SpicePack 1.1
Component Reference

Table of Contents

Introduction

Chili!Soft SpicePack is a set of COM components that handle commonly used ASP application functionality. These components can be instantiated and called from ASP scripts in order to send and receive e-mail and upload files from client browsers.

This section provides instructions for installing SpicePack and accessing the samples. It also provides reference information for using the SpicePack components.

Notes

- You must use SpicePack 1.1.0 with Chili!Soft ASP 3.6. Earlier versions of SpicePack are not compatible.
- SpicePack is sold as a separate product to use along with Chili!Soft ASP. To purchase SpicePack, go to:

  https://shop.chilisoft.com/securesales/

In this section:

Installing and Uninstalling SpicePack

This topic describes how to install SpicePack 1.1. Before you begin, make sure you meet the following requirements:

- Chili!Soft ASP 3.6 is installed on the computer to which you are installing SpicePack.
- You have 30 MB of disk space available.
- You are logged in as root.
- You know your product serial number. SpicePack requires a valid serial number in order to run. If you downloaded the product from the Web, you should have received an e-mail message that included your serial number. If you are installing this product from a CD-ROM, the serial number is printed on the CD-ROM case.

To install SpicePack, use the following procedure.

To install SpicePack

1. On the computer running Chili!Soft ASP 3.6, log in as root.
2. Stop ASP Server and Web server, as described in "Stopping and Restarting the ASP Server" and "Starting and Stopping the Web Server" in "Chapter 3: Managing Chili!Soft ASP."
3. If you are installing SpicePack from a CD-ROM, skip this step and go to step 4. If you are downloading SpicePack from the Web, download the SpicePack tar file to a temporary directory, and then extract the installation files by using the following command:

   ```
   #> tar -xvf Chilisoft_SpicePack_1.1_[platform].tar
   ```

   where [platform] is the name of the platform on which you are installing SpicePack (linux, solaris, aix, or hp-ux).

   This creates a directory named ChiliSoft_SpicePack_1.1_[platform] under the current directory and populates it with the SpicePack installation files.

4. Change directories to ChiliSoft_SpicePack_1.1_[platform]. If you downloaded SpicePack from the Web, this directory was created in step 3. If you are installing SpicePack from a CD-ROM, this directory is located on the CD-ROM.
5. Run the install.sh script by using the following command:

   ```
   #> ./install.sh
   ```

   This starts the SpicePack setup program.

6. The first screen of the setup program informs you that in order to install SpicePack, Chili!Soft ASP 3.6 must be installed and the Chili!Soft ASP Server and Web server must be stopped. To continue, enter **y** (yes).

   - or -

   To stop the installation, enter **n** (no).

7. Review the End-User License Agreement (EULA) that displays. Enter **yes** if you agree to its conditions.

   – or –

   Enter **no** if you do not agree. If you enter **no**, the Chili!Soft ASP setup program aborts the installation.

8. When prompted, enter the absolute path name of the Chili!Soft ASP 3.6 installation directory.
9. When prompted, enter the SpicePack product serial number.

   The setup program now completes the installation.

10. Restart the Web server and ASP Server as described in "Starting and Stopping the Web Server" and "Stopping and Restarting the ASP Server" in "Chapter 3: Managing Chili!Soft ASP."

The SpicePack components are now installed and ready to use. From the **Chili!Soft ASP Start Page** you can access a diagnostic application that verifies the components are functioning, along with sample applications that use the components, at:

http://[HOSTNAME]/caspsamp

where [HOSTNAME] is the hostname of your Web server.

When you uninstall Chili!Soft ASP 3.6, as described in "Uninstalling Chili!Soft ASP" in "Chapter 2: Installing and Configuring Chili!Soft ASP," SpicePack is also uninstalled. To uninstall SpicePack separately, use the following procedure.

To uninstall SpicePack

1. On the computer running Chili!Soft ASP 3.6, log in as root.
2. Change directories to /[C-ASP_INSTALL_DIR]/casp/lib-chilicom

   where [C-ASP_INSTALL_DIR] is the directory in which Chili!Soft ASP 3.6 is installed.

3. Remove the following files:

   libchilimail.so

   libchilipop3.so

   libchiliupload.so

Chili!Mail (SMTP) Component

The Chili!Mail component enables users to send e-mail messages from an ASP page to an SMTP e-mail server. The Chili!Mail component is compatible with the **NewMail** object included with the Microsoft Internet Information Services (IIS) CDONTS component. However, the Chili!Mail component does *not* support the following properties and methods of the **NewMail** object:

- **ContentBase**
- **ContentLocation**
- **MailFormat**
- **Version**
- **AttachURL**
- **SetLocaleIDs**

Other differences between the Microsoft **NewMail** object and the Chili!Mail component are described in the property and method descriptions that follow.

Chili!Mail Registry Settings

The Chili!Mail component does not use registry settings.

Chili!Mail Control Reference

The Chili!Mail component is registered with the ProgId of "CDONTS.NewMail."

The following ASP script written in VBScript creates an instance of the component:

```
Set mailer = Server.CreateObject("CDONTS.NewMail")
```

Chili!Mail Properties

The Chili!Mail component exposes the following properties:

- **To**
- **From**
- **Cc**
- **Body**
- **BodyFormat**
- **Subject**
- **Host**
- **Bcc**
- **Value**

- **Importance**
- **Retain**
- **WrapLength**

Chili!Mail To Property (String: Read/Write)

The **To** property specifies one or more message recipients. A full messaging address must be provided for each recipient, as shown in the following example:

```
"useraddress@company.com"
```

Addresses must be separated by a semicolon (;), as shown in the following example:

```
"user1@company1.com;user2@company2.com;user3@company3.com"
```

If both the **To** property and the **To** parameter of the **Send** method are supplied, the message is sent to all recipients in both lists.

Chili!Mail From Property (String: Read/Write)

The **From** property is a string that specifies the content of the From field of the message header. It cannot include spaces.

Note

The From field cannot exceed 255 characters, the limit for a single e-mail address. There is no character limit for the To, Cc, and Bcc fields.

Chili!Mail Cc Property (String: Read/Write)

The **Cc** property specifies one or more recipients of a copy of the message. A full messaging address must be provided for each recipient, as shown in the following example:

```
"useraddress@company.com"
```

Addresses must be separated by a semicolon (;), as shown in the following example:

```
"user1@company1.com;user2@company2.com;user3@company3.com"
```

Chili!Mail Bcc Property (String: Read/Write)

The **Bcc** property specifies one or more recipients of a blind copy of the message. A full messaging address must be provided for each recipient, as shown in the following example:

```
"useraddress@company.com"
```

Addresses must be separated by a semicolon (;), as shown in the following example:

```
"user1@company1.com;user2@company2.com;user3@company3.com"
```

Chili!Mail Subject Property (String: Read/Write)

The **Subject** property is a string that specifies the content of the subject line of the message. This property may be left empty.

Chili!Mail Body Property (String: Read/Write)

The **Body** property is string that specifies the content of the message. Line breaks should be sent as carriage-return/linefeed pairs, e.g., "Chr(13) & Chr(10)."

Chili!Mail BodyFormat Property (Long: Read/Write)

The **BodyFormat** property specifies the message format available for the Chili!Mail **Body** property. The values for the **BodyFormat** property can be set as follows:

- 0 indicates that the **Body** property can include HTML.
- 1 indicates that the **Body** property can include plain text only.

Chili!Mail Host Property (String: Read/Write)

The **Host** property is a string that specifies the valid DNS name (for example, "mail.myorg.com") or IP address of the SMTP mail server. The default is "localhost."

Chili!Mail Value Property (Read/Write)

The **Value** property adds one or more headers to the automatically generated headers, such as To, From, Subject, and Date. Possibilities for additional headers include File, Keywords, and Reference.

Certain headers, such as Reply-To, are widely accepted and used by various messaging systems. For such a header to be recognized by recipients, the character string in the header name must exactly match the accepted string.

In principle, you can put any combination of ASCII characters in the string, but some messaging systems might restrict the character set. The safest procedure is to limit the string to alphanumeric characters, dashes, and slashes, and in particular to avoid spaces.

You can set the **Value** property more than once. Each setting generates another header to be included with the existing headers.

Chili!Mail Importance Property (Long: Read/Write)

The **Importance** property specifies the importance of the message to be sent. Valid values are:

- 0 indicates low Importance
- 1 indicates normal importance
- 2 indicates high importance

Chili!Mail Retain Property (BOOLEAN: Read/Write)

The **Retain** property specifies whether or not message properties are retained after the **Send** method is called. If set to **True**, all of the properties are retained. If set to **False** (the default), all properties are cleared.

Chili!Mail WrapLength (Read/Write)

The **WrapLength** property applies to message content. It specifies the maximum number of characters allowed in a line before the line wraps; in other words, before it breaks and continues on the next line. The line breaks at the last space before the specified maximum number of characters has been reached. The default setting is 77. The maximum is 1,000.

Chili!Mail Methods

The Chili!Mail component provides the following methods:

- **AttachFile**
- **Send**

Chili!Mail AttachFile Method

The **AttachFile** method attaches a file to the message. Messages are multi-part mime encoded, and attachments follow the text portion of the message.

Arguments:

Source	*A string containing the absolute path name of the file to attach.*

CDONTS Note

All messages are Base64 encoded. There is no provision for specifying a different encoding method.

Chili!Mail Send Method

The **Send** method sends the message using the properties previously set. All arguments to this method are optional and override the properties previously set for the message (except for the **To** argument, which is combined with any previously set **To** property).

Calling the **Send** method re-sets all message properties in preparation for the next message, unless the **Retain** property is set to **True**. Multiple messages can be sent using the same instance of the Chili!Mail component.

Arguments:

From	*See the description of the property of the same name above.*
To	*See the description of the property of the same name above.*
Subject	*See the description of the property of the same name above.*

Body	*See the description of the property of the same name above.*
Importance	*See the description of the property of the same name above.*
Host	*See the description of the property of the same name above.*

Example 1:

```
Set mailmsg = Server.CreateObject("CDONTS.NewMail")

mailmsg.To = "youraccount@yourco.com"

mailmsg.From = "MailTest"

mailmsg.Body = "This is a test message." & Chr(10) & "This is the
second line."

mailmsg.Host = "mail.yourco.com"

mailmsg.Send
```

Example 2:

```
Set mailmsg = Server.CreateObject("CDONTS.NewMail")

Message = "This is a test message." & Chr(10) & "This is the second
line."

mailmsg.Send ("myaccount@yourco.com", "youraccount@yourco.com",
"Test Subject", Message, 2, "mail.yourco.com")
```

Chili!Upload (File Upload) Component

The Chili!Upload component enables users to save files uploaded by site visitors to the server.

Chili!Upload Registry Settings

The component does not use registry settings.

Chili!Upload Control Reference

The Chili!Upload component is registered with the ProgId of "Chili.Upload.1." The following VBScript excerpt creates an instance of the control.

```
Set report = Server.CreateObject( "Chili.Upload.1" )
```

Chili!Upload Properties

The Chili!Upload component exposes the following properties:

- **AllowOverwrite**
- **SizeLimit**
- **FileSize**
- **SourceFileName**
- **SourceFileExtension**

Chili!Upload AllowOverwrite Property (Read /Write)

The **AllowOverwrite** property determines whether or not the component overwrites existing files saved with the same absolute path name as the uploaded file.

Chili!Upload SizeLimit Property (Read/Write)

The **SizeLimit** property sets the maximum file size in bytes of uploaded files.

Chili!Upload FileSize Property (Read Only)

The **FileSize** property indicates the size in bytes of the uploaded file.

Chili!Upload SourceFileName Property (Read Only)

The **SourceFileName** property indicates the filename of the uploaded file.

Chili!Upload SourceFileExtension Property (Read Only)

The **SourceFileExtension** property indicates the file extension of the uploaded file.

Chili!Upload Methods

The Chili!Upload component exposes the following method:

- **SaveToFile**

Chili!Upload SaveToFile Method

The **SaveToFile** method saves the uploaded file to the location specified by the absolute path name provided by the user.

Arguments:

Path	*The absolute path name for the file, which specifies where it is to be saved.*

Example:

The following script uploads a file:

```
<FORM ACTION="fileupld.asp" METHOD="POST" ENCTYPE="multipart/form-
data">

<INPUT TYPE="FILE" NAME="FILE">

<INPUT TYPE="SUBMIT" VALUE="Send">

</FORM>
```

The following fileupld.asp script processes the upload:

```
<%

Response.Expires = 0

Set fbase = Server.CreateObject("Chili.Upload.1")

fbase.SizeLimit = 10000

fbase.SaveToFile("/opt/datafiles/test.dat")

%>

Done writing <%=fbase.FileSize%> bytes from user file
<%=fbase.SourceFileName%> (of type <%=fbase.SourceFileExtension%>)
```

Note

In a shared Web hosting environment, such as an ISP, you might not know the directory structure above the document root for your virtual host. In this situation, you cannot specify an absolute path name for the file, so you must use the **Server.mapPath** directive instead. The following example saves the uploaded file to the document root of the virtual host:

<%

Response.Expires = 0

```
Set fbase = Server.CreateObject("Chili.Upload.1")

fbase.SizeLimit = 10000

fbase.SaveToFile("Server.mapPath("/") & "/" & "test.dat")

%>

Done writing <%=fbase.FileSize%> bytes from user file <%=fbase.SourceFileName%>
(of type <%=fbase.SourceFileExtension%>)
```

Chili!POP3 (POP3) Component

The Chili!POP3 component retrieves e-mail messages from a POP3 server from an ASP script. This component has two main interfaces. The POP3 interface creates and controls the connection to a POP3 server. The Message interface exposes all of the properties of a single message. Additional interfaces are exposed to support retrieval of message lists and message attachments.

Chili!POP3 Registry Settings

The Chili!POP3 component does not use registry settings.

Chili!POP3 Control Reference

The Chili!POP3 component is registered with the ProgId of "CHILI.POP3.1."

The following ASP script written in VBScript creates an instance of the component:

```
Set pop3 = Server.CreateObject( "CHILI.POP3.1" )
```

Chili!POP3 POP3 Interface

The POP3 interface creates and controls a connection to a POP3 server.

POP3 Interface Properties

The POP3 interface exposes no properties.

POP3 Interface Collections

- **Messages**

POP3 Interface Messages Collection

The Messages collection is a collection of **Message** objects, as described later, in "Chili!POP3 Message Interface." This collection is read-only and does not support the standard **Append** or **Delete** collection methods.

POP3 Interface Methods

The following methods control a network connection to a POP3 server.

- **Connect**
- **Delete**
- **Disconnect**
- **Reset**

POP3 Interface Connect Method

The **Connect** method establishes a network connection to a POP3 server.

Arguments:

Host	*The hostname of the server with which to connect.*
UserId	*The User ID required for connecting with the server.*
Password	*The password required for connecting with the server.*

Example:

See the following Disconnect example.

POP3 Interface Disconnect Method

The **Disconnect** method disconnects from the POP3 server.

Example:

```
Set pop3 = Server.CreateObject("CHILI.POP3.1")

pop3.Connect "mail.foo.com", "myuserid", "mypsswd"

pop3.Disconnect
```

POP3 Interface Delete Method

The **Delete** method deletes a message on the POP3 server. This does not delete the message from the **Messages** collection.

Arguments:

Id	*0-based index for the message in the message collection*

Reset:

Returns the POP3 server to the beginning of the transaction state (Connected) and ignores any commands and their effect on the connection. For example, any messages that were deleted from the mailbox are restored to their un-deleted state.

Example:

```
Set pop3 = Server.CreateObject("CHILI.POP3.1")

pop3.Connect "mail.foo.com", "myuserid", "mypsswd"

pop3.Reset

pop3.Disconnect
```

Chili!POP3 Message Interface

The Chili!POP3 component **Message** interface provides access to the messages currently in the mail store on the connected server. The properties, methods and collections of the **Message** object are used to access those messages.

There are varying network costs associated with accessing the different properties of a message. For POP3 servers that support the optional TOP command, accessing any header information and the first few lines of the message can be accomplished without paying the data transfer overhead of moving the entire message from the server to the client.

Note

When requesting any of the properties that can be gathered without retrieving the entire message, the component first attempts the **TOP** command. If that command fails, the component then attempts to fulfill the property request via the full message **RETR** command.

Message Interface Properties

(LW means potentially "lightweight" on POP3 servers supporting the **TOP** command).

- **From(LW)**
- **Subject(LW)**
- **Size(LW)**
- **DateSent(LW)**
- **DateReceived(LW)**
- **MsgId(LW)**
- **MsgUID(LW)**
- **HasAttachments(LW)**
- **Message**

Message Interface From Property (Read Only)

The **From** property is a string that indicates who sent the e-mail message.

Message Interface Subject Property (Read Only)

The **Subject** property is a string that indicates the subject of the message. It may be null (empty string).

Message Interface Size Property (Read Only)

The **Size** property indicates the total size of the current message in bytes.

Message Interface DateSent Property (Read Only)

The **DateSent** property indicates the date and time that the message was sent.

Message Interface DateReceived Property (Read Only)

The **DateSent** property indicates the date and time that the message was received.

Message Interface MsgId Property (Read Only)

The **MsgId** property indicates the message ID of the current message in the collection.

Message Interface MsgUID Property (Read Only)

The **MsgUID** property indicates whether or not the server supports the **UIDL** command. It returns **0** if the server does not support this command.

Message Interface HasAttachments Property (Read Only)

The **HasAttachments** property provides an "educated guess" based on the message headers (to be lightweight) as to whether or not the message has attachments.

Message Interface Message Property (Read Only)

The **Message** property is string that specifies the content of the message.

Message Interface Collections

The Message interface collections are as follows:

- **To(LW)**
- **CC(LW)**
- **Headers(LW)**
- **Attachments**

Message Interface To Collection

The **To** collection is the list of e-mail addresses to which the message was sent. The collection is read-only and does not support the standard **Append** or **Delete** methods.

Message Interface CC Collection

The **CC** collection is the list of e-mail addresses to which the message was sent as a carbon copy. The collection is read-only and does not support the standard **Append** or **Delete** methods.

Message Interface Attachments Collection

The **Attachments** collection is the list of attachments to the current e-mail message, consisting of file name(s) and description(s). The collection is read-only and does not support the standard **Append** or **Delete** methods.

Message Interface Headers Collection

The **Interface Headers** collection is a collection of all of the message headers for the current e-mail message. This includes headers that are also accessible via friendly named fields or other collections, such as To, From, and DateSent. This collection can be accessed via the header name or index. This collection is read-only and does not support the standard **Append** or **Delete** methods.

Message Interface Methods

- **PreviewMessage**
- **SaveAttachments**

Message Interface PreviewMessage Method

The **PreviewMessage** method returns the specified number of lines of the message body. For servers that support the **TOP** command, this is performed without retrieving the entire message body. For messages with attachments or messages that consist entirely of binary data (which may be ascertained via the **Headers** collection) the first N lines of the message might not be meaningful to a human reader.

Arguments:

Lines	*The number of lines to return.*

Message Interface SaveAttachments Method

The **SaveAttachments** method saves e-mail attachments to a specified directory on the server.

Arguments:

Directory path on the Server	*The complete path name to the directory on the server where attachments are to be saved.*

Note

In a shared Web hosting environment, such as an ISP, you might not know the directory structure above the document root for your virtual host. In this situation, you cannot specify an absolute path name for the file, so you must use the **Server.mapPath** directive instead.

To, CC, and Headers Note

To, **CC**, **BCC**, and **Headers** are **BSTR** collections using the **Count** method to obtain the total number of items in the collection and the **Item** method to obtain each item. The difference is that for **To**, **CC** and **BCC**, the first argument of **Item** is a 0-based index, while for **Headers**, the first argument is a string that indicates the name of the header item (for example, **From**, **To**, **Subject**, and so forth).

Example:

```
Set pop3 = Server.CreateObject("CHILI.POP3.1")
pop3.Connect "mail.foo.com", "myuserid", "mypsswd"
For each item in pop3.Messages
   For each CC in Item.CC
         MsgBox CC
   next
next
pop3.Reset
pop3.Disconnect
```

Chili!POP3 Attachment Interface

The Chili!POP3 **Attachments** collection of the **Message** object provides access to the attachments currently in an e-mail. The properties and methods of the **Attachment** object are used to access those attachments.

Attachment Interface Properties

- **FileName**
- **ContentType**
- **FileSize**
- **Base64**

Attachment Interface FileName Property (Read Only)

The **FileName** property is a string that indicates the name of the attachment.

Attachment Interface ContentType Property (Read Only)

The **ContentType** property is a string that indicates the content type of the attachment.

Attachment Interface FileSize Property (Read Only)

The **FileSize** property is a number that indicates the size of the attachment in bytes.

Attachment Interface Base64 Property (Read Only)

The **Base64** property is a Boolean value that indicates whether or not the attachment is Base64 encoded.

Attachment Interface Methods

- **SaveToFile**
- **Read**

Attachment Interface SaveToFile Method

The **SaveToFile** method saves the attachment on the server.

Arguments:

Directory *The full directory path on the server.*

Note

In a shared Web hosting environment, such as an ISP, you might not know the directory structure above the document root for your virtual host. In this situation, you cannot specify an absolute path name for the file, so you must use the **Server.mapPath** directive instead.

Attachment Interface Read Method

The **Read** method reads the attachment.

Arguments:

Nsize *The number of bytes to read from the attachment. This argument is optional. If missing, the entire attachment is read.*

Pbytes *A safe array of bytes.*