How to Programmatically Read Email using Visual Studio Express

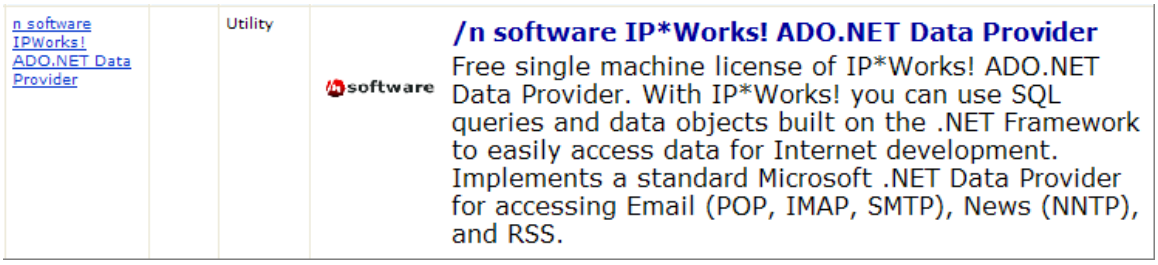
The .NET Framework 2.0 provides developers with the ability to send email using the SMTP protocol, but it does not provide direct support for reading email. In this article, we will take a look at one solution that you can use to programmatically read email from a mail server, using either the POP or IMAP protocol.

Contents

IP*Works! .NET setup and configuration
Introduction to IP*Works! .NET
References and additional resources

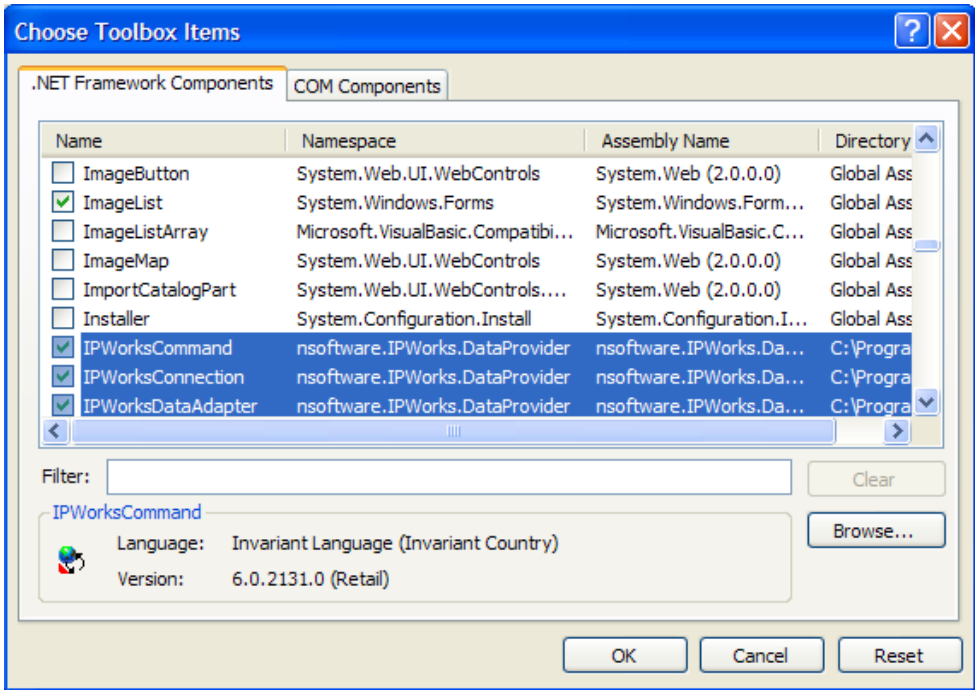
*IP*Works! .NET setup and configuration*

Your first task is to download and install the IP*Works! .NET software. The Express edition of IP*Works! .NET is made available to you when you register your Visual Studio Express product. The registration process can be initiated by selecting **Help | Register Product** from the main menu of your Express product. After you register, you will receive an email from Microsoft with a link to the registration benefits portal site where you will find the IP*Works! .NET download. Downloading and installing the IP*Works! .NET software is quick and easy.



Screenshot - The Registration Benefits Portal site lists your benefits and provides download links

After IP*Works! .NET has been installed, you can add the ADO.NET Data Provider components to your Express toolbox. The instructions can be viewed by loading the document at **Start | All Programs | IPWorks V6 ADO.NET Provider | Readme File** in your default Web browser. If you follow the instructions for Windows Forms development you will end up with three new components: *IPWorksCommand*, *IPWorksConnection*, and *IPWorksDataAdapter*. If you are instead using Visual Web Developer, a component named *IPWorksDataSource* will be added.



Note: It is not necessary to add components to your toolbox in order to use them; it is simply a matter of convenience.

*Introduction to IP*Works! .NET*

Even though the name of the product does not lend itself to being easily pronounced, IP*Works! .NET does provide a simple, unified, and powerful way of working with some common Internet protocols such as POP, IMAP, SMTP, RSS, and NNTP. In this section, we will show how you can utilize the IP*Works! ADO.NET Data Provider to work with these protocols in a consistent way.

If you use email on a regular basis, then you probably have at least some partial recognition of the acronyms POP, IMAP, and SMTP. Most email clients are capable of using the Post Office Protocol, a.k.a. POP, or the Internet Message Access Protocol, a.k.a. IMAP, to work with existing email on a server. The Simple Mail Transfer Protocol, a.k.a. SMTP, is used to send email messages from client machines to mail servers. As we mentioned earlier, the .NET Framework only provides native functionality for sending email using SMTP.

The Readme file that you used to configure your Toolbox also contains some example code that demonstrates how to retrieve email using IMAP and then put it into a DataGrid control for viewing. You could add the following code to an application as long as you also add a DataGrid control named dataGrid1. You would also need to modify the connection string used.

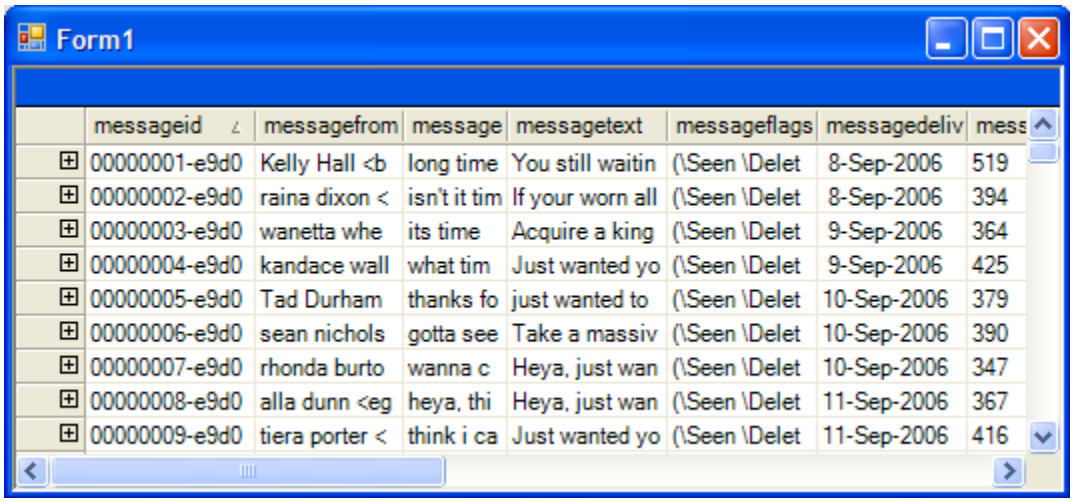
VB.NET:

```
Dim ipworksConnection1 As New IPWorksConnection( _
```

```
        "Protocol=imap; MailServer=myServer; User=myUser; Password=myPassword")
Dim ipworksCommand1 As New IPWorksCommand( _
        "SELECT * from INBOX", ipworksConnection1)
Dim ipworksDataAdapter1 As New IPWorksDataAdapter()
ipworksDataAdapter1.SelectCommand = ipworksCommand1

Dim dataTable1 As New DataTable()
ipworksDataAdapter1.Fill(0, 5, dataTable1)

Me.dataGrid1.DataSource = dataTable1
```

C#:

```
IPWorksConnection ipworksConnection1 = new
IPWorksConnection("Protocol=imap; MailServer=myServer; User=myUser;
        Password=myPassword");
IPWorksCommand ipworksCommand1 = new IPWorksCommand("SELECT * from INBOX",
        ipworksConnection1);
IPWorksDataAdapter ipworksDataAdapter1 = new IPWorksDataAdapter();
ipworksDataAdapter1.SelectCommand = ipworksCommand1;

System.Data.DataTable dataTable1 = new System.Data.DataTable();
ipworksDataAdapter1.Fill(dataTable1);

this.dataGrid1.DataSource = dataTable1;
```



Screenshot - Example Windows Form application with DataGrid populated with email messages

This example shows just how easy it is to use the IP*Works! .NET Data Provider to connect to a mail server and retrieve email. Since the IP*Works! Data Provider was implemented as an ADO.NET Data Provider, developers already accustomed to working with connections, commands, and data adapters will be able to quickly take advantage of existing .NET data objects for viewing, retrieving, and manipulating data.

When working with large sets of data that reside on remote servers, such as with email, you will typically want to limit what you select for transfer. One way to do this would be to use one of the data adapter method overloads for Fill. In the previous example, you could replace the call to Fill with:

```
ipworksDataAdapter1.Fill(0, 5, dataTable1);
```

The result of the alternate Fill command above is that the first five records are retrieved. Alternatively, you could use a more specific selection statement, like the following:

```
"SELECT messagesubject from INBOX WHERE searchcriteria ='SINCE 23-Oct-2006'"
```

This select statement will retrieve the message subject information from all Inbox email that was received on or after the specified date.

Other common SQL commands can be used to work with the IP*Works! .NET Data Provider, such as UPDATE, INSERT, and DELETE. You will need to refer to the help documentation for each connection type in order to know what commands can be used with which column names, however. For example, you can not perform an update to change the text of an existing email message as that operation is not supported by the IMAP protocol.

References and additional resources

For further documentation, sample code, and product information visit:
http://www.nsoftware.com/ipworks/