Outlook 2003 notifies you of one, is a separate question and one I can't explain.


Q: Can I turn off Microsoft Office Outlook 2003's automatic recognition of hyperlinks in email messages?

A: Out of the box, Outlook 2003 will always turn a URL in the body of an email message into a clickable link if you type a space or punctuation character after it or press Enter to start a new paragraph. Until now, there was no way to turn that behavior off, but Microsoft has released a hotfix that adds support for a new registry key to turn off automatic URL recognition. You can learn more about this hotfix and how to obtain it in the Microsoft article “When you type a URL in an e-mail message in Outlook 2003, the URL is recognized as a hyperlink” (http://support.microsoft.com/?id=941537). The hotfix requires Microsoft Office 2003 Service Pack 2 (SP2). After you install the hotfix, navigate in the registry editor to the HKEY_CURRENT_USER\Software\Microsoft\Office\12.0\Outlook\Preferences subkey and add a new DWORD value named DisableAutoUrlCompletion. To disable URL recognition and hyperlinking, set the value for DisableAutoUrlCompletion to 1.

Q: We're using Microsoft Office Outlook 2007 and want to add a task marked with a category named Service to a user's task list and have it appear to that person in the same color for the Service category that everyone else sees. Why won't Outlook let us create a new category and apply it to items in a user's folder?

A: Category names and colors are part of the user's mailbox settings. As with rules to process incoming items, only the mailbox owner can change the category names and colors. Thus, if you add a task and click the Categories button, you'll see only the categories that the mailbox owner has defined; you won't be able to add others.

However, a partial solution to your problem exists. Create the task in your own Tasks folder and assign it the Service category. Then move the task to the user's folder. When the user looks at his or her tasks, the task you created will be marked with the Service category, just won't have the special color that you specified for Service and won't be in the user's Master Category List. The user can then, of course, add the category to his or her Master Category List and assign the category a color.

Sue Mosher (examinsturtlefoot.com) is a contributing editor for Windows IT Pro and an independent consultant working on Outlook development and deployment issues. Her most recent book is Configuring Microsoft Outlook 2003 (Digital Press).

Simple Customer-Satisfaction Surveys

Custom Outlook forms let you easily pick users’ minds

by Jason Joy

If you support internal users at your organization, you need to know what they think of your service. Customer satisfaction surveys are indispensable for identifying problem areas and documenting excellent service to upper management. If you already have a survey process in place, consider the following functionality to enhance it. And if you aren't getting feedback from your users, it's never too late to start.

I've created a survey system that uses custom Outlook forms published to the Exchange Organizational Forms Library so that all users can open and view the forms. If you can't get publishing access to this library, you'll have to look for an alternative process. Also note that these forms are available only in Outlook; Outlook Web Access (OWA) or other email programs can't see the forms.

I chose Outlook as my survey-distribution method to improve response rate. I've participated in various survey methods over the years, including calling users directly and sending them to Web forms. And I've found that people are more likely to respond to a survey if it's easy for them to participate. This solution presents users with a single form and gives them convenient multiple-choice
selections. Since I’ve been running this system, my organization has sent out 2391 surveys and received 1194 responses, which represents a response rate just shy of 50 percent. When we directed users to a Web form, we were lucky to see response rates of 20 percent to 35 percent. I hope you’ll experience as much success with this system as I have.

In this article, I walk you through creating the survey system in two stages. After completing the first stage, you’ll have a custom Outlook survey form that you can manually send and collect. The surveys come from whoever initiates the survey distribution, and they’re returned to a single email account of your choosing for manual processing of the survey results. The second stage lets you automate survey distribution and collection. To complete the automation process, you need Microsoft Office Outlook 2003 and a ticket-tracking system accessible through a database connection. The database gathers the survey recipient’s email address and a reference or description of the ticket or incident the survey covers. My example uses a Microsoft SQL Server database, but you can easily modify it to reflect your environment.

Creating the Launch Form
The basic form that launches the survey simply populates the response form and assigns the return email address for the survey collector. To design the form, from the Outlook menu, go to Tools, Forms, Design a Form. In the Standard Forms Library option, choose Message, then click Open.

The only modification you need to make to this form is adding some VBScript code. To enter the code, open the Script Editor, which you can find in the menu under Form, View Code. In the Script Editor, enter the code that Listing 1 shows. (See page 1 for instructions for downloading the code in this article.) You want recipients to use the survey launch form to compose a message to you, not reply to your message. So, the code prevents the item (the original launch form) from opening and instead opens a response form for the user in the Outbox. Be sure to read the code comments in Listing 1 and modify the script to provide a valid email address for the user or account you want to receive the completed surveys.

After you add the code to the survey launch form, close the Script Editor. Then, publish the form by going to Tools, Forms, Publish Form As, and save the form to the Organizational Forms Library. Give the form a descriptive name such as Launch Survey. If you choose a different name, make sure you remember it because you’ll reference it later. Say no if asked to save form definitions. Close the form designer, and say no again when asked to save changes. The changes you made were saved when you published the form.

Creating the Response Form
The response form is what users see
and respond to. My sample survey form contains four multiple-choice questions and one comments area. You can expand this form by using your own layout and wording. To create the response form, go again to Tools, Forms, Design a Form, and choose the Message form from the Standard Forms Library.

In the form designer, make sure the Edit Compose Page is selected, then begin by deleting all the page elements (i.e., To, CC, Message body) except the Subject area. You use the Subject area to reference the survey topic. Use the Toolbox to add labels for your survey questions or any title text. If the Toolbox doesn’t appear by default, you can access it from the Forms menu by clicking Control Toolbox, which opens the Toolbox in a dialog box.

Next, add a combo box beside each question and provide a large text area for any comments. Figure 1 shows an example of how the completed form might look. Notice that I resized the Subject text box and changed the Subject label’s caption to read Reference. Feel free to change the layout or font or add or remove questions to suit your needs. Any changes you make at this point are purely aesthetic and have no effect on the form’s function.

After you’ve designed your response form, you need to add user-defined fields to the form. Begin by right-clicking a combo box and choosing Properties. On the Value tab, click New, and provide a descriptive name for the question the combo box is associated with. In this example, I named the first question OverallSatisfaction. After creating the field name, assign possible field values, separating them with a semicolon, as Figure 2 shows. Repeat the process for the remaining combo boxes. I named the other combo boxes TimeSatisfaction, TechSatisfaction, and ConfidenceSatisfaction and called the text area UserComments. Don’t forget to select the text-box properties option to allow multiline text. And for the comments box, set the field’s initial value to nothing by entering empty double quotes (“ “). Now select the Edit Read Page option in the form designer to design what the survey will look like when it’s returned. Delete the text area for the message body, and open the Field Chooser. (If you don’t see the Field Chooser in the Toolbox, you can access it in the Forms menu.) In the Field Chooser, change the drop-down option to User-defined Fields in Inbox. Drag each field you just created onto the form. You can rearrange and resize the text boxes as you want. And, again, don’t forget to set the text boxes to handle multiline text. You can also edit the captions to be more descriptive. Figure 3 shows what the returned survey form might look like.

To finalize the form, on the form designer Properties tab, check the box on the right side to Use form only for responses. Optionally, you can use the Version field at the top right to set a version number for the form. By giving the form a version number, you can use the Help, About This Form dialog box to pinpoint which version is in use if you need to troubleshoot any problems. Publish the form by going to Forms, Publish Form As, selecting the Organizational Forms Library, and naming the form Survey. If you choose a different name, it must match the form name in Listing 1. After publishing the form, you can close it, saying no if asked to save the changes or form definitions.

You now have a fully functioning manual survey system. To send a survey from Outlook, select File, New, Choose Form. In the Look In dropdown list, choose Organizational Forms Library if it isn’t already selected, then select the Launch Survey...
form and add the recipient’s email address. In the Subject line, provide a description of the incident you’re requesting feedback about.

The email address you provide in Listing 1 as the survey collector will receive the responses. To collect the results manually, you need access to the email account or if it’s a distribution list—DLO—your need to be a member of the list. You can then open each message to see the survey responses. To compile the responses, you have to either transcribe or copy and paste the information to another location, such as a Microsoft Excel spreadsheet.

**Automating the Process**

Automating these processes requires adding some Visual Basic for Applications (VBA) code to the Outlook client from which you’ll be sending the surveys. I recommend using Outlook 2003 to digitally sign the VBA code to allow macros to run without lowering your security setting. You can use earlier versions of Outlook to avoid security prompts, you’ll have to adjust the security setting to Low or Medium, which isn’t recommended.

So before adding the VBA code, you need to open Outlook under the mail profile of the account you’ll be sending from and check your security settings. Under Tools, Security, check that the security setting is anything except Very High. The default is usually set to High. In addition, under Help, About Microsoft Office Outlook, verify that Security Mode is set to User Controlled. If it’s set to Administrator Controlled, check with your administrator about making an exception. The administrator could either turn off administrator control by using a Group Policy Object (GPO) setting or could allow an exception in Outlook Security Settings folder to enable this user to send items without getting a prompt. An exception would be the only choice if you’re using Outlook 2002 or Outlook 2000. Make sure that changing any of these settings doesn’t conflict with your organization’s security policies.

Next, you need to create a digital certificate to sign your VBA code, which lets you enable the code even with Outlook security set to High. Locate the Microsoft Self Certification wizard, usually called selfcert.exe and found in the root of the Office folder in your Program Files folder. When you run the wizard, it prompts you to enter a certificate name. Give the

---

**Listing 2: VBA Code to Automate Survey Distribution**

Private Sub Application_Reminder(ByVal Item As Object)
Dim oltem As TaskItem
If Item.Subject = ”Send Surveys” Then
Set oltem = Item
oltem = MsgBox(”Send Surveys Now?”, vbYesNo, ”Send Surveys?”)
If oltem = vbYes Then

'**** Connect to the database. This is an example; you need to change the server, database, user ID, and password.
Set Conn = CreateObject(”ADODB.Connection”)
Conn.Open “Provider=SQLOLEDB;” & _
”Data Source=DBSERVER;” & _
”Initial Catalog=30Name;” & _
”User ID=dbUser;” & _
”Password=dpassword;”

'**** Sample SQL query to gather all tickets closed on day previous to Due Date of task.
Note that sample fields are text data types; you might need to adjust your query to account for different types.
sql = ”select TicketNumber, Summary, EmailID “ & _
”from TicketTable “ & _
”where DateClosed = i” & _
”dateAdd(”d”, -1, oltem.DueDate)) & _

'**** Run the SQL query and call the SendSurvey subroutine.
Set DBSearch = Conn.Execute(sql)
Do While Not DBSearch.EOF
TicketNumber = DBSearch.Fields(”TicketNumber”).Value
Recipient = DBSearch.Fields(”EmailID”).Value
SendSurvey Recipient, TicketNumber & _

DBSearch.MoveNext
Loop

'**** Mark the task as complete.
oltem.MarkComplete
End If
End Sub

Public Sub SendSurvey(ByVal str_recip As String, _
str_subject As String)
On Error Resume Next
Set na = Application.Session
Set fld = MAPISession.GetDefaultFolder(0)(”Outbox”)
Set MAPIMailItem = _
MAPISession.Session.MapleFolderItems.Add(”IPM.Note.Launch Survey”)
With MAPIMailItem
.To = str_recip
.Subject = str_subject
.Send
End With
End Sub
certificate a descriptive name such as SurveyCertificate. Return to Outlook, and open the Outlook Visual Basic editor using Alt-F1 or by going to Tools, Macros, Visual Basic Editor. Under the Tools menu, select Digital Signature. Click Choose, and select the certificate you just created. After you add any code in the following sections, you'll need to close Outlook to save the project and reopen Outlook to enable it. When you reopen Outlook, you'll see a message like the one that Figure 4 shows. Check the box to always trust, then enable the macros.

**Automating Survey Distribution**

My solution uses Outlook Tasks to manage the survey schedule and keep track of when surveys were sent. In Outlook’s Tasks folder, select New Task, and label the subject Send Surveys. Click Recurrence, and set the task to run daily. You also need to turn on the reminder and set an appropriate reminder time for your schedule.

Return to the Visual Basic Editor, and in the ThisOutlookSession module, enter the VBA code that Listing 2 shows. This code contains two subroutines: Application Reminder and SendSurvey. The Application Reminder subroutine starts when your reminder opens. If the reminder matches your task by the Subject line, the subroutine will prompt you to send the surveys. If you click Yes, the subroutine connects to a database to collect the email address and ticket numbers for the surveys, then sends the surveys. The subroutine marks the task as complete and creates a new task for the next day. If you click No, you’ll see a reminder dialog box to either dismiss or snooze the item until a later time. If you snooze the item, the subroutine will prompt you again at the chosen time. If you dismiss the reminder, the subroutine will ignore the task and won’t send any surveys that day. In addition, you’ll have to reopen the task to reset the reminder or manually mark it as complete.

At callout A in Listing 2, notice that you need to modify the database connection string and query for your environment. My example connects to a SQL Server database table called TicketTable, which Figure 5 shows. You extract the following key fields from the table and pass them to the SendSurvey subroutine: TicketNumber, Summary, and EmailID. This example uses the text data type for the fields; you might have to modify your query if you use different data types. The ticket number identifies the record to update in the database when collecting the survey responses. Also notice at callout B in Listing 2 that the code adds a set of characters (“/”//”) that separate the ticket number from the description in the Subject line. If you change this set of characters, you must also change them in the next section.

**Automating Survey Collection**

On the same computer you’re sending the surveys from, you can also automate the collection of the results. First, you need to make sure the email address you’re sending from is the same as the survey-collector address in Listing 1 so that the returned surveys will be sent to the inbox on this computer.

My solution collects the survey results by using an Outlook rule and a script. You begin by entering the code from Listing 3 into the VBA editor. When the rule calls this script, the code connects to the database and updates the appropriate record with the survey results. The code also references the user-defined fields you created on the response form. If you used different field names than I did, be sure to adjust them in this code so that they match.

Notice also that the ticket number that identifies the database record to update is in the subject line. As I just noted, the characters “/”//” separate the ticket number from the description, so if you change the format of the subject line, make sure you use the same delimiting characters or use another method for identifying the ticket number from the subject line.

You can now create a new Outlook

<table>
<thead>
<tr>
<th>TicketNumber</th>
<th>Summary</th>
<th>EmailID</th>
<th>Date/Closed</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1114</td>
<td>Govt. Open Outlook</td>
<td><a href="mailto:jennifer@outlook.com">jennifer@outlook.com</a></td>
<td>7/6/2006</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>99956</td>
</tr>
<tr>
<td>1116</td>
<td>Govt. Open Outlook</td>
<td><a href="mailto:jefferson@outlook.com">jefferson@outlook.com</a></td>
<td>7/6/2006</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>99956</td>
</tr>
<tr>
<td>1117</td>
<td>Add a Question to the Survey</td>
<td><a href="mailto:jennifer@outlook.com">jennifer@outlook.com</a></td>
<td>7/6/2006</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>99956</td>
</tr>
<tr>
<td>1118</td>
<td>Add a Question to the Survey</td>
<td><a href="mailto:jefferson@outlook.com">jefferson@outlook.com</a></td>
<td>7/6/2006</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>99956</td>
</tr>
<tr>
<td>1119</td>
<td>Add a Question to the Survey</td>
<td><a href="mailto:jennifer@outlook.com">jennifer@outlook.com</a></td>
<td>7/6/2006</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>99956</td>
</tr>
<tr>
<td>1120</td>
<td>Add a Question to the Survey</td>
<td><a href="mailto:jefferson@outlook.com">jefferson@outlook.com</a></td>
<td>7/6/2006</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>99956</td>
</tr>
</tbody>
</table>

**Figure 5: SQL Server table containing data that’s passed to SendSurvey subroutine**

**Listing 3: VBA Code to Automate Survey Collection**

```vba
Sub CompileSurveys(Item As Outlook.MailItem)
    Set Conn = CreateObject("ADODB.Connection")
    DBString = "Provider=SQL Server;" & _
    & "Data Source=SERVER;" & _
    & "Initial Catalog=BNName;" & _
    & "User Id=user" & _
    & "Password=pass&text=";
    Conn.Open DBString
    splitSubject = Split(Item.Subject, "/
    & 
    & TicketNumber = Trim(splitSubject(0))
    & 
    & SQL = _
    & "UPDATE TicketTable SET & _
    & "Q1 = " & Item.UserProperties.Item("OverallSatisfaction") & 
    & 
    & "Q2 = " & Item.UserProperties.Item("TimeSatisfaction") & _
    & "Q3 = " & Item.UserProperties.Item("TechSatisfaction") & _
    & "Q4 = " & Item.UserProperties.Item("ConfidenceSatisfaction") & _
    & "Comments = " & Item.UserProperties.Item("UserComments") & _
    & "WHERE TicketNumber = " & TicketNumber & 
    & 
    & Conn.Execute SQL
    & 
    & Set Conn = Nothing
End Sub
```

www.windowsitpro.com/microsoft/exchangeoutlook
rule under Tools, Rules and Alerts.
Start with a blank rule to check messages after they arrive. Under Select Conditions, choose uses the form name form, where form name is the name of your survey form, and click the link to open the selection box. Browse to the Organizational Forms Library, and select the Survey form. On Select Actions, choose Run a Script, where Script is the name of your survey-collection script. Click the link, and choose CompileSurveys to assign the script to the rule. You can also choose to either delete the item or move it to a specified folder to help keep the Inbox organized, and you can select Stop Processing more Rules to avoid conflicts with other rules. You can skip the last section for exclusions. Finish the rule by providing a descriptive name, and turn it on. Figure 6 shows a completed sample rule. If you've already started collecting surveys, you can run the rule now on items already in the Inbox. From the time the rule begins running, it will process the surveys as they come in, updating the appropriate database records with the user responses.

What Do You Think?
The survey-automation process is now complete, but here's a final, quick note about security. If you're using a username and password to connect to a database, weigh the risk of exposing them. The code to distribute and collect surveys resides on one machine under a single profile. Although this setup is reasonably protected by Windows security, avoid using a system administrator account and password, and where possible, use integrated security. If you use integrated security, you'll have to change the database-connection string in your code. If you're using a SQL Server database, replace the user ID and password section with Integrated Security=SSPI; if you're using another type of database, refer to your database documentation for the correct connection string to use. You're now ready to pick users' minds to improve your service and confirm that you're satisfying their needs.

Jason Joy (jflajason@gmail.com) is a systems engineer for a southeastern homebuilder. He specializes in desktop support, Web-based applications, and database solutions.

Comments or questions about Exchange & Outlook Administrator?
Email us at letters@windowstip.com

Exchange Ideas
Tips, news, and community resources for messaging admins

Exchange FAQs

Q: What are the Exchange Server 2007 server roles?

A: Exchange Server 2003 has a basic set of server roles: a back-end server that hosts Information Stores (ISs) and performs the bulk of processing and front-end servers that accept client requests and proxy them to the appropriate back-end server. Exchange 2007 has a more granular set of server roles to give flexibility to larger Exchange deployments, while still letting small-to-medium deployments host all the required roles on one server, if appropriate. The five Exchange 2007 server roles are:

- Mailbox—hosts mailbox and public folder data. This role provides Messaging API (MAPI) access for Outlook clients. Additionally, a variation exists called the Clustered Mailbox role when the Exchange server hosts as part of a clustered environment; the Clustered Mailbox role requires no other Exchange roles running on the cluster instance.
- Client Access—similar to the old front-end server role, the Client Access role provides all other client protocol access, apart from MAPI (i.e., Outlook Web Access—OWA, POP3, IMAP, and Microsoft ActiveSync).
- Unified Messaging—provides client access to mailbox, address box, and calendar via telephone and voice. This role requires special IP-PBX or VoIP gateway software.
- Hub Transport—mail-routing server that forwards mail to another hub transport server, edge