



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

ASP Guide for Clients



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

- [INTRODUCTION](#)
 - [What is ASP?](#)
 - [The ASP Process](#)
 - [ASP: the Product](#)
- [COMPONENTS](#)
 - [ECOM – Using the External Gateway for Transactions](#)
 - [Sample Scripts](#)
 - [SPICE PACK COMPONENTS](#)
 - [Chili!Mail 1.1: SMTP](#)
 - [Chili!POP3 1.1: POP3](#)
 - [The POP3 Interface](#)
 - [The Message Interface](#)
 - [Chili!Upload 1.1](#)
- [WHAT IS POSTGRESQL?](#)
- [ADDING A PostgreSQL DATABASE](#)
 - [Installing the PostgreSQL Driver](#)
 - [Adding the User DSN on to your computer](#)
 - [Instructions for Linking in your Database](#)
- [APPENDIX – FrontPage 2000 Support](#)
 - [FrontPage Database Features](#)
 - [FrontPage Data Connections](#)
 - [File/Folder Database Connections](#)
 - [Access Database Example](#)
 - [Postgres Database Example](#)
 - [Custom Connections using File DSNs](#)
 - [Chili!Soft ASP and the FrontPage Server Extensions](#)
 - [Important Facts to Remember](#)
- [APPENDIX – Who is using PostgreSQL?](#)
- [ASP Websites](#)
 - [Free ASP Sites](#)
 - [ASP Tutorial Sites](#)



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

INTRODUCTION

What is ASP?

Active Server Pages allow you to take advantage of server-side scripting. Furthermore, ASP provides an array of objects and components which manage the interaction between the browser and the web server. Scripting languages such as VBScript and JScript are used to manipulate these objects.

ASP is not actually a language in itself. Meaning there is no ASP code per se, but VBScript or JScript or whichever scripting language you

decide you want to use. VBScript is the most widely used language for ASP.

<http://www.programmersresource.com/articles/whatisasp.asp>

The ASP Process

Originally, the Internet was composed of many static pages, but this is rapidly developing into a vast collection of dynamically flowing information.

Browsers use Hypertext Transport Protocol (HTTP) to call upon a HTML file from a web server. The server responds by forwarding the HTML file to the browser. This has worked well for static pages and 'straight' HTML files.

New developments in dynamic file transfer have evolved quickly. These new gateway designs include Common Gateway Interface (CGI) and Internet Server Application Programming Interface (ISAPI). These gateway interfaces permit the browser to make requests via forms which go to executable programs running on the server which produce HTML output that is sent back to the browser.

Unlike CGI programs, ASP script can be integrated within HTML files, which makes processing these requests faster and simpler.

A server that receives a request with an .asp extension immediately transfers the request to the ASP server. The .asp script is processed by the ASP server that returns HTML to the web server. In turn the web server forwards back to the browser this HTML.

ASP: the Product

ASP, the product will be offered through all of Web Secure's existing hosting packages and as a new product, the "ASPHost", which will be offered on its own.

As ASP is a Microsoft developed technology, it is relatively easy to learn. It does not require an in depth knowledge of programming languages such as perl. This is what makes it so attractive to web designers especially those at 'entry level'.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

The minimum package that is available with the capacity to provide an add-on ASP service would be TheMarketeer hosting package. Although ASP will also be included with the following packages:



TheBuisness



TheMerchant

ASP through a Linux-based ChilliSoft ASP solution is currently being provided by Web Secure with full support offered for ASP. Traditionally Web Secure has run UNIX servers, so therefore we have up until now, been unable to offer this solution. The solution we now offer is a superior solution that scales well and allows us to provide Microsoft Access 2000 and FrontPage 2000 support. Load stress tests have proved that the Web Secure server is able to scale to a large number of simultaneous connections whilst maintaining acceptable performance and is more reliable than windows-based solutions. Since the introduction of this service there has been no downtime. Web Secure is proud to report that our customers are extremely pleased with our level of service regarding ASP support, with several commenting on the speed of the service.

COMPONENTS

Web Secure currently offers the following ASP components¹:

ECOM – Using the External Gateway for Transactions

Using Web Secure's TheMerchant with ASP.

Ecommerce on the ASP server is enabled by including ecom.asp in your ASP script. ecom.asp will be located in the /secure directory off your website base - if it is not, please contact Web Secure Technical Support.

The /secure directory is linked to our https server so you can collect sensitive information securely. Any documents and subdirectories under /secure will be available under the following URL :

https://asp.Web Secure.com.au/your_domain.com.au/

For example, if your domain was www.shopping.com and your script to collect and process credit cards was called "checkout.asp", you would place "checkout.asp" in www.shopping.com/secure and redirect clients to the following URL :

<https://asp.Web Secure.com.au/shopping.com/checkout.asp>



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

ecom.asp provides the following API :

ecom_process (MID,PASSWORD,AMOUNT,CCNUM,CCEXP,COMMENT)

returns : 1 on success, 0 on failure

(Please note : this return code does NOT indicate if the card was successfully processed - it provides very basic sanity checking of the input data, and success here only indicates the request was passed on to the payment gateway.)

All arguments to ecom_process are strings, and all fields must be supplied and non-empty.

MID = Your merchant ID as provided by Web Secure. Should be a 2 or 3 digit number. (e.g. "12")

PASSWORD = Your external access password.

AMOUNT = The amount you wish to charge, including the cents. (e.g. "15.44" = \$15.44)

CCNUM = The credit card number, without spaces or separators. (e.g. "4564123412341234")

CCEXP = credit card expiry date, in the form mm/yy. (e.g. "12/03" for December 2003)

COMMENT = A comment for the transaction, for later tracking. (e.g. "Order #111")

If ecom_process returns 0, the reason for the rejection will be in the ecom_problem variable. (see example script below). You must check if ecom_process succeeds before calling any of the functions below, or your script may return an error.

Upon successful completion of the ecom_process call, the following functions can be used.

ecom_status

returns string ("approved","declined" or "failed")

This function returns the status of the transaction.

◆◆ "approved" indicates the transaction was successful.

¹ Component descriptions from Chili!Soft ASP Documentation:

<http://www.chilisoft.com/caspdoc/>



"declined" indicates the transaction was declined at the bank for some reason. (See the ecom_response_text command below).



"failed" indicates there was a problem with the Web Secure - Bank gateway, or an incorrect Merchant ID / password combination was entered. It could indicate the gateway is down, or that the transaction communication was dropped. This is a rare occurrence. The majority of "failed" responses are caused by incorrect passwords or Merchant ID's. If you continually receive "failed" status, please double-check your



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

password - if there is still a problem contact Web Secure Technical Support at support@Web Secure.com.au

ecom_status should be the first result checked. If the transaction was "failed", other functions may return undefined results.

ecom_txnref

returns string This function returns the bank transaction reference. This number should be recorded for reference. May be empty if the ecom_status is "failed" or "declined".

ecom_settlement_date

returns string This function returns the date on which the transaction took place, in Sydney Australia, Eastern Standard Time (EST).

ecom_response_text

returns string This function returns the text response from the bank. If the transaction was "declined", this function will provide more details. (e.g. "Invalid card", "Insufficient funds".)

Some banks will return "SIGNATURE REQUIRED" in this field for a successful transaction - this is NOT an error - the transaction has gone through successfully.

ecom_card_desc

returns string This function returns the type of card that was processed. (eg "visa", "mastercard"). May be undefined if ecom_status was "declined" or "failed".

ecom_dump

returns string This function returns the full response from the bank. This should only be used for debugging purposes, as the results are not formatted or parsed, and field order may change.

**The following functions are for Quest (National Australia Bank) customers only.

ecom_amount

returns string

This function returns the amount that was processed. The value is in cents (eg \$125.23 = "12523").

ecom_card_number ecom_card_expiry

returns string

These functions return the card number and expiry date respectively.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

ecom_rrn

returns string

Returns the banks rrn number for the transaction.

Sample Scripts

The following sample scripts are provided as an example of how to use ecom.asp. It assumes that your form has collected and checked the required information. Please note that it is vitally important to do safety and sanity checking of any input to your script.

There are two examples, for VBScript and JavaScript ASP.

If your ASP script is not located in the /secure directory, you will need to adjust the #include line appropriately.

Example 1: VBScript

```
<!--#include file="ecom.asp" -->  
<%
```

```
'To store the result from ecom_process  
Dim process_result
```

```
MID = "44"  
PASSWORD = "mypassword"
```

```
'The following are hard-coded for example only.
```

```
AMOUNT = "43.65"  
CCNUM = "4111111111111111"  
CCEXP = "03/01"  
COMMENT = "Order #2323"
```

```
'Process the transaction  
process_result=ecom_process(MID,PASSWORD,AMOUNT,CCNUM,CCEXP,COMMENT)
```

```
'Check the result from the process call.
```

```
if process_result = 0 then  
'Process call failed -check ecom_problem for the problem.  
Response.write "The following error occurred, (credit card was not
```

```
processed) <BR><BR>"  
Response.write ecom_problem  
else
```



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

'Process call succeeded.

```
if ecom_status = "approved" then
Response.write "The transaction was approved.<BR>"
Response.write "The transaction reference for this order
```

is:"

```
Response.write ecom_txnref
Response.write "<BR>The type of card processed was : "
```

```
Response.write ecom_card_desc
```

```
elseif ecom_status = "declined" then
Response.write "The transaction was declined.<BR>"
Response.write "The reason given is : "
Response.write ecom_response_text
```

```
elseif ecom_status = "failed" then
Response.write "The transaction failed. This may be due
to network "
Response.write "problems. Please try again in a few
minutes. If the "
Response.write "problem persists, please contact the
site owner."
```

```
end if
end if
```

```
%>
```

Example 2: JavaScript

```
<%@ Language=JavaScript%>
<!--#include file="ecom.asp" -->
<%
```

```
MID = "44";
PASSWORD = "mypassword";
```

```
//The following fields are hard-coded for example only.
```

```
AMOUNT = "43.65";
CCNUM = "4111111111111111";
CCEXP = "03/01";
COMMENT = "Order #2323";
```



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
//Process the transaction
process_result = ecom_process(MID,PASSWORD,AMOUNT,CCNUM,CCEXP,COMMENT);

//Check the result from the process call.

if (process_result == 0) {
//Process call failed -check ecom_problem for the problem.
Response.write("The following error occurred, (credit card was not

processed) <BR><BR>");
Response.write(ecom_problem);
}
else {

//Process call succeeded.

if (ecom_status() == "approved") {
Response.write("The transaction was approved.<BR>");
Response.write("The transaction reference for this order

is : ");
Response.write(ecom_txnref());
Response.write("<BR>The type of card processed was : ");
Response.write(ecom_card_desc());

}

if (ecom_status() == "declined") {
Response.write("The transaction was declined.<BR>");
Response.write("The reason given is : ");
Response.write(ecom_response_text());

}

if (ecom_status() == "failed") {
Response.write("The transaction failed. This may be due
to network ");
Response.write("problems. Please try again in a few
minutes. If the ");
Response.write("problem persists, please contact the
site owner.");
}
}
}
}
%>
```



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

SPICE PACK COMPONENTS

The Chili!Soft ASP Spice Pack is a collection of server side ActiveX controls that implement and provide many functions common to Active Server Page applications.

The Chili!Soft ASP Spice Pack includes the following server side ActiveX components:

Chili!Mail 1.1: SMTP

Simple Mail Transfer Protocol COM object used to send emails from an ASP page to an email server.

The Chili!Soft Mail 1.1 ActiveX Control enables users to create and send email messages via SMTP from ASP scripts. The Chili!Soft Mail control is designed to be closely compatible with the NewMail object that is included with the Microsoft IIS CDONTS component.

The Chili!Soft Mail Control does not support the following NewMail properties and methods at all:



BodyFormat



ContentBase



ContentLocation



MailFormat



Version



AttachURL



SetLocaleIDs

Other differences between the controls are noted in the individual Property and Method descriptions that follow.

The control makes use of no registry settings.

The Chili!Mail 1.1 ActiveX Control is registered with the ProgId of "CDONTS.NewMail". The following VBScript excerpt shows creating an instance of the control.

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



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

The Chili!Mail 1.1 ActiveX Control exposes the following properties and methods:

Properties:

◆◆ To

(String: Read/Write)

The value you use to set the **To** property can represent a single recipient or a list of recipients. Each recipient must be represented by a full messaging address:

"useraddress@company.com"

Multiple recipients on the list are separated by semicolons:

"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 mail message is sent to all recipients on both lists.

◆◆ From

(String: Read/Write)

A string of what should be sent as the From part of the message header. May not include spaces.

◆◆ Cc

(String: Read/Write)

The value you use to set the **Cc** property can represent a single recipient or a list of recipients. Each recipient must be represented by a full messaging address:

"useraddress@company.com"

Multiple recipients on the list are separated by semicolons:

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

◆◆ Bcc

(String: Read/Write)

The value you use to set the **Bcc** property can represent a single recipient or a list of recipients. Each recipient must be represented by a full messaging address:

"useraddress@company.com"

Multiple recipients on the list are separated by semicolons:

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

◆◆ Subject

(String: Read/Write)

A string to be sent as the subject line of the message. May be left empty.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

◆◆ Host

(String: Read/Write)

The symbolic name (e.g. "mail.myorg.com") or ip address (ww.xx.yy.zz) of the SMTP mail server the control should use to send the message. Defaults to "localhost".

◆◆ Body

(String: Read/Write)

The text portion of the message to be sent. Line breaks should be sent as LINEFEED characters ("Chr(10)").

CDONTS Note: the Body property in the Chili!Soft implementation must be text. There is no option for sending HTML, nor for providing an IStream object as input for this property.

◆◆ Value (Read/Write) The **Value** property is used to add 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. If you wish such a header to be recognized by the recipients of the NewMail object, you must be sure that the character string in the header's name exactly matches the accepted string.

In principle you can put any combination of ASCII characters in the string, but some messaging systems may have restrictions on 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.



Importance

(Long: Read/Write)

The importance of the message to be sent. Valid values are:



0 Low Importance



1 Normal Importance



2 High Importance





Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

Retain
(BOOLEAN: Read/Write)

If set to True, all of the properties will be retained after the Send method is called. If set to False (the default), all properties are cleared after the Send method is called.

Methods:

◆◆ AttachFile Adds a file attachment to the message. Messages are multi-part mime encoded, and attachments follow the text portion of the message.

Arguments

Source	<i>A string containing the full path and filename of the file that should be attached to the message.</i>
---------------	---

CDONTS Note: The Source argument can only be a string. There is no provision for providing an IStream pointer as a source for the file to attach. All messages are encoded Base64. There is no provision for specifying a different encoding method.

◆◆ Send Send the message using the properties previously set. All arguments to this method are optional and override the properties previously set for the message (with the exception of 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 may be sent using the same instance of the Chili!Soft Mail control.

Arguments

From	See the description of the property of the same name above.
To	<i>See the description of the property of the same name above.</i>
Subject	<i>See the description of the property of the same name above.</i>
Body	<i>See the description of the property of the same name above.</i>
Importance	<i>See the description of the property of the same name above.</i>
Host	<i>See the description of the property of the same name above.</i>

Example 1: You can modify the From, To, Subject and Body lines, but you must leave the others as they are, in particular the CreateObject must stay as it is.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

<%

```
set objMail=CreateObject("CDONTS.NewMail")
objMail.From="whoever@yourdomain.com"
objMail.To="whoever@yourcustomer.com"
objMail.Subject="Your Subject"
objMail.Body= "This is a test message." & Chr(10) & "This is the
second line."
objMail.Host = "localhost"
objMail.Send
set objMail=Nothing
```

%>

Chili!POP3 1.1: POP3

Post Office Protocol COM object used to retrieve email from a mail server, using the same protocol used by most email clients.

The Chili!Pop3 1.1 Control enables users to retrieve messages from a POP3 server using ASP scripts. This component has two main interfaces. The first, the POP3 interface, deals with the message collection by retrieving messages and controlling messages from a POP3 server. The second interface, 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.

The control makes use of no registry settings.

The Chili!Pop3 1.1 Control is registered with the ProgId of "CHILI.POP3.1". The following VBScript excerpt shows creating an instance of the control.

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

The Chili!Pop3 1.1 ActiveX Control exposes the following properties and methods.

The POP3 Interface

Properties The POP3 Interface exposes no properties.

Collections Messages

Methods



Connect



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741



Delete



Disconnect



Reset

Messages A collection of Message Objects, as described below. The collection is "read-only", and does not support the standard Append or Delete collection methods.

Connect Establishes network connect to POP3 server.

Arguments

Host	Server to connect to
UserId	<i>User Id to use with server to connect to</i>
Password	<i>Password to use with server to connect to</i>

Example: See the Disconnect example below.

Disconnect Disconnects from POP3 server.

Example:

```
Set pop3 = Server.CreateObject("CHILI.POP3.1")  
pop3.Connect "mail.foo.com", "myuserid", "mypsswd"  
pop3.Disconnect
```

Delete

Delete a message on the POP3 server. It does not delete the message from the Message collection. Arguments

Id	<i>0 based index for the message in the message collection</i>
-----------	--

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 all mail which may have been requested to be deleted from mailbox is restored to the un-deleted state.

Example:

```
Set pop3 = Server.CreateObject("CHILI.POP3.1")  
pop3.Connect "mail.foo.com", "myuserid", "mypsswd"
```



pop3.Reset

pop3.Disconnect

The Message Interface

The POP3 control's Messages collection provides access to the messages currently in the mail store on the connected server. The properties, methods and collections of the Message object described here are used to access those messages.

There are varying costs associated with accessing 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 may 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 may be gathered without retrieving the entire message the control will first try the TOP command. If that fails the control will then attempt to fulfil the property request via the full message RETR command.

Properties

(LW = potentially "lightweight" on POP3 servers supporting TOP command).



From(LW)



Subject(LW)



Size(LW)



DateSent(LW)



DateReceived(LW)



MsgId(LW)



MsgUID(LW)



HasAttachments(LW)



Message



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

Collections



To(LW)



CC(LW)



Headers(LW)



Attachments Methods



PreviewMessage



SaveAttachments

Property Details

From (read)

A string that indicates who sent the mail message.

Subject (read)

A string that indicates what the message may be about. May be null (empty string).

Size(read)

The total size of the current message in bytes.

DateSent(read)

The date and time that the message was sent.

DateReceived(read)

The date and time that the message was received.

MsgId(read)

The message id of the current message in the collection.

MsgUID(read)



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

On servers that support the UIDL command, a unique identifier for this message. Unlike the MsgId property, the MsgUID property should not change across POP3 sessions, allowing for more efficient use of previewing and server resources. 0 if the server does not support the UIDL command.

HasAttachments(read)

An "educated guess" based on the message headers (to be lightweight) at whether or not the message has attachments.

Message (read)

The text portion of the message.

Collection Details

To A collection of email addresses that the mail message was sent to. The collection is "read-only", and does not support the standard Append or Delete collection methods.

A collection of email addresses that mail message was sent to as a carbon copy. The collection is "read-only", and does not support the standard Append or Delete collection methods.

Attachments A collection of attachments of the current mail message, consisting of file name and description. The collection is "read-only", and does not support the standard Append or Delete collection methods.

Headers

A collection of all of the message headers for this message. Includes those headers also accessible via more friendly named fields or other collections (e.g. To, From, DateSent, etc...). May be accessed via the header name or index. The collection is "read-only", and does not support the standard Append or Delete collection methods.

Method Details

PreviewMessage 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 are only binary data (which may be ascertained via the Headers collection) the first N lines of the message may or may not be meaningful to a human reader.

Arguments

Lines	<i>The number of lines desired</i>
-------	------------------------------------



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

SaveAttachments Gets the attachment(s) out of the email and then places them in the specific directory on the server.

Arguments

Directory path on the Server	<i>The full path to where to place the attachment(s) to the email as (a) file(s).</i>
-------------------------------------	---

Note: To, CC and Headers These are all BSTR collections. Using **Count** method to obtain the total number of item in the collection and using **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 indicates the name of the header item (for example, "From", "To", "Subject", etc).

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
```

The Attachment Interface

The POP3 control's Attachments collection in Message object provides access to the attachments currently in a mail. The properties, methods of the Attachment object described here are used to access those attachments.

Properties



FileName



ContentType



FileSize



Base64

Methods



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

◆◆ SaveToFile

◆◆ Read

Property Details

FileName (read)

A string that indicates the name of the attachment.

ContentType (read)

A string that indicates the content type of this attachment.

FileSize (read)

A number indicates the size of the attachment in byte.

Base64 (read)

A Boolean value indicates if the attachment is Base64 encoded.

Method Details

SaveToFile

Save the attachment on the server

Arguments

Directory	<i>The full directory path on the server.</i>
------------------	---

Read

Save the attachment on the server. Arguments

Nsize	<i>The number of bytes to read from the attachment. This argument is optional. If missing the whole attachment will be read.</i>
Pbytes	<i>A safearray of Byte.</i>

Chili!Upload 1.1

Used to upload files from a client (typically a Web browser) to the server and then save these files to the server's file system. The Chili!Upload 1.1 ActiveX Control enables users to save files uploaded by site visitors to the server. The control makes use of no registry settings.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

The Chili!Upload 1.1 ActiveX Control is registered with the ProgId of "Chili.Upload.1". The following VBScript excerpt shows creating an instance of the control.

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

The Chili!Upload 1.1 ActiveX Control exposes the following properties and methods.

Properties:



AllowOverwrite
(read/write)

Determines if the component will overwrite existing files saved with the same path.



SizeLimit
(read/write)

Sets the maximum file size in bytes that can be uploaded with this component.



FileSize
(read only)

The size in bytes of the uploaded file.

SourceFileName
(read only)

The full path and name of the file uploaded by the user.

SourceFileExtension
(read only)

The file extension of the file uploaded by the user.

Methods:

SaveToFile

Saves the stored file from the given filebase with the given identifier to a file with the given path.

Arguments

Contact Web Secure

Path	The full path to the file that the file is to be saved as.
------	--

Support (support@Web Secure.com.au) for more information on the full path. Example: A script that uploads the file:

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



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
<INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>
```

A script that 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%>)
```

WHAT IS POSTGRESQL?

PostgreSQL is an enhancement of the POSTGRES database management system. PostgreSQL has a powerful data model and rich data types. PostgreSQL is a sophisticated Object-Relational DBMS, supporting almost all SQL constructs, including subselects, transactions, and user-defined types and functions.²

The relational database model successfully replaced previous models in part because of its simplicity. Postgres offers substantial power by incorporating the following four additional basic concepts in such a way that users can easily extend the system:

Classes

Inheritance

Types

Functions

Other features that provide additional power and flexibility:

Constraints

Triggers

Rules



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

Transaction integrity

These features put Postgres into the category of databases referred to as *object-relational*. Note that this is distinct from those referred to as *object-oriented*, which in general are not as well suited to supporting the traditional relational database languages. So, although Postgres has some object-oriented features, it is firmly in the relational database category. In fact, some commercial databases have recently incorporated features pioneered by Postgres.

A relational database is a database that is perceived by its users as a collection of tables (and nothing else but tables). A table consists of rows and columns where each row represents a record and each column represents an attribute of the records contained in the table.

ADDING A PostgreSQL DATABASE

This section will provide information on the installation of the PostgreSQL driver and instructions for linking an Access database onto your machine.

Installing the PostgreSQL Driver

On the web page http://www.WebSecure.com.au/support/hosting/asp_adding_postgres.html, click to download the latest PostgreSQL ODBC Drive, which is a zip file.

Save the file onto your hard drive.

Unzip the file and follow the instructions in the set up dialog boxes.

Adding the User DSN on to your computer

Choose the Windows *Start* button on your desktop,



Select *Settings* and then *Control Panel*² All of this information and statistics can be found on the PostgreSQL website – www.postgresql.org

Under the *User DSN* tab, click on



Choose *PostgreSQL ODBC + Driver* and click *Finish*.

The Server name is caelum.WebSecure.com.au

The User Name will have been given to you by the Web Secure Technical Support Team, please enter it here.

The Password will also have been given to you by Technical Support, enter that in the correct



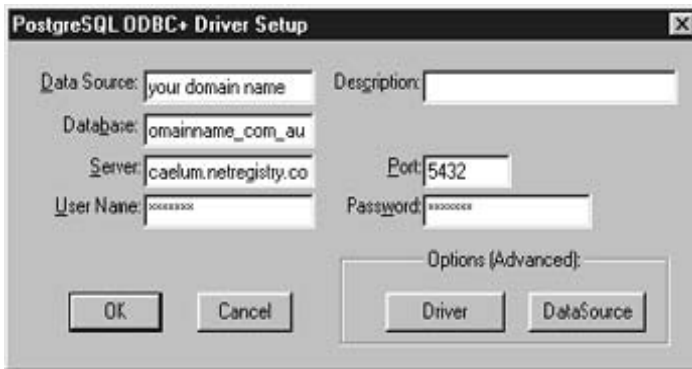
Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

box.

Note: The User Name and Password will be the same as the ftp login and password.



It is **very** important that settings under the *Options (Advanced)* stay exactly as they are, in particular that the *ReadOnly* boxes are *NEVER* checked. If they are checked and you have completed this setup with the database linked in on your machine, then you will have to begin the whole setup process again and make sure that those boxes are *NOT* checked.

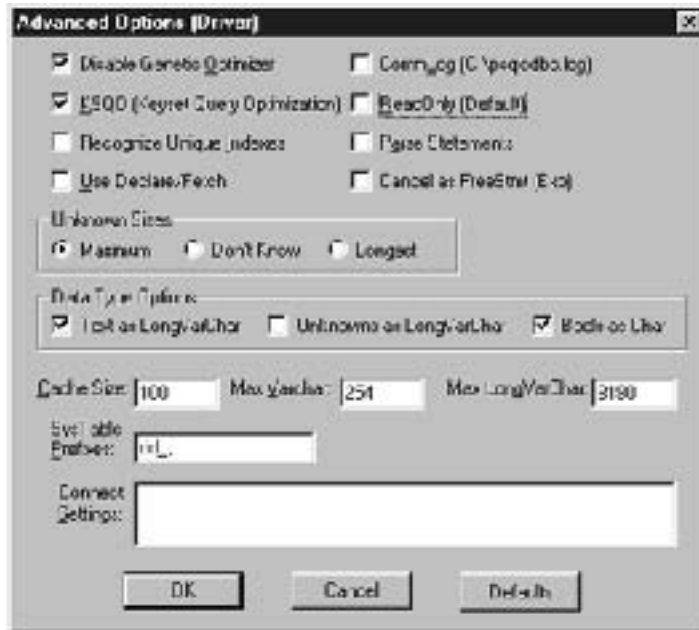
The *Advanced Options (Driver)* screen, with the *ReadOnly (Default)* **UN**checked:



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741



Advanced Options DataSource (PostgreSQL), with the ReadOnly box UNchecked:



Please add these settings exactly as they are before linking in your database. See Instructions for Linking in your Database.

Most Important Features of a PostgreSQL Database:

Postgres is connection pooled therefore those PostgreSQL customers using ASP will find their pages are much faster.

Postgres provides access to a fully-featured SQL database server which MS Access cannot offer.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

Updating Postgres can be done from your own machine, rather than relying on Web Secure to supply the latest version to all our customers. You can do this from the comfort of your own surroundings/office/home.

Commercial and in Confidence Copyright © 2001 by Web Secure Pty Ltd

This work is copyright. Other than as permitted by law, no part of this document may be reproduced, stored in a retrieval system or transmitted in any form by any process without prior written permission.

The uptime of a site running Postgres is increased, as it is UNIX based. Whereas Windows/NT-based MS Access is often rebooted and is a lot less stable.

Instructions for Linking in your Database

To link in a database, open MS Access 2000³. Choose File, Open.



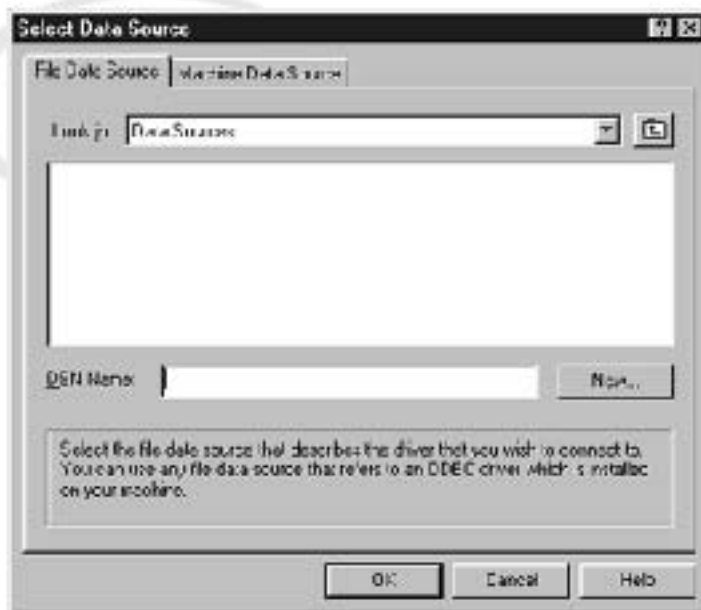
Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741



Change Files of type from All Files (*.*) to ODBC Databases. Now Select Data Source.



Click on the *Machine Data Source* tab.

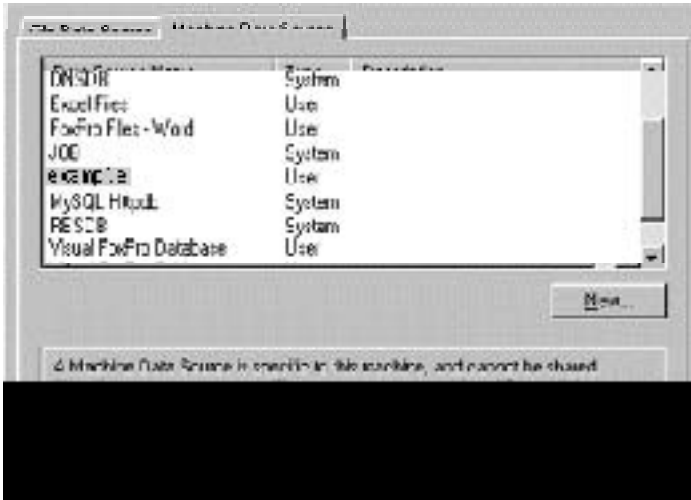
³ These instructions vary if you are using Access 97 or an earlier version. We recommend that you use Access 2000.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741



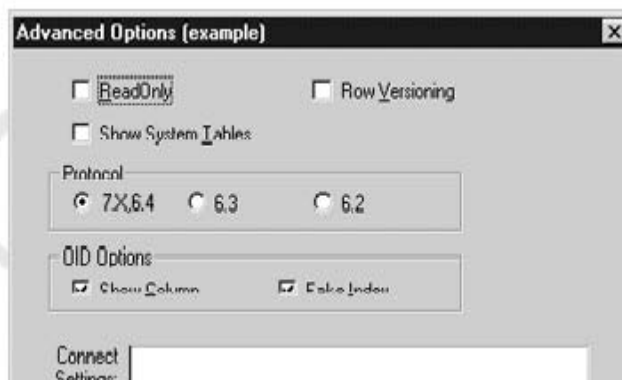
Choose the database, in this case, *example*.

Enter the information for the PostgreSQL Connection as below.



Be sure to thoroughly check the *Options (Advanced) Driver* screen to see that *ReadOnly* (Default) is NOT checked.

Also check the *Advanced Options Connection*, to make sure that *ReadOnly* is NOT checked.





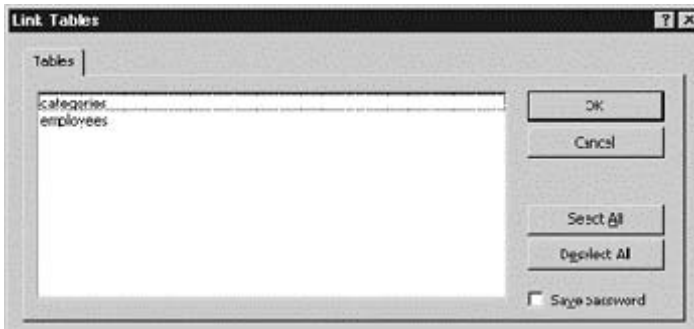
Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

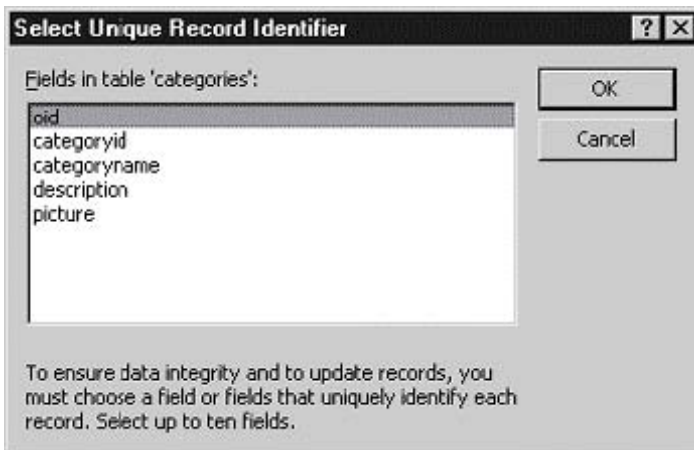


Now you have to link the tables that you will need.



Either *Select All* or select individual tables that you require as in the example below:

It is **very** important that you identify the Unique Record Identifier/Primary Key for each table. You will be prompted for each table with a box like this one:



Please note: If you do not select the Primary Key for each table, you will NOT be able to update your database. Once this is identified, then the tables are added to the ODBC Database.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

APPENDIX – FrontPage 2000 Support

From: <http://www.chilisoft.com/caspdoc/>

FrontPage offers several features that make it simple to add interactive database connectivity to your Web site. You can:

Easily incorporate database queries into your site.

Choose from several different methods for displaying the database information.

Create a custom form in minutes and send results to an existing database.

Once a connection is made to a database, FrontPage makes it easy to display data on a Web page with the Database Results Wizard. By displaying the contents of your ODBC database on a Web page, users will see the most recent data each time they view and refresh a page.

Please contact Web Secure Support – support@websecure.net.au if you wish to utilise database features.

FrontPage Database Features

FrontPage Data Connections

FrontPage uses FrontPage Database Connections to specify the name, location, and type of database to access, along with any driver-specific information, user names, passwords, etc. You enter this information in FrontPage, and it is written to the Global.asa file as a Connection String.

File/Folder Database Connections

FrontPage supports direct access to database files for file-based databases such as Access, dBase, FoxPro, Excel, Paradox, Excel, .txt, .csv, etc. FrontPage constructs a Connection String for the file and stores it in Global.asa. The following example shows the ASP code written to Global.asa for a dBase database:

```
Application("InvSamplnWeb_ConnectionString") = "DRIVER={Microsoft dBase Driver (*.dbf)}; DefaultDir=URL=invsamp/"
```

The "URL=" parameter signifies that the DefaultDir is a Web-relative path. Script in Global.asa maps this relative path to an absolute file system path required by the driver.

Access Database Example

Note - copy and paste the code.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
<HTML>
<HEAD>
<TITLE>Access Database Example</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF>
<H3>Connection Testing</H3>
<p>Enter a DSN and a Table name, and the sample code will dump the returned
records into an HTML table.</p>
<form method="POST" action="msaccess.asp">
```

<p>Database Name (Your domain name with dots replaced with underscores, eg

www.Web Secure.com.au would be Web Secure_com_au) :


```
<input type="text" name="DSN" size="20"></p>
```

<p>Username:


```
<input type="text" name="UID" size="20"></p>
```

<p>Password:


```
<input type="text" name="PWD" size="20"></p>
```

<p>Table Name:


```
<input type="text" name="Table" size="20"></p>
```

```
<p><input type="submit" value="Submit" name="B1"><input type="reset"
value="Reset" name="B2"></p>
```

```
</form>
```

```
<hr>
```

```
<%
```

```
'First, check we have enough information.
```

```
if (Request("DSN") <> "") and (Request("Table") <> "") and (Request("UID") <>
"") and (Request("PWD") <> "") then
```

```
'Create the DSN
```

```
connStr = "DSN=aspmdb;Database=\\kramer\aspsamba\" & Request("DSN") &
```

```
";UID=" & Request("UID") & ";PWD=" & Request("PWD") & ";"
```

```
'Show the DSN to the user, with the password removed.
```

```
Response.Write "DSN : DSN=aspmdb;Database=\\kramer\aspsamba\" &
```

```
Request("DSN") & ";UID=" & Request("UID") & ";PWD=XXXXXXX;"
```

```
'Create the Connection Object
```

```
Set Conn = Server.CreateObject("ADODB.Connection")
```

```
Conn.ConnectionTimeout = 30
```

```
Conn.CommandTimeout = 30
```

```
'Open the Connection object
```

```
Conn.Open connStr
```



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
'SQL Command to execute, which will return a RecordSet object
Set RS = Conn.Execute("SELECT * FROM " & Request("Table"))
```

```
'Finally, loop through the recordset to display result in an HTML table...
%>
```

```
<P>
<TABLE BORDER=1>
<TR>
<% Fori=0to RS.Fields.Count-1%>

<TD><B><% = RS(i).Name %></B></TD>
<% Next %>
</TR>
<% Do While Not RS.EOF %>
```

```
<TR>
<% Fori=0to RS.Fields.Count-1%>

<TD VALIGN=TOP><% = RS(i) %></TD>
<% Next %>
</TR>
<%
RS.MoveNext
```

```
Loop
RS.Close
Conn.Close
```

```
end if
%>
</TABLE>
<BR>
<BR>
</BODY>
</HTML>
```

Postgres Database Example

Note - copy and paste the code.

```
<HTML>
<HEAD>
<TITLE>Postgres Database Example</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF>
```



<H3>Connection Testing</H3>

<p>Enter a DSN and a Table name, and the sample code will dump the returned records into an HTML table.</p>

<form method="POST" action="postgres.asp">

<p>Database Name (Your domain name with dots replaced with underscores, eg

www.Web Secure.com.au would be Web Secure_com_au) :

<input type="text" name="DSN" size="20"></p>

<p>Username:

<input type="text" name="UID" size="20"></p>

<p>Password:

<input type="text" name="PWD" size="20"></p>

<p>Table Name:

<input type="text" name="Table" size="20"></p>

<p><input type="submit" value="Submit" name="B1"><input type="reset" value="Reset" name="B2"></p>

</form>

<hr>

<%

'First, check we have enough information.

if (Request("DSN") <> "") and (Request("Table") <> "") and (Request("UID") <> "") and (Request("PWD") <> "") then

'Create the DSN

connStr = "DSN=aspsql;Database=" & Request("DSN") & ";UID=" &

Request("UID") & ";PWD=" & Request("PWD") & ";"

'Show the DSN to the user, with the password removed.

Response.Write "DSN:" & "dsn=aspsql;Database=" & Request("DSN") &

";UID=" & Request("UID") & ";PWD=XXXXXXX;
"

'Create the Connection Object

Set Conn = Server.CreateObject("ADODB.Connection")

Conn.ConnectionTimeout = 30

Conn.CommandTimeout = 30

'Open the Connection object

Conn.Open connStr

'SQL Command to execute, which will return a RecordSet object

Set RS = Conn.Execute("SELECT * FROM " & Request("Table"))

'Finally, loop through the recordset to display result in an HTML table...

%>



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
<P>
<TABLE BORDER=1>
<TR>
<% Fori=0to RS.Fields.Count-1%>

<TD><B><% = RS(i).Name %></B></TD>
<% Next %>

</TR>

<% Do While Not RS.EOF %>

<TR>

<% Fori=0to RS.Fields.Count-1%>

<TD VALIGN=TOP><% = RS(i) %></TD>

<% Next %>

</TR>

<%

RS.MoveNext
Loop
RS.Close
Conn.Close

end if
%>
</TABLE>
<BR>
<BR>
</BODY>
</HTML>
```

Note:

Excel file connections are not supported on UNIX or Linux. Chili!Soft ASP for UNIX/Linux can connect to an Access database, but the Access database must reside on Windows NT, please contact Web Secure Support for assistance, support@websecure.net.au.

Custom Connections using File DSNs

For assistance with custom connections, please contact Web Secure Support, support@websecure.net.au.



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

Users can create File DSNs that connect to ODBC data sources using the Microsoft ODBC Admin tool. Once the resulting *.dsn file is imported to the Web, FrontPage users can create a connection that points to it. Note that if you are using a file-based database such as dBase, your *.dsn file will contain a hard-coded file reference that may not work when moved to a different server.

The Connection String constructed by FrontPage would look like:

```
Application("Inventory_ConnectionString") = "File  
Name=URL=/Inventory_file_dsn.dsn"
```

A File DSN can be stored in protected directories available only to the Chili!Soft ASP server and the user that created it. The use of File DSNs is typically the most secure approach to providing database access on a shared Web server because the File DSN can contain the user name and password for the database, and when Chili!Soft ASP is running in Inherited User mode, it can only open the file containing the DSN if it is running as a user with permission to open the file.

Chili!Soft ASP and the FrontPage Server Extensions

Chili!Soft ASP supports FrontPage-generated ASP applications on Linux. Included in the FrontPage package are "FrontPage Server Extensions" that enable users to create Web pages for a variety of Web Servers. You may use Chili!Soft ASP and the FrontPage Server Extensions together with the same Web server. There are two points to remember, however.

First, when Chili!Soft ASP and the FrontPage Server Extensions are together on the same machine, you cannot create author ASP pages directly. You must first create any ASP pages generated by the Database Results Wizard on your local workstation (by using a disk-based Web, Personal Web Server, or IIS). After the pages are moved to the remote server running Chili!Soft ASP and the FrontPage Server Extensions, you can use FrontPage to alter the ASP or HTML pages directly.

Second, if you are using Chili!Soft ASP and the FrontPage Server Extensions together with the Apache Web server on Linux, you will need to obtain a special version of the Chili!Soft ASP module for Apache. Further details are provided in the **Configuring Apache** topic of the Chili!Soft ASP for Linux Installation sections. See

<http://www.chilisoft.com/caspdoc/>

Contact Web Secure Support (support@websecure.net.au) from more information on database connectivity, database features and connections using file DSN.

Important Facts to Remember

Remember the following when creating and deploying FrontPage ASP application Web sites with Chili!Soft ASP:



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

Chili!Soft ASP for UNIX or Linux can dynamically recognize ASP applications created by FrontPage 2000, but only if the application is not in a nested subweb. Contact Web Secure Support, support@Web Secure.com.au, if you need assistance with this.

Access databases will run with Chili!Soft ASP on Windows NT, but they will not run on UNIX or Linux servers. Use dBase databases instead.

If you are converting Access to dBase, convert the database (using the Access Export table feature) and import the resulting folder of files to your FrontPage web. Then use the Database Results Wizard.

If you have moved your dBase-based Web application to UUNIX, and Chili!Soft ASP can't open your database, make sure that file extension of your dBase files is all caps (i.e. *.DBF.)

dBase databases do not support multi-table joins on UNIX.

If EnableParentPaths is **False** in the Chili!Soft ASP registry, the CreateObject("Scripting.FileSystemObject") calls generated in Global.asa by FrontPage will not work. Either change EnableParentPaths to **True** (the default), or change the code generated by FrontPage in the Global.asa file to Server.CreateObject("Scripting.FileSystemObject").

APPENDIX – Who is using PostgreSQL?

Birkenstock® – <http://www.birenstock.com.au> aucom – <http://www.au.com> jpcorn – <http://www.jp.com> The University of California, Berkeley - <http://www.berkeley.edu/> Oxford University – <http://www.ox.ac.uk> University of Waterloo (Canada) - <http://www.ace.uwaterloo.ca> University of Wuppertal (Germany) - <http://www.hrzdb.uni-wuppertal.de/world/telsuch/> UNAM (Mexico) - <http://www.deyatelnost.org> High Pressure Research Centre (Poland) - <http://www.unipress.waw.pl/ihaps> Akio Solutions (France) - <http://www.akio-solutions.com>

ASP Websites

Free ASP Sites

<http://www.aspcode.net/>

<http://www.aspfree.com/>



Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

ASP Tutorial Sites

<http://www.learnasp.com/>

Intro to ASP: <http://www.builder.com/Programming/ASPIntro/?dd.cn.txt.0701.10>

<http://www.asp101.com/>

<http://asptracker.com/>

For intranet developers: <http://www.asppipeline.com/>

<http://aspwire.com/>

<http://aspcode.com/>

Active Server Corner: <http://www.kamath.com/>

<http://www.programmersresource.com/>

<http://www.halcyonsoft.com/> - Instant ASP - Active Server Pages on any Platform

<http://www.aspdeveloper.net/> <http://www.devx.com/free/codelib/default.asp> - Source Code Library