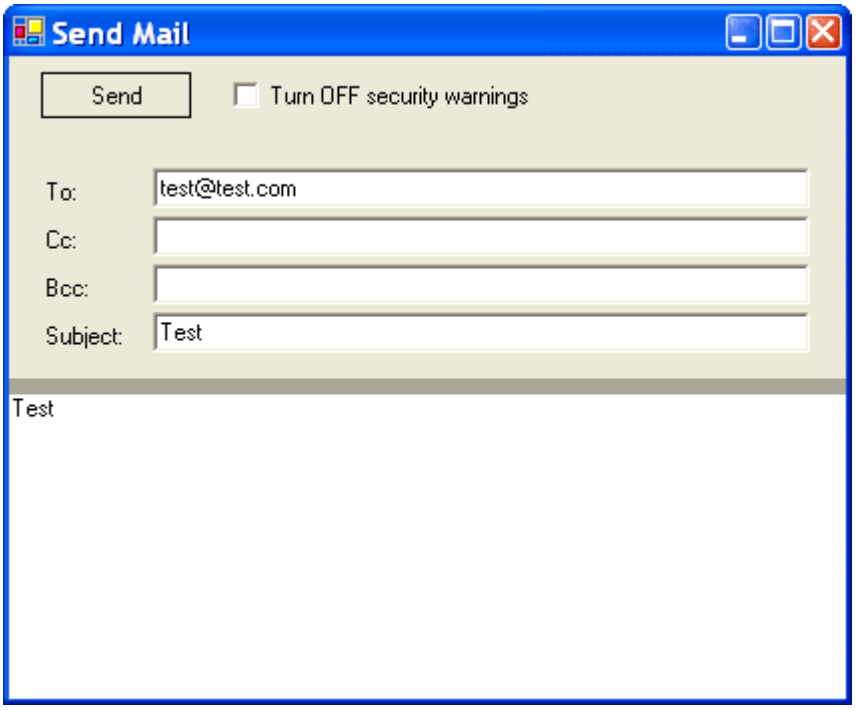
That’s all. Just remember that the DisableOOMWarnings property of the TOlSecurityManager class disables Outlook Security when you use protected objects from Outlook Object Model (OOM) and if you set this property to True.

Also remember that you should obligatory turn on Outlook Security inside the “finally” section. You should be aware that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a protected finalization code after each elementary call of the protected objects as we showed in the example above.

We have shown above how to disable Outlook Security when we use protected Outlook objects. But you can disable Outlook Security when you use protected CDO objects or protected Simple MAPI functions. Just use other properties of TOlSecurityManager, namely DisableCDOWarnings and DisableSMAPIWarnings.

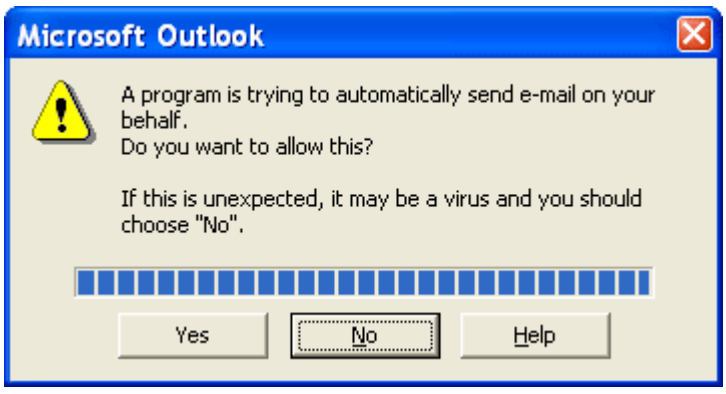
We will not comment on the whole code of this add-in and give you an opportunity to examine it yourself.

Example #2 – Automate Outlook



The second example you can find in the QDemo\SendMail subfolder of the "Outlook Security Manager .NET" folder. This example shows how to use Outlook Security Manager from applications that automate Outlook. With this application you can send messages using Outlook as a "mail engine". In this project there is one form, a small copy of the Outlook Mail window, with an Outlook Security switcher.

If you check this check box you disable Outlook Security and this application can send a message. Otherwise, will not be able to send it and get this warning:



As well as in the first example protected Outlook objects are used here. To disable Outlook Security we use the DisableOOMWarnings property. But there is one peculiarity. To use Outlook Security

Manager inside an application interacting with Outlook you should connect it to Outlook.Application used by this application.

To do this you can use the ConnectTo method of the TOlSecurityManager class. For example (this code you can find in the form):

```
var
  Outlook: Outlook2000.TOutlookApplication;
begin
  if (chkOnOff.Checked)
  {
    SecurityManager.ConnectTo(outlookApp);
    SecurityManager.DisableOOMWarnings = true;
  }
  try
  {
    …some actions with protected Outlook objects…
  }
  finally
  {
    SecurityManager.DisableOOMWarnings = false;
  }
```

Thus, if you want to disable Outlook security use Outlook Security Manager and remember to call the ConnectTo method before disabling Outlook Security.

We remind you once again. Please remember that you should obligatory turn on Outlook Security inside the “finally” section. You should be aware that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a protected finalization code after each elementary call of the protected objects as we showed in the example above.

Deployment

To deploy your software that uses Outlook Security Manager you should include the secman.dll file to your setup package and register it on an end-user computer as a COM object. To register it you can use the regsvr32 utility or special options of your installer (vsdrfCOMSelfReg for example).

Please pay attention that you should place secman.dll as a **shared DLL** into the shared folder of Windows, Common Files \ Outlook Security Manager. Please do not unregister secman.dll if it existed before you install your product. The complete information about deploying shared files you can find on MSDN website: Windows 2000 Application Specifications: Component Sharing

You can find the redistributable version of secman.dll in the Redistributable subfolder of Outlook Security Manager folder.

Attention!

You should understand that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a finalization code after each elementary call of the protected objects as we showed in the examples above.

The OlSecurityManager Class

The TOlSecurityManager class is a wrapper over Outlook Security Manager COM object. Its properties allow a developer to disable / enable Outlook security warnings for Outlook objects interoperability, CDO and Simple MAPI calls.

Namespace: **AddInExpress.Outlook**

DisableCDOWarnings

Disables / enables security warnings when using protected CDO objects.

```
[Visual Basic]
  Public Property DisableCDOWarnings As Boolean

[C#]
  public bool DisableCDOWarnings {get; set;}

[C++]
  public:
    __property bool get_DisableCDOWarnings ()
    __property void set_DisableCDOWarnings ( bool )
```

Set this property to True to disable Outlook security warnings when you call protected CDO objects.

DisableOOMWarnings

Disables / enables the security warnings when using protected Outlook objects.

```
[Visual Basic]
  Public Property DisableOOMWarnings As Boolean

[C#]
  public bool DisableOOMWarnings {get; set;}

[C++]
```

```
  public:
    __property bool get_DisableOOMWarnings ()
    __property void set_DisableOOMWarnings ( bool )
```

Set this property to True to disable Outlook security warnings when you call protected Outlook objects.

DisableSMAPIWarnings

Disables / enables the security warnings when using Simple MAPI functions.

```
[Visual Basic]
  Public Property DisableSMAPIWarnings As Boolean

[C#]
  public bool DisableSMAPIWarnings {get; set;}

[C++]
  public:
    __property bool get_DisableSMAPIWarnings ()
    __property void set_DisableSAMPIWarnings ( bool )
```

Set this property to True to disable Outlook security warnings when you call Simple MAPI functions.

Check

Checks the possibility to disable Outlook security warnings.

```
[Visual Basic]
Public Function Check(ByVal kind As OutlookSecurity.osmWarningKind) _
  As OutlookSecurity.osmResult

Public Enum osmResult As Integer
Public Enum osmWarningKind As Integer

[C#]
public OutlookSecurity.osmResult Check ( OutlookSecurity.osmWarningKind kind )
```

```
public sealed enum osmResult : System.Enum
public sealed enum osmWarningKind : System.Enum

[C++]
public OutlookSecurity::osmResult Check ( OutlookSecurity::osmWarningKind kind )

public __value enum osmResult
public __value enum osmWarningKind
```

Determines the possibility to disable security warnings for the security kind specified by the AKind parameter. The kind parameter can be:

- **osmObjectModel** – the possibility will be determined for protected Outlook objects.
- **osmCDO** - the possibility will be determined for protected CDO objects.
- **osmSimpleMAPI** - the possibility will be determined for protected Simple MAPI function.

The Check function returns one of the following values:

- **osmOK** – Outlook Security can be disabled.
- **osmDLLNotLoaded** – Outlook Security cannot be disabled because Outlook Security Manager is not registered.
- **osmSecurityGuardNotFound** – Outlook Security cannot be disabled.
- **osmUnknownOlVersion** – Outlook Security cannot be disabled because Outlook Security Manager cannot determine Outlook version.
- **osmCDONotFound** - Outlook Security cannot be disabled because CDO is not installed.

ConnectTo

Connects to an existing Outlook.Application object.

```
[Visual Basic]
  Public Sub ConnectTo(ByVal outlookApp As Object)
```

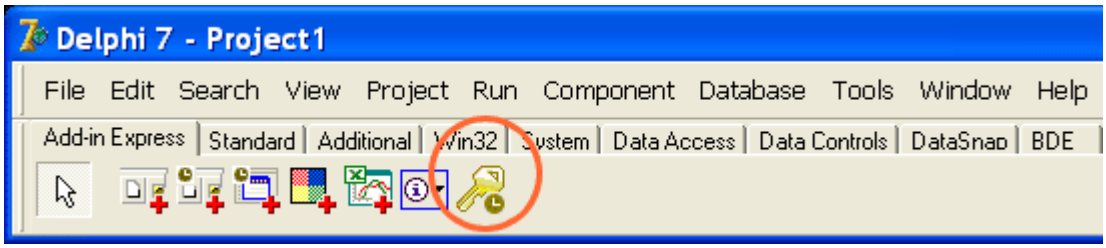
```
[C#]
  public void ConnectTo (System.Object outlookApp)

[C++]
  public void ConnectTo (System::Object outlookApp)
```

Connects to the Outlook.Application object used by the application. This method is important for applications interacting with Outlook and should be called before the DisableXXX properties are used. This method is optional for Outlook add-ins.

VCL Edition Developer's Guide

The examples described below are developed in Borland Delphi 7. Please note that to add a reference to Outlook Security Manager into your project, just place the olsecuritymanager component from the Add-in Express tab on the Component Palette to your form or another container:



Or you can use the uses clause and the Create constructor:

```
Uses OutlookSecMan;

…

SecurityManager := TOlSecurityManager.Create(nil);
```

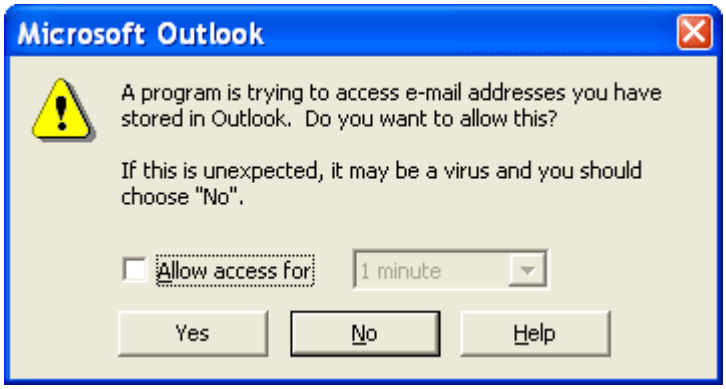
Example #1 – Outlook Add-ins

The first example is an Outlook COM add-in based on Add-in Express. You can find it in the QDemo\COMAddIn subfolder of the “Outlook Security Manager VCL” folder.

Microsoft assumes that COM add-ins can be trusted and this completely solves the security problem. Aha! We have tried many times to make sure this is true; but trusted add-ins work in Outlook 2003 only. That is why this example will be useful for you if you develop add-ins that support Outlook 2000 and 2002 as well as Outlook 2003. Please note that all described below is true for Outlook 2000 and 2002.

This add-in adds to the Explorer window a new command bar with three buttons. The first button is a switcher that disables or enables Outlook Security, the second button shows Outlook contacts, and the third shows the details of a selected message.

When Outlook Security is enabled (by default) you get a security message when click the contact or details buttons. If you disable Outlook Security the alert message doesn’t appear.



How this works. Open the add-in project in Delphi IDE and then open the add-in module. You can find the SecurityManager component, an instance of the TOlSecurityManager class. In the DoEnumContatcs and DoGetInfo methods of the add-in module you can see the following lines:


```
if CmdBar.Controls[0].AsButton.State = adxMsoButtonDown then
  SecurityManager.DisableOOMWarnings := True;
try
  …some actions with protected Outlook objects…
finally
  if CmdBar.Controls[0].AsButton.State = adxMsoButtonDown then
    SecurityManager.DisableOOMWarnings := False;
end;
```

The first “if” sentence disables Outlook Security if the security button is down. The “finally” section enables Outlook Security. Between these lines there is a code that uses protected Outlook objects.

That’s all. Just remember that the DisableOOMWarnings property of the TOlSecurityManager class disables Outlook Security when you use protected objects from Outlook Object Model (OOM) and if you set this property to True.

Also remember that you should obligatory turn on Outlook Security inside the “finally” section. You should be aware that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a protected finalization code after each elementary call of the protected objects as we showed in the example above.

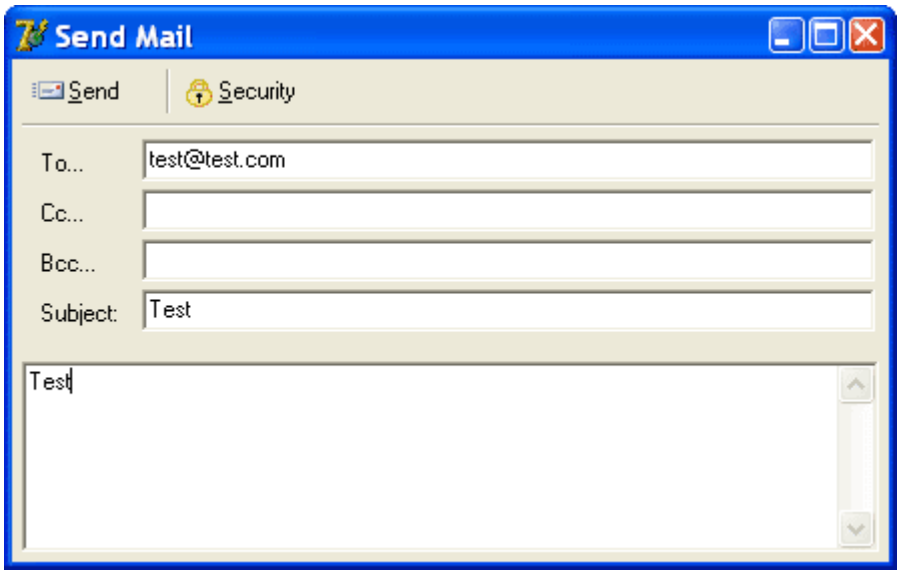
We have shown above how to disable Outlook Security when we use protected Outlook objects. But you can disable Outlook Security when you use protected CDO objects or protected Simple MAPI functions. Just use other properties of TOlSecurityManager, namely DisableCDOWarnings and DisableSMAPIWarnings.

We will not comment the whole code of this add-in and give you an opportunity to examine it yourself.

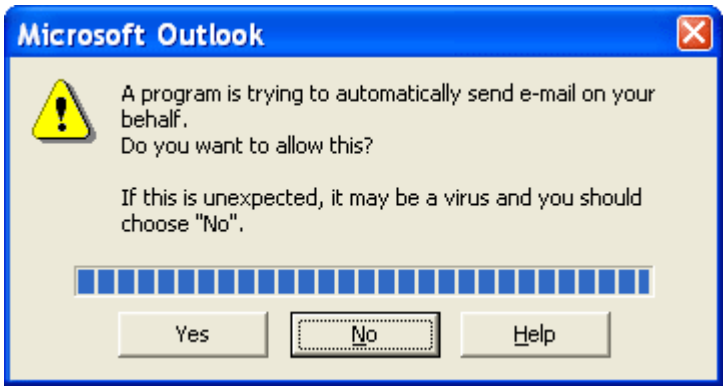
Example #2 – Automate Outlook

The second example you can find in the QDemo\SendMail subfolder of the "Outlook Security Manager VCL" folder. This example shows how to use Outlook Security Manager from applications that automate Outlook. With this application you can send messages using Outlook as a "mail engine".

In this project there is one form, a small copy of the Outlook Mail window, with an Outlook Security switcher.



If you check this button you disable Outlook Security and this application can send a message. Otherwise, will not be able to send it and get this warning:



As well as in the first example protected Outlook objects are used here. To disable Outlook Security we use the DisableOOMWarnings property. But there is one peculiarity. To use Outlook Security Manager inside an application interacting with Outlook you should connect it to Outlook.Application used by this application.

To do this you can use the ConnectTo method of the TOlSecurityManager class. For example (this code you can find in the form):

```
var
  Outlook: Outlook2000.TOutlookApplication;
begin
  Outlook := TOutlookApplication.Create(nil);
  SecurityManager.ConnectTo(Outlook.Application);
  SecurityManager.DisableOOMWarnings = True;;
  try
    …some actions with protected Outlook objects…
  finally
    SecurityManager.DisableOOMWarnings := False;
    Outlook.Free;
  end;
```

Thus, if you want to disable Outlook security use Outlook Security Manager and remember to call the ConnectTo method before disabling Outlook Security.

We remind you once again. Please remember that you should obligatory turn on Outlook Security inside the “finally” section. You should be aware that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a finalization code after each elementary call of the protected objects as we showed in the example above.

Deployment

To deploy your software that uses Outlook Security Manager you should include the secman.dll file to your setup package and register it on an end-user computer as a COM object. To register it you can use the regsvr32 utility or special options of your installer.

Please pay attention that you should place secman.dll as a **shared DLL** into the shared folder of Windows, Common Files \ Outlook Security Manager. Please do not unregister secman.dll if it existed before you install your product. The complete information about deploying shared files you can find on MSDN website: Windows 2000 Application Specifications: Component Sharing

You can find the redistributable version of secman.dll in the Redistributable subfolder of Outlook Security Manager folder.

Attention!

You should understand that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a finalization code after each elementary call of the protected objects as we showed in the examples above.

The TOlSecurityManager Class

The TOlSecurityManager class is a wrapper over Outlook Security Manager COM object. Its properties allow a developer to disable / enable Outlook security warnings for Outlook objects interoperability, CDO and Simple MAPI calls.

DisableCDOWarnings

Disables / enables security warnings when using protected CDO objects.

```
public
  property DisableCDOWarnings: boolean default False;
```

Set this property to True to disable Outlook security warnings when you call protected CDO objects.

DisableOOMWarnings

Disables / enables the security warnings when using protected Outlook objects.

```
public
  property DisableOOMWarnings: boolean default False;
```

Set this property to True to disable Outlook security warnings when you call protected Outlook objects.

DisableSMAPIWarnings

Disables / enables the security warnings when using Simple MAPI functions.

```
public
  property DisableSMAPIWarnings: boolean default False;
```

Set this property to True to disable Outlook security warnings when you call Simple MAPI functions.

Check

Checks the possibility to disable Outlook security warnings.

```
public
  function Check(const AKind: TosmWarningKind): TosmResult;

type
  TosmWarningKind = (osmObjectModel, osmCDO, osmSimpleMAPI);
  TosmResult = (osmOK, osmDLLNotLoaded, osmSecurityGuardNotFound,
    osmUnknownOlVersion, osmCDONotFound);
```

Determines the possibility to disable security warnings for the security kind specified by the AKind parameter. The AKind parameter can be:

- **osmObjectModel** – the possibility will be determined for protected Outlook objects.
- **osmCDO** - the possibility will be determined for protected CDO objects.
- **osmSimpleMAPI** - the possibility will be determined for protected Simple MAPI function.

The Check function returns one of the following values:

- **osmOK** – Outlook Security can be disabled.
- **osmDLLNotLoaded** – Outlook Security cannot be disabled because Outlook Security Manager is not registered.
- **osmSecurityGuardNotFound** – Outlook Security cannot be disabled.
- **osmUnknownOlVersion** – Outlook Security cannot be disabled because Outlook Security Manager cannot determine Outlook version.
- **osmCDONotFound** - Outlook Security cannot be disabled because CDO is not installed.

ConnectTo

Connects to an existing Outlook.Application object.

```
public
  procedure ConnectTo(OutlookApp: OleVariant);
```

Connects to the Outlook.Application object used by the application. This method is important for applications interacting with Outlook and should be called before the DisableXXX properties are used. This method is optional for Outlook add-ins.

ActiveX Edition Developer's Guide

The examples described below are developed in Visual Basic 6. Please note that to create an instance of Outlook Security Manager you can use the following code:

```
Dim SecurityMananger As Object
Set SecurityManager = CreateObject("AddInExpress.OutlookSecurityManager")
```

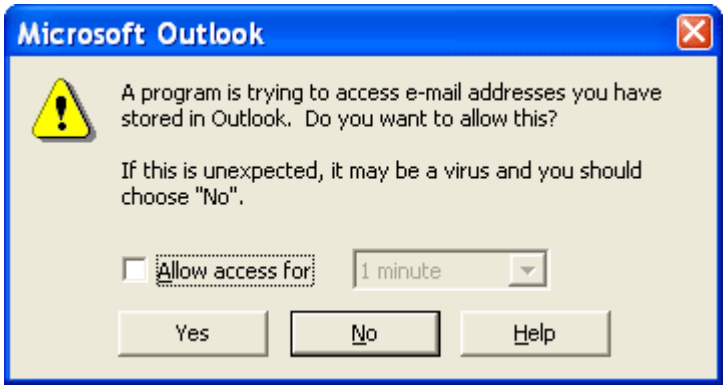
Example #1 – Outlook Add-ins

The first example is an Outlook COM add-in developed in Visual Basic 6 and based on the standard add-in approach. You can find it in the QDemo\COMAddIn subfolder of the "Outlook Security Manager ActiveX" folder.

Microsoft assumes that COM add-ins can be trusted and this completely solves the security problem. Aha! We have tried many times to make sure this is true; but trusted add-ins work in Outlook 2003 only. That is why this example will be useful for you if you develop add-ins that support Outlook 2000 and 2002 as well as Outlook 2003. Please note that all described below is true for Outlook 2000 and 2002.

This add-in adds to the Explorer window a new command bar with three buttons. The first button is a switcher that disables or enables Outlook Security, the second button shows Outlook contacts, and the third shows the details of a selected message.

When Outlook Security is enabled (by default) you get a security message when click the contact or details buttons. If you disable Outlook Security the alert message doesn't appear.



How this works. Open the add-in project in VB IDE and then open the Connect module. You can find the SecurityManager variable, an instance of the AddInExpress.OutlookSecurityManager COM object. In the Click event handlers you can see the following lines:

```
SecurityManager.DisableOOMWarnings = True
On Error Goto Finally

…some actions with protected Outlook objects…

Finally:
 SecurityManager.DisableOOMWarnings = False
```

The first "if" sentence disables Outlook Security if the security button is down. The "finally" section enables Outlook Security. Between these lines there is a code that uses protected Outlook objects.

That’s all. Just remember that the DisableOOMWarnings property of the OutlookSecurityManager object disables Outlook Security when you use protected objects from Outlook Object Model (OOM) and if you set this property to True.

Also remember that you should obligatory turn on Outlook Security inside the “finally” section. You should be aware that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a finalization code after each elementary call of the protected objects as we showed in the example above.

We have shown above how to disable Outlook Security when we use protected Outlook objects. But you can disable Outlook Security when you use protected CDO objects or protected Simple MAPI functions. Just use other properties of TOlSecurityManager, namely DisableCDOWarnings and DisableSMAPIWarnings.

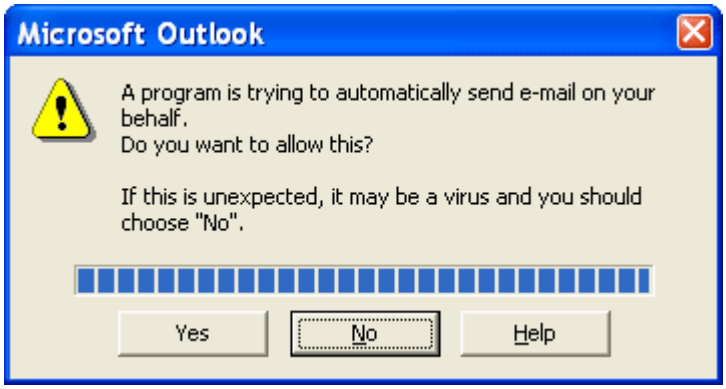
We will not comment the whole code of this add-in and give you an opportunity to examine it yourself.

Example #2 – Automate Outlook

The second example you can find in the QDemo\SendMail subfolder of the “Outlook Security Manager ActiveX“ folder. This example shows how to use Outlook Security Manager from applications that automate Outlook. With this application you can send messages using Outlook as a “mail engine”.

In this project there is one form, a small copy of the Outlook Mail window, with an Outlook Security switcher.

If you select the OFF radio button you disable Outlook Security and this application can send a message. Otherwise, will not be able to send it and get this warning:



As well as in the first example protected Outlook objects are used here. To disable Outlook Security we use the DisableOOMWarnings property. But there is one peculiarity. To use Outlook Security Manager inside an application interacting with Outlook you should connect it to Outlook.Application used by this application.

To do this you can use the ConnectTo method of the OutlookSecurityManager object. For example (this code you can find in the form):

```
Dim Outlook As Object

Set Outlook = CreateObject("Outlook.Application")

SecurityManager.ConnectTo Outlook
SecurityManager.DisableOOMWarnings = True
On Error Goto Finally

…some actions with protected Outlook objects…

Finally:
 SecurityManager.DisableOOMWarnings = False
 Set Outlook = Nothing
```

Thus, if you want to disable Outlook security use Outlook Security Manager and remember to call the ConnectTo method before disabling Outlook Security.

We remind you once again. Please remember that you should obligatory turn on Outlook Security inside the “finally” section. You should be aware that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a finalization code after each elementary call of the protected objects as we showed in the example above.

Deployment

To deploy your software that uses Outlook Security Manager you should include the secman.dll and osmax.ocx files to your setup package and register them on an end-user computer as a COM objects. To register them you can use the regsvr32 utility or special options of your installer.

Please pay attention that you should place secman.dll and osmax.ocx as **shared DLLs** into the shared folder of Windows, Common Files \ Outlook Security Manager. Please do not unregister them if they existed before you install your product. The complete information about deploying shared files you can find on MSDN website: Windows 2000 Application Specifications: Component Sharing

You can find the redistributable version of secman.dll in the Redistributable subfolder of the Outlook Security Manager folder.

Attention!

You should understand that Outlook Security Manager can potentially open a door for unsafe content. To avoid this you must enable Outlook security within a finalization code after each elementary call of the protected objects as we showed in the examples above.

The TOlSecurityManager Object

The TOlSecurityManager class is a wrapper over Outlook Security Manager COM object. Its properties allow a developer to disable / enable Outlook security warnings for Outlook objects interoperability, CDO and Simple MAPI calls.

DisableCDOWarnings

Disables / enables security warnings when using protected CDO objects.

```
public property DisableCDOWarnings As Boolean
```

Set this property to True to disable Outlook security warnings when you call protected CDO objects.

DisableOOMWarnings

Disables / enables the security warnings when using protected Outlook objects.

```
public property DisableOOMWarnings As Boolean
```

Set this property to True to disable Outlook security warnings when you call protected Outlook objects.

DisableSMAPIWarnings

Disables / enables the security warnings when using Simple MAPI functions.

```
public property DisableSMAPIWarnings As Boolean
```

Set this property to True to disable Outlook security warnings when you call Simple MAPI functions.

Check

Checks the possibility to disable Outlook security warnings.

```
public function Check(AKind As Integer) As Integer
```

Determines the possibility to disable security warnings for the security kind specified by the AKind parameter. The AKind parameter can be:

- **osmObjectModel** – the possibility will be determined for protected Outlook objects.
- **osmCDO** - the possibility will be determined for protected CDO objects.
- **osmSimpleMAPI** - the possibility will be determined for protected Simple MAPI function.

The Check function returns one of the following values:

- **osmOK** – Outlook Security can be disabled.
- **osmDLLNotLoaded** – Outlook Security cannot be disabled because Outlook Security Manager is not registered.
- **osmSecurityGuardNotFound** – Outlook Security cannot be disabled.
- **osmUnknownOlVersion** – Outlook Security cannot be disabled because Outlook Security Manager cannot determine Outlook version.
- **osmCDONotFound** - Outlook Security cannot be disabled because CDO is not installed.

ConnectTo

Connects to an existing Outlook.Application object.

```
public sub ConnectTo(OutlookApp As Object)
```

Connects to the Outlook.Application object used by the application. This method is important for applications interacting with Outlook and should be called before the DisableXXX properties are used. This method is optional for Outlook add-ins.