

NEW!

CramSessionComprehensive **Study Guides**

A+
Adobe
C++
Cisco CCNA

**Your Trusted
Study Resource
for
Technical
Certifications**

Written by experts.
The most popular
study guides
on the web.

In Versatile
PDF file format

Check out these great features
at www.cramsession.com

> **Discussion Boards**

<http://boards.cramsession.com>

> **Info Center**

<http://infocenter.cramsession.com>

> **SkillDrill**

<http://www.skilldrill.com>

> **Newsletters**

<http://newsletters.cramsession.com/default.asp>

> **CramChallenge Questions**

<http://newsletters.cramsession.com/signup/default.asp#cramchallenge>

> **Discounts & Freebies**

<http://newsletters.cramsession.com/signup/ProdInfo.asp>

INFORMATION TECHNOLOGY

Outlook 2000 & Exchange Server 5.5 Solutions

Version 3.0.0

Microsoft Office
Microsoft Windows 2000
Microsoft Windows XP
Network Security
Network+
Networking
Nortel Networks
Novell
Oracle
Proxy Server
Red Hat Linux
SAIR Linux
SANS
SCO
Server+
SQL
Sun Solaris
Unix
Visual Basic
Web Design

Notice: While every precaution has been taken in the preparation of this material, neither the author nor Cramsession.com assumes any liability in the event of loss or damage directly or indirectly caused by any inaccuracies or incompleteness of the material contained in this document. The information in this document is provided and distributed "as-is", without any expressed or implied warranty. Your use of the information in this document is solely at your own risk, and Cramsession.com cannot be held liable for any damages incurred through the use of this material. The use of product names in this work is for information purposes only, and does not constitute an endorsement by, or affiliation with Cramsession.com. Product names used in this work may be registered trademarks of their manufacturers. This document is protected under US and international copyright laws and is intended for individual, personal use only.

For more details, visit our [legal page](#).



CramSession
Prepare for Success!



Outlook 2000 & Exchange Server 5.5 Solutions

Version 3.0.0

NOTICE: Got the **NEWest Version?**
Make sure by clicking here!

Abstract:

This study guide will provide you with information to prepare for the MS 70-105 exam, Designing and Implementing Collaborative Solutions with Microsoft Outlook 2000 and Microsoft Exchange Server 5.5. Exam topics include Using Forms Technology, Web Technology, COM add-ins, MTS and VBA, Event Scripting and Routing Objects to Implement Solutions.

Find even more help here:

- > **Feedback & Discussion Board for this exam**
- > Read & Write Reviews of this study guide
- > Rate this Cramsession study guide



Contents:

Outlook Workgroup Solutions..... 4

View based Solution 5

Custom Outlook Form 5

Exchange Folder Based Solution 7

COM..... 7

Outlook Object Model 9

 AppointmentItem 9

 JournalItem 9

 MailItem 9

 MeetingRequestItem10

 NoteItem10

 PostItem10

 RemoteItem10

 ReportItem.....10

 TaskItem11

 TaskRequestItem11

 Other Objects11

Address Book12

Customization12

CDO Object Model13

Scripting Architecture14

Routing Architecture.....15

 Routing Engine15

 Routing Objects16

 Process Map Actions16

 Routing Wizard16

 Object Model17

 Permissions18



Outlook 2000 & Exchange Server 5.5 Solutions

CDO Rendering and Web Solutions18
 RenderingApplication Object19
 ContainerRenderer Object19
 ObjectRenderer Object.....19
Debugging20
Exchange Data Services.....20
Solution Distribution21



Outlook Workgroup Solutions

With Outlook, you can create workgroup solutions with no programming involved or create advanced forms by using custom controls, properties, and VBScript. You can also create a form using another Office program or by using custom views in a public folder. There are three basic approaches to creating Outlook solutions for you to follow:

- For instant workgroup solution without coding, use existing items, such as tasks or appointment items, put the items in a public folder, and assign a view to the folder:
 - Step 1: Create a public folder.
 - Step 2: Create an Outlook item in a public folder.
 - Step 3: Create and use a custom view in a public folder.
 - Step 4: Give permissions to those you want to use the folder.
- For extending the use of pre-built item without coding, modify an existing item by adding additional pages and fields.
 - Step 1: Open the item on which you want to base the form.
 - Step 2: Set attributes for the form in design mode.
 - Step 3: Design the form by moving, removing, and adding fields and pages in the form via the Field Chooser.
 - Step 4: Test and publish your form.
- For a very advanced application, you need to create advanced forms via the Control Toolbox and VBScript to access properties, events, methods, and objects within Outlook.
 - Step 1: Choose the Outlook item on which to base your custom form.
 - Step 2: Extend the form by using custom controls from the Control Toolbox.
 - Step 3: Store the Control's value by binding the control to an Outlook field.
 - Step 4: Use Outlook object browser to manipulate the classes, properties, methods, events, and constants available from the Outlook object library if necessary.
 - Step 5: Use Microsoft Script Debugger for testing and correcting errors in the VBScript code of your form.

You can also use Office document forms (such as Word and Excel) to create special applications. Note that Office documents can be used as forms, but you will not be able to create custom form pages.



There are different limitations associated with different implementation methods. You should visit: <http://support.microsoft.com/support/kb/articles/Q266/4/28.ASP> and find out what limitations can affect your outcomes.

View based Solution

For the fastest implementation, you can customize Microsoft Outlook by creating a custom view that suits your needs. A Custom view is created simply by selecting the fields that Outlook displays as well as entering or changing the contents of fields directly in the table view. This method, of course, does not offer the flexibility the other methods provide.

Custom Outlook Form

Before you decide to customize a form to fulfill your needs, consider the following:

- Which form most closely provides the needed functionality? Choose the one that requires the least customization.
- Which standard fields are available for the form? Choose the form whose fields most closely match you needs.
- What is the nature of your solutions? If the form will be sent, use a mail message form. If the solution is for tracking information in a folder, a post/contact/task/appointment form would be more appropriate.

When adding fields to your custom form, remember the following:

- When you drag a field from the Field Chooser, the field automatically binds to the appropriate control, which makes your job much easier.
- When you drag a control from the Control Toolbox, you must bind the control to a form if you want to save a value to or from it.
- TextBox, CheckBox, ListBox, ComboBox, and OptionButton controls are likely candidates for binding. Other relatively static controls such as Label and Image controls are generally not bound.
- Validation Formula gives you the ability to concisely validate user inputs. Simple field validation uses operators like >, <, = ...etc.
- Item-level validation requires more CPU power. Form-level validation occurs only when the user is submitting the form, which may be more appropriate.
- Although there is no precise limit to the number of fields and controls you can use on a custom form, MS recommends that custom Outlook forms have less than 300 controls or user-defined fields, otherwise the form may not function correctly.



Outlook 2000 & Exchange Server 5.5 Solutions

- For security purposes you may want to hide some fields in certain situations. For example, to hide fields when printing and saving a form, click Properties on the shortcut menu, select the Validation page and clear the Include this field for Printing and Save As check box.

You can program your custom form to display a web-based interface. To do this, when in Design mode, go into the Control Toolbox -> Additional Controls and select Microsoft Web Browser Control.

The Custom Outlook form only supports the Click event for custom controls. Since many controls are specifically designed to work with other events, they may not function in Outlook. In this case, you may want to create custom functionality with VBA and COM to interact with the form.

To create HELP for your user, the simple way is to use TextBox control. For a more sophisticated help, do the following:

- Place a CommandButton control on the form and set the caption to "Help."
- Create a CommandButton_Click Sub procedure in VBScript to run VBScript code when the user clicks the button. This code should use the CreateObject method to access your custom ActiveX component that provides help functions.

If you are going to deploy the custom form in a typical folder-based form solution, you should

1. Publish the form in the folder or forms library where you want to use it. Note that the name of the form is also the Message Class.
2. Make this new form the default form for that folder. The Outlook 2000 Forms Administrator utility, Formswap.exe, allows you to change the default form that Outlook uses to display items. To download a copy, check out <http://www.microsoft.com/office/ork/2000/journ/OutToolsIntro.htm>.
3. For any existing items in the folder that you want to use this new form, change the Message Class field in each of the items in the folder to match the form's Message Class.

Finally, to protect your form from being modified, when in design mode, click the Properties page and select the Protect Form Design. This allows you to set a password for the form.



Exchange Folder Based Solution

To build a groupware solution in a Microsoft Exchange public folder, MS recommends you:

1. Prepare the public folder with Exchange Server Administrator.
2. Decide which type of folder and/or form to customize.
3. Open a new form and customize it.
4. Publish the form to the desired folder.
5. Set the custom form as the default form for the folder.
6. If there is any existing item in the folder, update these items so they will use the new form when they are opened.

Note: These steps are basically the same for both personal folder or public folder based application. The only difference is where to publish the form to. Also, remember that you will need to republish the form to the folder using the same name if you have modified it.

COM

COM add-ins are supported by all Office programs, not only Outlook. Since COM add-ins run in-process with the host program, custom code generally runs faster than code implemented in VBA. Also, basic COM add-in architecture is consistent across all Office programs, meaning you can create one COM add-in to use with more than one Office program.

When you register your add-in under HKEY_LOCAL_MACHINE, it will be available to every user on this particular machine. There are 3 properties that must be written to the registry to properly control a COM add-in's behavior:

FriendlyName

Holds the name of the add-in that will be shown to the user in the Add-in Manager.

Description

Holds the string that will appear at the bottom of the Add-in Manager when selected.

LoadBehavior

This is a **DWORD** value that specifies the way the COM add-in should be loaded.

Possible values are:

- 0 - Disconnected.
- 1 - Connected.
- 2 - Load at Startup.
- 8 - Load on Demand only.



16 - Loaded and connected the first time the user runs the host application after registration.

For integrating COM with Outlook, COM add-ins must implement a specific interface, namely the IDTExtensibility2. The IDTExtensibility2 interface has 5 event procedures that must be implemented, so Outlook can talk with the COM add-in. And by using these events, you can write dynamic add-ins that runs based on different triggering events:

OnConnection - Occurs when the add-in is connected to the host application.

OnAddInsUpdate - Occurs when changes are made to the collection of add-ins in the Add-in Manager.

OnStartupComplete - Occurs when the host application's startup is completed.

OnBeginShutdown - Occurs before the host application begins unloading.

OnDisconnection - Occurs when the add-in is disconnected from the host application.

To work with COM add-in, you need one of the following development environments:

- Visual Basic 5.0 or later
- Microsoft Office Developer
- Other COM-compliant development environment, like Visual C++ or Visual J++

You may want to use VB6.0 as your development tool. The project type for COM Add-in is ActiveX DLL. Microsoft has a document with step by step instructions on writing code to create COM add-in with VB. The location of the URL is:

<http://msdn.microsoft.com/library/default.asp?URL=/library/techart/trcomad.htm>

According to MS, every line of code within your COM Add-in must use error trapping to prevent any potential run-time errors from occurring. When the OnBeginShutdown event fires in the COM Add-in, meaning when you try to exit Outlook, the forms and folder windows have already been closed. If you try to set these objects to nothing using the COM Add-in's OnDisconnect or OnBeginShutdown events, errors will occur as the object variables are no longer valid. The recommended workaround is to implement the Close event for the Explorer or Inspector object, and then set the corresponding object variable to Nothing at the end of that event.



Outlook Object Model

Items in Outlook are represented by the fundamental objects in the Outlook object model. They represent mail messages, appointments or meetings, meeting requests, tasks, task requests, contacts, journal entries, posts, mail delivery reports, remote mail items, and notes.

Note that Outlook does not support a full object model such as those in Word or Microsoft Excel, which means there are limitations as to what you can develop.

AppointmentItem

- Represents an appointment.
- Store in the Calendar folder.
- Appointment can be either a one-time or recurring appointment.
- Appointment becomes a meeting when the MeetingStatus property is set to olMeeting and one or more resources are designated, including personnel and physical resources.
- The end result of the above is the creation of a MeetingRequestItem object.
- Message class: IPM.Appointment
- ContactItem
- Represents a contact.
- Resides in Contacts folder.
- A contact is any person you have contact with.
- Message class: IPM.Contact

JournalItem

- Represents a journal entry - a record of all Outlook-moderated transactions for any given time.
- Resides in Journal folder.
- Message class: IPM.Activity

MailItem

- Represents a mail message.
- Resides in the Inbox folder.
- Can reside in any other mail folder.
- This is the default item object - the major element of Outlook, such as your Inbox
- Two subordinate objects: RemoteItem and ReportItem objects for handling remote mail items and mail transport system reports, respectively.
- Message class: IPM.Note



MeetingRequestItem

- Represents a change to the recipient's Calendar folder, for example if someone requests a meeting that may change his/her schedule - created automatically when you set the MeetingStatus property of an AppointmentItem object to olMeeting and send it to one or more users.
- Initiated by another party.
- Also initiated via group action.
- No corresponding object exists in the Items collection.
- Return the AppointmentItem object as response as a result of the GetAssociatedAppointment method.
- Message class: IPM.Schedule.Meeting.Request

NoteItem

- Represents a note.
- A note is an annotation attached to a document.
- Resides in Notes folder.
- Message class: IPM.StickyNote

PostItem

- Represents a post in a public folder.
- A public folder is a folder other users can browse.
- It's posted / saved directly to the target public folder without the need to specify recipient.
- The Post method is used to save the post to the target public folder instead of mailing it.
- Message class: IPM.Post

RemoteItem

- Represents a remote item in Inbox or another mail folder.
- Similar to the MailItem object, except that it contains only the Subject, Received, Date, Time, Sender, and Size properties and the first 256 characters of the body of the message.
- Gives someone who is connecting in remote mode enough information to decide if he/she should download the entire corresponding mail message.
- Message class: IPM.Remote

ReportItem

- Represents a mail-delivery report in Inbox or another mail folder.
- Contains a report or error message from the mail transport system.
- Reports include the commonly found nondelivery report.
- Message class: IPM.Report



TaskItem

- Represents a task.
- A task can be an assigned, delegated, or self-imposed task.
- A task can be performed within a specified time frame.
- Resides in Tasks folder.
- Delegated when you assign them to one or more delegates via the Assign method.
- Message class: IPM.Task

TaskRequestItem

- Represents a change to the recipient's task list
- Initiated either by another party or as a result of a group assignment.
- Created automatically when you apply the Assign method to a TaskItem object to delegate the associated task to someone else.
- Manipulated by the GetAssociatedTask method.
- Message class: IPM.TaskRequest

Other Objects

- AnswerWizard Object is for adding or removing application's Answer Wizard files to the Answer Wizard files available to Outlook.
- COMAddIns Collection is for determining if a COM add-in has been installed, or to automatically make a COM add-in available to Outlook.
- DistListItem Object represents a distribution list in a contacts folder.
- Explorers Collection is for you to access every Outlook explorer window in addition to the active explorer window.
- Inspectors Collection holds Inspector objects representing all inspector windows.
- LanguageSettings Object is for determining what language files are installed and in use.
- Links Collection holds Link objects to represent all items linked to a particular item.
- OutlookBarGroups Collection is for adding and removing groups in the Outlook Bar.
- OutlookBarPane Object is for gaining access to groups and shortcuts in the Outlook Bar.
- OutlookBarShortcuts Collection is for managing the shortcuts in an Outlook Bar group.
- OutlookBarStorage Object is for gaining access to groups and shortcuts in the Outlook Bar.
- Panes Collection is for gaining access to the OutlookBarPane object.
- PropertyPages Collection contains PropertyPage objects to represent all the custom pages added to a dialog box.



- PropertyPageSite Object allows for the notification of Outlook that the custom property page has changed.
- Selection Collection contains the set of Outlook items that represent the items currently selected in an explorer window.
- SyncObjects Collection is for you to start, stop, and monitor the progress of offline-folder synchronization for a specific a profile.

Address Book

With Microsoft Outlook object model, one can access information stored in various address books. However, the preferred method of displaying an address book dialog box is to use the AddressBook method in the CDO object model.

As an alternative, use a command button control on an Outlook form that is bound to a recipient field, so that when the user clicks the button, Outlook displays the address book dialog box and the recipient that the user selects will be added to the field. Although Outlook items other than mail message do not have recipients fields, these fields are functional on non-mail forms even though they are not listed.

Customization

Folder Home Pages

- A Web-page view that gives users a quick overview of their appointments, tasks, and messages.
- Developers can develop Web-style view and user interface for Outlook-based applications.

COM Add-ins

- Dynamic-link library that can be loaded by any Office 2000 application and has full access to the Outlook object model.
- Eventually will replace the legacy Exchange Client Extensions.
- You can use Visual Basic version 5.0 or later to create a COM add-in.
- The most effective way to create COM Add-in according to MS is to use Visual Basic version 6.0's add-in designers and Microsoft Office 2000 Developer's Visual Basic Editor.
- Implementing COM Add-in interface requires you to add procedures to handle the five events that constitute the interface. Add-in module must contain procedures for all five events, but the events do not actually have to do anything.
- Visual Basic 6.0 and Office 2000 Developer supply you with the Package and Deployment wizard to automate the tasks for preparing your COM add-in for distribution and installation as compiled DLL.



Team Folders

- Available through the Office Update Web site separately.
- Developer can build sophisticated team-collaboration applications with Outlook and folder home pages via wizard.
- Folder home page uses the Outlook 2000 View Control to embed the functionality of an Outlook explorer window in a folder home page.

VBA

- VBA is supported in Outlook 2000.
- VBA code can be modified by the end users! To be secure, the entire project needs to be password-protected.
- Outlook can support only one VBA project at a time, meaning additional code cannot be added simply by attaching another project. As a workaround, you may export the code to one or more module files, distribute those files to users, and then depend on the users to open the Visual Basic Editor and import the modules.
- You may distribute an entire project file to users, but any previous modifications to a user's project would be lost.
- Users with their own code or import code written by others will receive warning message unless they sign the VBA project or lower their macro security level.

CDO Object Model

CDO stands for Collaboration Data Objects, which is shipped with Microsoft Exchange Server 5.5 and Microsoft Outlook 98/2000. It provides greater functionality than the library available in the Active Messaging 1.1 library shipped with Microsoft Exchange 5.0. It supports capabilities beyond simple Messaging and into the areas of Calendaring, Collaboration, and Workflow, that could greatly simplify the development of heavy-duty resource-scheduling applications requiring information to be displayed through a calendar.

CDO 1.2 is a scripting-object library used to design applications on both the client and server-side. CDO itself is in-process self-registered COM server and is language independent.

We can manipulate CDO with many programming languages, such as Microsoft Visual Basic Scripting Edition, Jscript, and Microsoft Visual Basic for Applications. CDO supports multiple concurrent sessions, meaning you can build applications for clients to log on to Exchange Server 5.5 with an account that is currently logged-on, as well as to support other multi-user environments.

Remember, the primary focus when deploying CDO is for Microsoft Exchange Server 5.5 Scripting and Routing. CDO Rendering Library implements HTML rendering of the CDO objects. It allows you to create instances of programmable rendering objects



that you can reference with automation controllers, to provide web-based messaging solution on top of CDO.

Factors to consider when designing your routing solutions may include:

- Product availability - Is this a client-server environment, or this a peer-to-peer environment?
- Time and cost
- Importance - Server-side solutions should be implemented if you want to avoid possible failure in the process. It is typically more difficult to tightly control the actions of a large group of people in a client-side setup.
- Number of users - Server-side routing solutions should be the choice if you have many users involved.

Scripting Architecture

Exchange Server 5.5's Event Service is a Microsoft Windows NT Service that runs the *events.exe* program using a Microsoft Windows NT account same as the Microsoft Exchange Service account. It's major components include:

- Event Configuration (Binding Data)
- Event Service (*events.exe*)
- Change List
- Server Folder (public folder or private folder)
- Script Code
- Scripting Agent (*scripto.dll/ss.dll*)
- Routing Engine (*exrteng.dll/exrtobj.dll*)

If a specific event occurs in a folder, the Exchange Scripting Agent script is fired and corresponding actions can be triggered. There are four different types of events which can occur:

- Folder_OnMessageCreated
- Folder_OnMessageDeleted
- Message_OnChange
- Folder_OnTimer

Exchange Server 5.5 event service supply us with three intrinsic objects for getting detailed information of the item, which has fired the above events:

- EventDetails.Session
- EventDetails.FolderID
- EventDetails.MessageID



Outlook 2000 & Exchange Server 5.5 Solutions

The system folder at Folders\System Folders\Events Root\EventConfig_<Your Servername> holds an entry for each Microsoft Scripting and Routing script created. This folder is installed by default as part of the Exchange Server 5.5 setup. The default Windows NT Account specified by the installation is used to run the events.exe service. However, you may use another account if you so choose. This account must be granted with Log on as a service rights in the Microsoft Windows NT User Manager.

Change list is a function used with the Incremental Change Synchronization (ICS) to monitor, export, and import changed items between external datasource and internal information store, or just between two information stores. We use ICS mainly for local replication and for backup of information store data.

Each script you create runs on the server, not on the client. If you have installed a script in a users private mailbox, it is not necessary for the user to be logged on. The script will always run, but it cannot access any resource of a local client machine. Scripts are stored as a hidden item in the folder where you have installed the Exchange Scripting Agent script. They are only visible through the Agents tab in Microsoft Outlook 9x/2000, or through a MAPI utility like MDBVU32.EXE. This utility is included in the Microsoft Exchange Server 5.5 CD-ROM, and can be installed or run separately.

Routing Architecture

Routing Objects are provided for the development of email-based routing and approval applications. There are 3 main components: Engine, Objects and Map actions.

Routing Engine

- A custom event handler running on the Microsoft Exchange Server 5.5 Event Service.
- To execute and track multiple process instances within a public or private folder.
- Also to execute flow-control activities.
- Entirely driven by events, mainly the OnMessageCreated and the OnTimer event that occur in the Exchange Server 5.5 folder where the engine is installed. Whenever a new message arrives or a timer is expired it will update the execution state of a process.
- Call Visual Basic Scripting Edition functions for extra activities.



Routing Objects

- COM objects
- For the creation and manipulation of process maps and define the series of states the engine will track, in order to determine what activities are to be performed at each step.

Process Map Actions

- Default set of common VBScript functions for routing.
- Functions can be extended to provide additional functionality.
- Data (process map, tracking table, log, original posted message) for an individual routing process is stored in a process instance.
- Process instance itself is stored on a hidden message in the folder being monitored.
- A default set of Visual Basic Script actions is included in the file ROUTING.VBS for common routing and approval applications.
- Actions can be for Evaluation or Functional purposes.
- Evaluation Actions:
 - IsTimeout
 - IsReceipt
 - IsApprovalMsg
 - IsApprovedTable
 - IsInvalidRecip
 - IsOOF
 - IsPost
- Functional Actions:
 - Send
 - Receive
 - Consolidate
 - FinalizeReport
 - PreProcessing
 - AutoSet
- Some actions are intrinsic, meaning they are built-in to the routing engine. You use them primarily for process and flow-control.

Routing Wizard

- Microsoft Exchange Routing Wizard is included in the Microsoft Exchange Server 5.5 Service Pack 1 (or higher).
- You use it to create and modify Exchange Server 5.5 Routing scripts.
- It needs to be installed separately.
- It requires Outlook 98/2000 to be installed on the same machine.
- Routing Wizard installs a process map into an Exchange Server 5.5 folder to define the steps in the route.



- You may also use the Agent Editor, which is included at the MSDN October 98's Platform SDK Update section.

Object Model

RouteDetails Object

- Contains the details of the route.
- Top-level object, meaning we must use it to obtain objects on the next level.
- Provides the process instance object ProcInstance.
- Create WorkItems.

ProcInstance Object

- An instance of a process being executed.
- Consists of the original MAPI message.
- Top-level object.
- Created independently or obtained through the RouteDetails object.
- Must be associated with the specific message where it resides by using the Open method.

Map Object

- Process map used to execute and track a particular process instance.
- There must be one default map in every routing-enabled folder.
- Resides in a hidden message in the folder.
- Top-level object.
- Created independently.
- The map you create can override the default map of a folder.

Row Object

- Represents a single row or activity in the process map.
- Top-level object.
- Created independently.
- You are responsible for providing values for the Action, ActivityID, and Flags properties.
- ActivityID is a unique line number identifying the row.
- Action is the name of the action.
- Flags define the type of action.

Log Object

- For logging the execution of individual activities in the map.
- Non top-level object
- Obtained via the Log property of a ProcInstance object.



Participant Object

- To refer to and manipulate the e-mail addresses for routing applications.

VoteTable Object

- For programmatically creating Microsoft-Outlook-compatible "Voting Button" properties on a message.

WorkItem Object

- Represents the data to be routed or approved.
- A Message object wrapped with additional interfaces for correlation and consolidation – handled by the routing engine automatically.

Permissions

- Proper NT permission settings are required to run scripts properly.
- Exchange Scripting Agent creates a CDO session based on mailbox identity of the last Scripting Agent script editor.
- The Scripting Agent runs on the Microsoft Exchange Server service account, which means the Scripting Agent has the same NT security context as the Exchange Server itself.
- Permissions must be assigned to a particular mailbox to allow it to create and edit scripts. This is done via the Microsoft Exchange admin program. Note that you need at least Author permissions for this purpose.
- Additionally, you must have Owner permission to a particular public folder to see the Agents tab in the folder properties dialog.

CDO Rendering and Web Solutions

- Objects in the CDO Rendering Library can be classified as top-level objects, child objects, and collections:
 - A top-level object can be created directly by your code without being derived from any other object.
 - A child object must be derived from another object.
 - A collection is a group of objects of the same type.
- Top-level CDO Rendering objects include:
 - RenderingApplication
 - ContainerRenderer
 - ObjectRenderer objects
- Other objects are accessible only through the 3 top-level objects.
- All CDO Rendering Library objects can be considered as relative to a RenderingApplication object.
- CDO Rendering Library supports server-side script, and the CDO Library supports client-side script.



Outlook 2000 & Exchange Server 5.5 Solutions

- HTML Rendering: output is generated (from objects and properties referenced in an .ASP file invoked) and is sent in HTML format to client Web browser.
- An .ASP file is a HTML file containing hypertext, client-side script, and Active Server Pages script. ASP script can be delineated by either the <% and %> tags or the <SCRIPT RUNAT=SERVER> and </SCRIPT> tags.
- With Microsoft Outlook HTML Form Converter, forms created with Outlook can be converted. After conversion, the resulting HTML/ASP forms can be used by anyone running Outlook Web Access. This allows users to access data on Exchange Server computer using an Internet browser that supports frames and JavaScript.
- Web Access uses only files of these types: .asp, .htm and .gif.
- Web Access provides Web-based public access to Microsoft Exchange Server public folders and the global address list. In addition, authenticated users can log on to their personal accounts to read private e-mail and send messages. They can also publish information on the Internet via public folder without manually writing HTML documents.
- Outlook Web Access is included in the Exchange Server version 5.5 distribution CD. For Outlook HTML Form Converter, you need to run the FCsetup.exe program in the \Eng\Formscnv folder on the distribution CD of Exchange Server Service Pack 1.

RenderingApplication Object

- Can be created either through early or late binding. Early binding is usually preferable, as it enforces type checking and generates more efficient code.
- Provides framework for a rendering application.
- You set options on it that are inherited by all rendering objects created by the CreateRenderer method.
- Interface instantiated by it can handle event logging and performance monitoring.

ContainerRenderer Object

- Renders the rows of a container object (such as an address book) as an HTML table.
- Inherits all the functionality of the object renderer.

ObjectRenderer Object

- Can be applied to a CDO object and render its selected properties.



Debugging

- Always test your script in a non-production environment before you deploy it on the production machines.
- Make sure that you are logged-on to the Exchange Server computer with the same Windows NT account for the Microsoft Exchange Server Event Service.
- The mailbox used for testing must have *Owner* permissions on the system folder `Folders\System Folders\Events Root\EventConfig_<Your Servername>`.
- You should install Microsoft Script Debugger properly on the Microsoft Exchange Server computer via NT Option Pack or Control Panel - Add/Remove Programs.
- To debug a Microsoft Exchange Scripting and Routing script, put a STOP statement in your code to have the Microsoft Script Debugger popup.
- You cannot modify the code while you are in the debugger window. That means you need to close the debugger first before changing your code.
- You cannot debug Microsoft Exchange Server Scripting and Routing scripts remotely unless you are using third-party remote-control software.
- You may also use the agent log - a hidden message in each folder where you have a Microsoft Exchange Server Scripting and Routing script installed. To access this agent log, you open the particular private or public folder and right click it, then choose Properties, Agents. Then select your already installed agent and choose Edit.

Exchange Data Services

- Data Access Objects (DAO) is the programming interface to access and manipulate database objects deployed by Exchange Server. DAO are objects that work with the Jet database engine. Once created, a DAO object can be accessed and manipulated by any application that can use the Jet engine. Outlook is one of this kind of application.
- Open DataBase Connectivity (ODBC) is a standard, database access method developed by Microsoft to make it possible to access any data from any application, regardless of which database management system is handling the data. This is accomplished by inserting a middle layer, called a database driver, between an application and the DBMS, to translate the application's data queries into commands that the DBMS understands. Both the application and the DBMS must be ODBC-compliant.
- ActiveX Data Objects is Microsoft's newest high-level interface for data objects for replacing Data Access Objects and Remote Data Objects. One reason to replace them is that they have performance bias for accessing relational databases, while ADO can be used to access all sorts of different types of data, including both relational and non-relational. ADO works with



Outlook 2000 & Exchange Server 5.5 Solutions

- OLE DB and ODBC to form the main components of Microsoft's Universal Data Access (UDA) specification.
- To learn more about DAO, please visit [MSDN Online Library](#). Keep in mind that Active Data Object ADO is now seen as the replacement for the aging DAO technology, so it is always recommended over other methods.
 - Exchange data files are in ISAM (Indexed Sequential Access Method) format, a method for managing how a computer accesses records and files stored on a hard disk.
 - Although storing data sequentially, ISAM provides direct access to specific records through an index.
 - When you install the Exchange Data Source driver, Setup will write a set of default values to the Windows Registry in the Engines and ISAM Formats subkeys
 - Msexch35.dll is the driver used for external access to Microsoft Exchange folders.
 - Jet\3.5\Engines\Exchange folder includes initialization settings for Msexch35.dll. Entry type as follows:
win32=<path>\MSEXCH35.dll

Solution Distribution

- All of your program information from Outlook is stored in a file with the extension .pst. PST file-based deployment is not recommended for multi-user environment.

If you want to distribute a VBA project to the client computers, copy the .OTM file from the source computer to all the target computers and then restart Outlook. The OTM file is the file that holds your VBA project for Outlook.

- For large-scale deployment, consider converting your Visual Basic code into an Outlook COM Add-in.
- Users must add the COM Add-ins menu choice to their menu system before they can use your Add-in. To add this menu item, select Tools | Customize and click on the Commands tab to display the COM Add-ins menu item, and then click on the Tools entry in the Categories column.

A Cabinet (.cab) file is a single, compressed file that can contain multiple files. This is the preferred way for COM component distribution, as data compression is performed across file boundaries, significantly improving the compression ratio. Also, in the Cabinet Properties tab, you can select to Sign Cabinet for applying a digital signature onto it.

- If you have a personal certificate and private key from a CAB, click Use Certificate file. If you just want to sign the cabinet file for testing with the Visual Studio test certificate, click Use Test Certificate.
- Creating a CAB file involves creating an .INF file and then run the CABARC utility. The INF file is a text file that specifies the files that need to be present



Outlook 2000 & Exchange Server 5.5 Solutions

- or downloaded for your control to run. Since COM add-ins are ActiveX DLLs, they can be included easily.
- CABARC should be run in the directory that contains your source files and the INF file. The files to be archived in the CAB file should be listed on the command line in the same order they are listed in the INF file. The -s option allows you to reserve space in the cabinet for code signing.

Special Thanks to [Michael Yu](#) for contributing this Cramsession. Make sure to visit his site at: <http://michaelyu.freeservers.com/>