ExInbox

The ExInbox ia simple-to-use POP3 (Post Office Protocol) client library that allows mail readers to connect to many POP servers and manage email. It implements the client side of RFC 1939. The ExInbox is a free implementation of POP3 protocol for Win32 platforms. The email client can download the entire
email message or only message's header. Non-blocking mode supported. Use the ExInbox (a **POP3** implementation) to get the email messages. Use the ExEMail (a **SMTP** implementation) to send your messages.

Here's a piece of code that shows how easy is to read the messages in VB:

```
Dim WithEvents ibx As Inbox

Private Sub Form_Load()
   Set ibx = New Inbox
   ibx.Read "mail.somewhere.com", "user", "password"
End Sub

Private Sub ibx_Read()
   Dim i As Long
   For i = 0 To ibx.Count - 1
      With ibx(i)
         Debug.Print "From: " & .From & " Subject: " & .Subject & " " & .Size
      End With
   Next
End Sub
```

How to get support?

To keep your business applications running, you need support you can count on.

Here are few hints what to do when you're stuck on your programming:

- Check out the samples - they are here to provide some quick info on how things should be done
- Check out the how-to questions using the eXHelper tool
- Check out the help - includes documentation for each method, property or event
- Check out if you have the latest version, and if you don't have it send an update request here.
- Submit your problem(question) here.

Don't forget that you can contact our development team if you have ideas or requests for new components, by sending us an e-mail at support@exontrol.com (please include the name of the product in the subject, ex: exgrid) . We're sure our team of developers will try to find a way to make you happy - and us too, since we helped.

Regards,
Exontrol Development Team

https://www.exontrol.com

constants StateEnum

StateEnum constans

Name	Value	Description
Ready	0	Ready
Connecting	1	Connecting
Connected	2	Connected
Authorization	3	Authorization
Authorized	4	Authorized
Listing	5	Listing
Listed	6	Listed
Loading	7	Loading
Loaded	8	Loaded
Executing	9	Executing
Executed	10	Executed
Disconnecting	11	Disconnecting
Disconnected	12	Disconnected

Attachment object

The Attachment object holds information about message attachments.

Name	Description
Data	Gets the data of the attachment.
EncodeType	Gets the type of encoding.
LineFrom	Gets the line from.
LineTo	Gets the line to.
Name	Gets the name of the attachment.
Save	Saves the attachment to a file.
Type	Gets the type of the attachment.

property Attachment.Data as Variant

Gets the data of the attachment.

Type	Description
Variant	A safe array of bytes as VARIANT (VT_ARRAY | VT_UI1) or VT_EMPTY if no data is provided for the attachement.,

The Data property returns the data of the attachment as a safe array of bytes. The Save method saves the attachment's data to a file. The for each statement of VB can be used to enumerate the bytes within the array.

The following VB sample enumerates byte by byte, the data of the attachement:

```
Dim h As Variant
For Each h In a.Data
   Debug.Print h
Next
```

property Attachment.EncodeType as String

Gets the type of encoding.

Type	Description
String	A string expression that indicates the type of encoding.

The attachment's encoding type defines how the attachment's content is encoded. The "Content-Transfer-Encoding" header field defines the attachment's encoding type. See the RFC 2045 for all possible values.

property Attachment.LineFrom as Long

Gets the line from.

Type	Description
Long	A long expression that indicates the index of line that indicates where the attachment begins in the message's content

For instance, the following sample prints the first line in the first attachment:

```
With ibx(Index)
    Debug.Print .Line(.Attachment(0).LineFrom)
End With
```

The following sample prints all lines of the attachment:

```
Dim msg As Message, i As Long
Set msg = ibx(Index)
With msg.Attachment(0)
    For i = .LineFrom To .LineTo
        Debug.Print msg.Line(i)
    Next
End With
```

property Attachment.LineTo as Long

Gets the line to.

Type	Description
Long	A long expression that indicates the index of last line in the message that specifies the attachment.

For instance, the following sample prints the last line in the first attachment:

```
With ibx(Index)
    Debug.Print .Line(.Attachment(0).LineTo)
End With
```

The following sample prints all lines in the attachment:

```
Dim msg As Message, i As Long
Set msg = ibx(Index)
With msg.Attachment(0)
    For i = .LineFrom To .LineTo
        Debug.Print msg.Line(i)
    Next
End With
```

property Attachment.Name as String

Gets the name of the attachment.

Type	Description
String	A string expression that indicates the attachment's name

Usually the attachment's name specifies the name of the file that was attached to the message.

method Attachment.Save ([FileName as Variant])

Saves the attachment to a file.

Type	Description
FileName as Variant	A string expression that defines the name of the file to create and save the attachment's data. If no FileName parameter is provided, the Save saves the data using the attachment's name under the current directory.

The Save method saves the data of the attachment to a file. The Name property specifies the name of the attachment. Please be aware that providing no FileName, indicates that the Save method saves the data using the attachment's name under the current directory. The Data property returns the data of the attachment as a safe array of bytes.

The following sample saves all attachments) into the c:\temp folder:

```
With ibx
   Dim iMessage As Long
   For iMessage = 0 To .Count - 1
      With .Item(iMessage)
         Dim iAttachement As Long
         For iAttachement = 0 To .AttachmentsCount - 1
            With .Attachment(iAttachement)
               .Save "c:\temp\" & iMessage & "." & .Name
            End With
         Next
      End With
   Next
End With
```

The file name of each attachment is prefixed by the message's index (0-based), succeeded by the attachment's name such as : 12.image01.png

property Attachment.Type as String

Gets the type of the attachment.

Type	Description
String	A string expression that indicates the type of the attachment.

A string expression that defines the type of the attachment. The attachment's type is defined in the "Content-Type" header field. See the RFC 2045 for more details.

Inbox object

The Inbox object supports the following properties and methods:

Name	Description
Count	Gets the number of messages.
Execute	Executes a command on the server.
Host	Retrieves or sets a value that indicates the incoming mail server address.
Item	Gets the message giving its index.
Pass	Retrieves or sets the account password.
Read	Connects to the host and read messages one by one.
State	Retrieves the connection's state.
Timeout	Specifies the amount of time (in seconds) the control will wait for the server response.
User	Retrieves or sets the account name.

property Inbox.Count as Long

Gets the number of messages.

Type	Description
Long	A long expression that defines the count of messages in the Inbox object.

Use the Item property to access a message in the messages collection. The messages collection is zero based. For instance, the following sample prints the subject for each message in the messages collection:

```
Dim i As Long
For i = 0 To ibx.Count - 1
    With ibx(i)
        Debug.Print .Subject
    End With
Next
```

method Inbox.Execute ([Host as Variant], [User as Variant], [Pass as Variant], [Command as Variant])

Executes a command on the server.

Type	Description
Host as Variant	A string expression that indicates the incoming mail server address. A POP3 server. You can use IP address or domain names as well: Samples: 193.226.40.161, mail.microsoft.com
User as Variant	Specifies the user account. Some servers require the full email address as account, some not. Ask your ISP provider about your account name. Samples: dean, dean@exontrol.net. The User property is used when control sends the USER command, a POP3 command.
Pass as Variant	A string expression that indicates the account's password.
Command as Variant	A string expression that indicates the POP3 command. See the RFC 1939 for the list of supported commands.

Use the Execute method to execute a command on the server. Use the Execute event to get the answer of the server after it executed the command.

For instance, you can use the Execute command to delete a message from the server. Use the Index property to get the index of message on the server. Attention! That index is not the same with the index of the message in the messages collection. ibx.Execute "193.226.40.161", "james", "cucubau", "DELE 1" delete the first message on the server

property Inbox.Host as String

Retrieves or sets a value that indicates the incoming mail server address.

Type	Description
String	A string expression that indicates the incoming mail server address.

Use the Read method to read messages from the server. use the Execute method to execute commands on the server.

property Inbox.Item (Index as Variant) as Message

Gets the message giving its index.

Type	Description
Index as Variant	A long expression that indicates the index of message in the messages collection. Attention! The Index property is not the same thing. The Index property gets the index of the message on the server.
Message	A Message object that holds information about an e-mail message.

Use the Item property to access to Message objects in the messages collection. Use the Count property to get the count of messages in the Inbox object.

property Inbox.Pass as String

Retrieves or sets the account password.

Type	Description
String	A string expression that indicates the password for the account. The string is used by PASS command (a POP3 command).

Use the Read method to read messages from the server. use the Execute method to execute commands on the server

method Inbox.Read ([Host as Variant], [User as Variant], [Pass as Variant])

Connects to the host and read messages one by one.

Type	Description
Host as Variant	A string expression that indicates the incoming mail server address. A POP3 server. You can use IP address or domain names as well: Samples: 193.226.40.161, mail.somewhere.com
User as Variant	Specifies the user account. Some servers require the full email address as account, some not. Ask your ISP provider about your account name. Samples: useraccount, useraccount@somewere.com. The User property is used when control sends the USER command, a POP3 command.
Pass as Variant	A string expression that indicates the POP3 command. See the RFC 1939 for the list of supported commands

The Read method connects the client to the server and gets all the messages for the given account. By default, the control loads the entire message one by one. Use the Reading event to cancel reading of a message.

For instance, the following sample prints the messages, on the server mail.somewhere.com for the account: mike@somewhere.com (or simple mike if the server accepts) :

```
Dim WithEvents ibx As Inbox

Private Sub Form_Load()
   Set ibx = New Inbox
   ibx.Read "mail.somewhere.com", "mike@somewhere.com", "password"
End Sub

Private Sub ibx_Read()
   Dim i As Long
   For i = 0 To ibx.Count - 1
      With ibx(i)
         Debug.Print .Subject
      End With
   Next
```

```
End Sub

Private Sub ibx_Reading(ByVal Index As Long, Cancel As Boolean)
    With ibx(Index)
        Cancel = .Size > 10240
    End With
End Sub
```

property Inbox.State as StateEnum

Retrieves the connection's state.

Type	Description
StateEnum	A StateEnum expression that indicates the connection's state.

Use the State property to check whether the component is ready or it is busy.

property Inbox.Timeout as Long

Specifies the amount of time (in seconds) the control will wait for the server response.

Type	Description
Long	A long expression that specifies the amount of time (in seconds) the control will wait for the server response.

By default, the Timeout property is 30 seconds.

property Inbox.User as String

Retrieves or sets the account name.

Type	Description
String	A string expression that indicates the account's name. This is usually the same as the part of your e-mail address to the left of the "at" sign (@). Some servers requires the entire e-mail address as account name.

Use the Read method to read messages from the server. use the Execute method to execute commands on the server.

Message object

The Message object holds a collection of lines that defines the e-mail message in MIME format. The Message object supports the following properties and methods:

Name	Description
Attachment	Gets the attachment given its index.
AttachmentsCount	Gets the count of message attachments.
Count	Gets the number of the lines in the message including the message's header.
From	Gets the message's sender.
Header	Retrieves the header field attributes giving the name for the header field.
HTML	Gets the message's html text.
ID	Gets the message's identifier.
Index	Gets the index of the message on the server.
Line	Gets the line in the message giving its index.
Load	Loads the message from a file.
Refresh	Refreshes the message.
Save	Saves the message to a file.
Size	Specifies the message's size.
Subject	Gets the subject of the message.
Text	Gets the message's plain text.
UserData	Retrieves or sets an extra value associated to the object.

property Message.Attachment ([Index as Variant]) as Attachment

Gets the message attachment given its index.

Type	Description
Index as Variant	A long expression that indicates the index of the attachment in the attachments collection.
Attachment	An Attachment object that holds information about the message attachment.

Use the Attachment property to access to the message attachments. Use the AttachmentCount property to get the number of message attachments.

property Message.AttachmentsCount as Long

Gets the count of message attachments.

Type	Description
Long	A long expression that indicates the count of the message attachments.

Use the AttachmentsCount property to count the attachments in the message. If the message has no attachments the AttachmensCount property returns 0.

property Message.Count as Long

Gets the number of the lines in the message including the header.

Type	Description
Long	A long expression that indicates the number of lines in the message.

The Count property counts the lines of the message's header. Use the Line property to access a specific line in the message.

The following sample prints the entire message's content:

```
 With ibx(i)
    For j = 0 To .Count - 1
        Debug.Print .Line(j)
    Next
End With
```

property Message.From as String

Gets the message's sender.

Type	Description
String	A string expression that indicates the message's sender

The Form property specifies the email address of the message's sender. The name for the header field is "From". See the RFC 2045 for details about "From" header field.

property Message.Header (Name as String) as String

Retrieves the header field attributes giving the name for the header field.

Type	Description
Name as String	A string expression that defines the name of the header field, or empty to get the entire message's header.
String	A string expression that indicates the header field attributes.

Is the user passes empty string to Header property, it retrieves the entire message's header. See the RFC 2045 for the list of the header field names.

property Message.HTML as String

Gets the message's html text.

Type	Description
String	A string expression that indicates the message's text text

The HTML property gets the message's html text, while the Text property retrieves the message's plain text.

property Message.ID as String

Gets the message's identifier.

Type	Description
String	A string expression that defines the message's identifier.

Use the message's identifier to identify uniquely the messages. The message's identifier is defined by the server. The name of the header field that defines the message's identifier is "Message-ID". See RFC 2045 for details.

property Message.Index as Long

Gets the index of the message on the server.

Type	Description
Long	A long expression that indicates the index of message on the server.

The Index property is not the index of the message in the messages collection. Use the Index property to identify the message on the server. For instance you can use the Index property to delete the message from the server using DELE command. See RFC 1939 for DELE command details.

property Message.Line (Index as Variant) as String

Gets the line in the message giving its index.

Type	Description
Index as Variant	A long expression that indicates the index of line requested
String	A string expression that specifies the message's line.

Use the Count property to count the number of lines in the message.

The following sample prints the entire message's content:

```
With ibx(i)
  For j = 0 To .Count - 1
    Debug.Print .Line(j)
  Next
End With
```

method Message.Load (FileName as Variant)

Loads the message from a file.

Type	Description
FileName as Variant	A string expression that indicates the file name

Use Save method to save a message to a file.

method Message.Refresh ()

Refreshes the message.

Type	Description

Use the Refresh method to reload message from the server, if it wasn't loaded by canceling it in Reading event.

method Message.Save ([FileName as Variant])

Saves the message to a file.

Type	Description
FileName as Variant	A string expression that specifies the file name where the message is saving.

Use Load method to load a message from a file.

property Message.Size as Long

Specifies the message's size.

Type	Description
Long	A long expression that indicates the message's size in bytes.

property Message.Subject as String

Gets the subject of the message.

Type	Description
String	A string expression that indicates the message's subject.

property Message.Text as String

Gets the message's plain text.

Type	Description
String	A string expression that indicates the message's plain text

The Text property retrieves the message's plain text, while the HTML property gets the message's html text.

property Message.UserData as Variant

Retrieves or sets an extra value associated to the object.

Type	Description
Variant	A Variant expression that indicates the object's extra data.

The UserData property is not used by the component.

ExInbox events

Here's the list of messages supported by Exontrol! ExInbox ActiveX object

Name	Description
Debug	Fired each time when the control receives new data from the host.
Error	An error occurred.
Execute	Fired once that Execute method is done.
Read	Occurs once that Read method is done, and all messages were loaded successfully.
Reading	Fired just before reading the message.
Refresh	Fired when the Refresh method ended.
StateChanged	Fired when the connection's state is changing.

event Debug (Command as Boolean, Description as String)

Fired each time when the control receives new data from the host.

Type	Description
Command as Boolean	A Boolean expression that indicates whether the Description specifies a POP3 command or a result sent by the server
Description as String	A string expression that indicates the command or the result sent by the mail server.

The Debug event is fired each time when the control or the server communicates. Use the Debug event to watch how the control fetches the email messages. Use the Debug event to notify your application that the server replies to control's command. Use the Read event to notify your application that the control finished to retrieves the messages. The following sample prints the commands sent by the control to a mail server:

```
Private Sub ibx_Debug(ByVal Command As Boolean, ByVal Description As String)
    If Command = True Then
        Debug.Print Description
    End If
End Sub
```

event Error (Error as Long, Description as String)

An error occurred.

Type	Description
Error as Long	A long expression that indicates the error's number.
Description as String	A string expression that indicates the error's description.

Use the Error event to notify your application that the control fails to fetch the messages. Use the Read event to notify your application that all message were fetched. If the Error event occurs any of the other events are not fired.

event Execute (Result as String)

Fired once that Execute method is done.

Type	Description
Result as String	A string expression that indicates the string sent by the server when the control executes an user command.

The Execute event is fired when control finished the Execute method. Use the Result argument to know what server replies.

event Read ()

Occurs once that Read method is done, and all messages were loaded successfully.

Type	Description

Use the Read event to notify your application that the Read method was done, and all messages were loaded successfully. Use the Error event to notify your application that an error occurs. If the Error event occurs any of the other events are not fired. Use the Reading event to cancel reading the entire message during Read method.

The following sample shows how to print the subject for all messages:

```
Private Sub ibx_Read()
        Dim i As Long
        For i = 0 To ibx.Count - 1
            With ibx(i)
                Debug.Print .Subject
            End With
        Next
End Sub
```

event Reading (Index as Long, Cancel as Boolean)

Fired just before reading the message.

Type	Description
Index as Long	A long expression that indicates the index of Message being read.
Cancel as Boolean	A boolean expression that specifies whether the reading of the entire message is canceled.

Use the Reading event to cancel reading messages that are huge. Use the Reading event to notify your application that a new message is reading. Use the Read event to know the moment when the collection of Message objects are loaded. If a message was canceled during Reading event you have to use the Refresh method to reload the message from the server. Use the UserData property to stores an extra data for the message.

For instance, the following sample cancels the reading of messages that exceed 2M bytes:

```
Private Sub ibx_Reading(ByVal Index As Long, Cancel As Boolean)
    With ibx(Index)
        Cancel = .Size > 2048000
    End With
End Sub
```

event Refresh (Index as Long)

Fired when the Refresh method ended.

Type	Description
Index as Long	A long expression that indicates the index of message that was refreshed.

Use the Refresh event to notify your application when a message was reloaded from the server. Use the Refresh method to reload a message from the server.

event StateChanged (oldState as StateEnum, newState as StateEnum)

Fired when the connection's state is changing.

Type	Description
oldState as StateEnum	A StateEnum expression that indicates the state of the connection before changing its state.
newState as StateEnum	A StateEnum expression that indicates the current state.

Use the StateChanged event to notify your application when the connection's state was changed.