WhizBase is programming language tailor-made for easy development of dynamic web content. Although it can be used for building any kind of web site or web application, it stands out with its simplicity of publishing databases on-line. It uses simple syntax to accomplish complex tasks. The results produced by using it significantly exceed the invested resources.WhizBase is actually a script interpreter, which means that its programming instructions are not compiled, but interpreted at runtime.

WhizBase

User's manual

for WhizBase 7.6.0.24

Faik Djikic (February 2015)

1. Introduction

WBSP, (WhizBase Server Pages) is a general-purpose scripting tool (hypertext preprocessor) for Web development and can be embedded into HTML, XHTML, XML, RTF, ASCII, JavaScript, VBScript and any other text based file format.

Its syntax is different from any programming language like C, Java, Perl, VB, and also form other hypertext preprocessors like PHP or ASP. It is easy to learn, and has very short learning curve before one can produce useful results.

Since its first release in 1998 it has been used for building few hundreds sites including eye catching web presentations, web applications, real-time monitoring systems, etc.

The general ideas that drove us during the development of WhizBase included:

- enable non-programmers to make dynamic content
- avoid changing the development routine of our clients
- enrich existing HTML, do not require building everything from the scratch
- make it easy for network and system administrators to build efficient intranet solutions
- make maintenance fast, simple and easy
- use self-explaining syntax easy visible in WYSIWYG editors
- make our software flexible for specific needs
- make secure environments both for developers and hosting providers

2. What is WhizBase

WhizBase is the friendly middleman between your database where you keep your data and the web where you need it to be!

Sounds nice, but does it work?

Here's an example code:

```
<html>
    <body>
    Hello!
    Your IP address is <b>$wbe[REMOTE_ADDR]</b>.<br>
    Today is $wbfn[weekdays], $wbfn[fdt(dd-mmm-yyyy)].
    </body>
</html>
```

In WYSIWYG HTML editor it looks like this:

```
Hello! Your IP address is $wbe[REMOTE_ADDR].
Today is $wbfn[weekdays], $wbfn[fdt(dd-mmm-yyyy)].
```

And this is what output of this script may be:

```
Hello! Your IP address is 192.168.0.203.
Today is Monday, 14-Apr-2008.
```

Even from this short example, you can see how it is different from a script written in programming languages like Perl, JavaScript or C or from a code written using other hypertext preprocessors like PHP or ASP. There are no start and end tags, but very simple placeholders that will be replaced with proper content during page processing, without changing any other part of the page, including text-formatting applied to placeholder.

Since all WBSP code is processed on the server, the visitors receive only the results of the script and have no idea what code was used to generate the page. If you configure WBSP on your server to process files with extension HTM, there is no way your visitors can know what's powering your site.

The great thing about the WBSP is that it can be used to produce really impressive results with only few WBSP instructions included in HTML, and still it can also be used to build a really complex web-based applications. We strongly believe that anyone can produce useful results in less than 30 minutes.

3. What's new in version 7

New features in WhizBase 7 include:
- Digest authentication
- Hash calculation using MD5, SHA1, SHA256, SHA384, SHA512
- Encryption/decryption using AES, AES192, AES256, Blowfish, CAST, DES, RC2, RC4, RC5, TripleDES, UNIXcrypt
- Simple XOR encryption/decryption
- Retrieving RSS feeds
- Retrieving Atom feeds
- Reading and parsing XML files
- Reading and parsing JSON files
- Escape sequences for WhizBase reserved characters
- For and ForEach loops
- Increment and decrement functions

4. Technical information

Application type: HTML preprocessor - RWADE

Operating systems: Windows® 95/98/NT/2000/Me/XP/2003/Vista/2008/Windows7

Minimum system requirements: Pentium 100 MHz, 64 MB RAM, 30 MB HDD space, Windows®-based web server that supports standard CGI applications and filtering

Supported databases: All database types supported by Microsoft® Jet Engine 3.5 and 4.0 using DAO and ADO objects (including Microsoft® Access versions 2.0 to 2003 (MDB files) and Microsoft® Access versions 2007-2010 (ACCDB files), dBASE™ versions III, IV and 5.0, Paradox™ versions 3.x, 4.x and 5.x, FoxPro® versions 2.0, 2.5, 2.6 and 3.0, ASCII files in tabular format) and all databases that support ODBC connections

Engine file size:~ 1 MB

Supported reports: RTF, XML, HTML, DHTML, TXT, RSS, JavaScript, CSS, VB Script, WML, Adobe® FLASH® external data file, XHTML, XAML, WAP and more.

Executable: wbsp.exe -WBSP engine file.

Engine file: original file name WBSP.exe should not be changed.

Windows® and Microsoft® are registered trademarks of Microsoft® Corporation. Adobe® and FLASH® are registered trademarks of Adobe® Corporation.

5. Virtual directories

WhizBase includes support for virtual directories since version 6.1. There are some important notes regarding virtual directories in WB:

- virtual directory must have the same name as physical directory

Physical path:	Virtual path:
c:\inetpub\vhosts\admin	/admin/

- physical path must be added to AllowedPaths

```
Physical path:
c:\inetpub\vhosts\admin

AllowedPaths=c:\inetpub\vhosts\admin
```

- all WhizBase files (wbsp, sr, inc and ic) contained in virtual directory must use function $WBVDHR to refer to wwwroot directory of the site

```
Inside wwwroot directory:
WB_BaseName=/database/biblio.mdb

Inside virtual directory:
WB_BaseName=$wbvdhr{}/database/biblio.mdb
```

- to be able to use $WBVDHR function, server side variable VirtualDirHomeRef must be set in wbsp.ssc file.

6. CGI mode

Unlike previous versions that were either CGI program or hypertext pre-processor, WhizBase 5 and newer can be used both ways.
To use WhizBase in CGI mode you still need valid WBSP file, but instead of calling WBSP file directly (e.g. http://someserver/somefile.wbsp) you should insert in your URL engine file followed by full wbsp file path relative to the web server's root directory (e.g. http://someserver/cgi-bin/wbsp.exe/somefile.wbsp).

Due to security reasons we have disabled direct execution of theenginefile<BR>(e.g.http://someserver/cgi-bin/wbsp.exe?wb_basename=somebase.mdb&wb_rcdset=sometable&wb_command=q will not work).

7. Test mode

Test mode is feature added in version 2000. Its main purpose is to check if your WWW server has all files needed by WBSP properly installed.
To start WhizBase Server Pages test mode first create wbsp file named test.wbsp with following data:

```
[FormFields]
wb_command=T
wb_mailserver=your mailserver (e.g. mail.yourdomain.com)
wb_errmail=your email address (e.g. webmaster@yourdomain.com)
```

and upload it to your server (either local or Internet). Then open it using your browser by typing it's URL into browser's address bar (e.g. http://www.yourdomain.com/test.wbsp)

During the test WhizBase will create few databases and tables (and delete them afterwards) to test drivers. It will also execute a simple JavaScript code (server-side script) to test scripting host and send a simple test mail to email address specified in wb_errmail using mail server specified in wb_mailserver. It will also show WBSP.ssc settings for virtual host.

WARNING: To test the server installation and settings, WBSP creates following database tables: WhizBaseTest.mdb, WBDBIII.dbf, WBDBIV.dbf, WBDB5.dbf, WBPDX3.db, WBPDX4.db, WBPDX5.db, WBFOX20.dbf, WBFOX25.dbf, WBFOX26.dbf, WBFOX30.dbf, WBTEXT, schema.ini and WBODBC. If the files with these names exist, by any chance, in a directory where test.wbsp is located, they will be deleted.

You DO NOT need registration key to run this test on your WWW server!

8. Escape characters

Due to WhizBase's unique nature and parsing methods it is highly recommended not to use characters that are specific for WhizBase syntax in ordinary text not related to

WhizBase (e.g. using square brackets [] inside WhizBase report function can cause syntax error). Therefore we included escape sequences that can be used instead of those characters. To make WHizBase interpret these sequences and replace them with proper content, variable WB_UseEscapes must be set to True.

Character	Escape sequence
$	&$;
|	&|;
{	&{;
}	&};
[	&[;
]	&];

9. Comments

WhizBase version 6 adds support for comments. To avoid mistakes and removing parts of programming code written in other programming techniques and/or languages WhizBase uses hash and asterisk characters for opening and asterisk and hash characters for closing of the comments:

e.g.
#* This text will not be processed by WhizBase,
nor it will be sent to the browser.
*#

Multiline comments are supported in the body of the reports and subreports (below <!--WB_BeginTemplate-->), and in included files ($WBINC and $WBRINC).
If you want to **comment the variables in the configuration section(s)** use the single hash (without asterisk) at the beginning of the row containing the variable:

e.g.
#WB_TempName=$default$
the line above will be completely ignored by WhizBase

or use #* and *# in the single row:

e.g.
WB_Query=ID>1 #* and Year < 2001 *# and ID<100
the line above will be processed to:
WB_Query=ID>1 and ID<100

Beside documenting purposes comments can help you in debugging your WhizBase code - simply comment the part you suspect that generates the error and test the page again.

10. Subroutines

WhizBase version 6 added support for subroutines. Subroutines are code snippets written inside <!--WB_BeginSub_*subname*--> and <!--WB_EndSub--> where *subname* is the name that will be used for calling the subroutine using $WBSUB function and it must be unique in the scope of the main wbsp file and all sub reports and include files called from main wbsp file.

e.g.

```
<!--WB_BeginSub_diskarea-->
$wbformat[$wbcalc[$wbgetv[r]^2*3.1415926535897932384626433832795]|#.000
]
<!--WB_EndSub-->
```

Multiline Subroutines are supported in the body of the reports and sub reports (below <!--WB_BeginTemplate-->) and anywhere in included files ($WBINC and $WBRINC).
Please **do not** place subroutines **in the configuration section(s)** because it will have no effects.

11. Exceptions

WhizBase function $WBINC can not accept other WhizBase functions as arguments, but there is a similar function - $WBRINC that can be used when you need to process the function's arguments.

12. Path rules

WBSP engine has it's own path rules:

- All relative paths are relative to the location of the current WBSP file.
- Root dir is represented by slash (/) character and it represents document root (wwwroot) directory of current virtual host, and not root directory of current disk.
- In order to access files on same disk but located above the document root directory as well as to access files located on other disks or computers, developer must use absolute paths.
  To enable usage of absolute paths, AbsolutePath variable must be set to true in server configuration section for current virtual host in wbsp.ssc file. If this variable is not set to true, an error will be generated when WBSP receives a reference to an absolute path.

Path rules apply to all WhizBase variables, sections and functions that use external files. Here are some examples:

```
[Include]
/shopping.inc
```

```
../dbsettings/admin.inc
[FormFields]
wb_basename=\\DatabaseSrv\user23\access\shopping.mdb
wb_logFile=c:\logs\user23\wbsp.log
<!--BeginTemplate-->
<html><body><p>
$wbinc[/globaltop.htm]
<!--WB_BeginDetail-->
Your registration code is:<br>
$wbrun[c:\scripts\user23\registration.js
|JavaScript|MkRegCode($wbv{CopyID})]
<!--WB_EndDetail-->
$wbinc[shoppingfooter.htm]
</body></html>
```

Lines shown in red color contain file names with absolute path.

13. Installation

This install guide will help you manually install and configure WhizBase on your Windows 9x/Me/NT/2000/XP/2003 web servers.

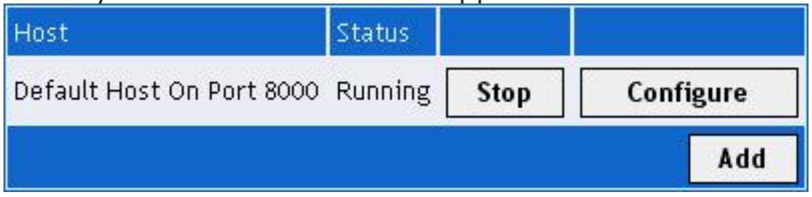
This guide provides manual installation support for:
Internet Information Server 4,5,6
Windows 2008 and IIS 7
Apache
Xitami
OmniHTTPd 2.0b1 and up
Netscape Servers
Abyss Web Server

WhizBase can be used either as a scripting engine or as a CGI program.
Before applying the server specific instructions, WhizBase files should be installed using the executable installer. Please read file **installwhizbase.pdf**.

13.1 Installing WhizBase on Windows with Abyss Web Server

This section contains notes and hints specific to Abyss Web Server. To download Abyss Web Server installation package please visit http://www.aprelium.com/abyssws/download.php.

- Open Abyss Web Server's console
- In the Hosts table, press Configure in the row corresponding to the host to which you want to add WBSP support

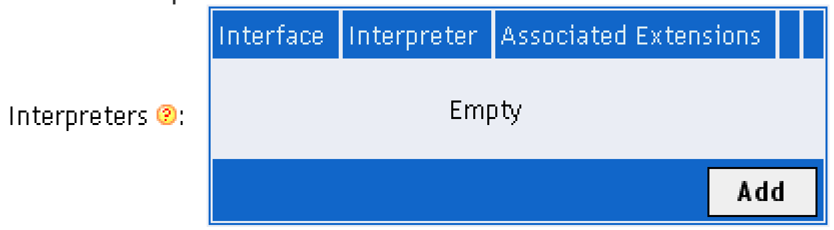


Host	Status		
Default Host On Port 8000	Running	Stop	Configure
			Add

- Click Scripting Parameters icon

  Scripting Parameters
- Check Enable Scripts Execution

  ✓ Enable Scripts Execution
- In the Interpreters table click Add button

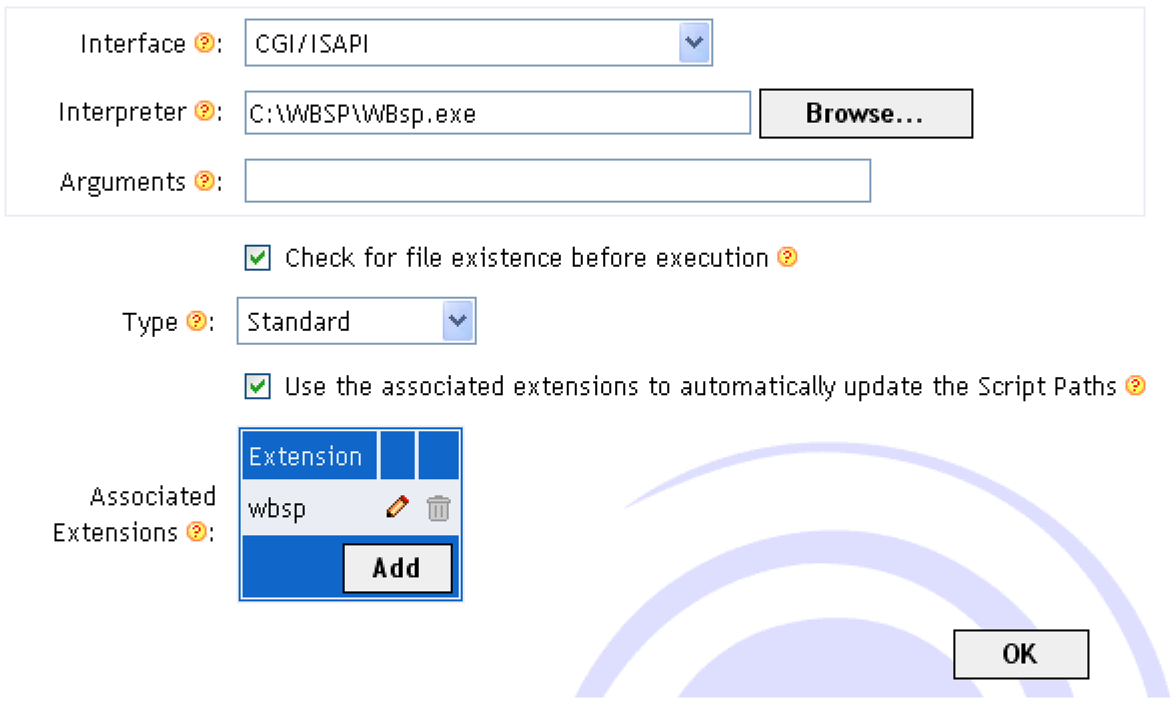


Interpreters:

Interface	Interpreter	Associated Extensions		
		Empty		

Add

- and set the parameters as shown bellow



Interface:	CGI/ISAPI
Interpreter:	C:\WBSP\WBsp.exe
Arguments:	

✓ Check for file existence before execution

Type: Standard

✓ Use the associated extensions to automatically update the Script Paths

Associated Extensions:

Extension		
wbsp		

Add

OK

- In the Interpreter field, press Browse..., go to the directory where you have installed WBSP, locate wbsp.exe, and click on it
  (in the example above we assume that you have installed WBSP in c:\wbsp\)
- Set Type to Standard
- Check Use the associated extensions to automatically update the Script Paths
- Press Add in the Associated Extensions table
- Enter wbsp in the Extension field and press OK
- Press OK
- Press OK in the Scripting Parameters dialog
- Click Index Files icon

  Index Files
- Press Add in the Index Files table
- Enter default.wbsp in the File Name field and press OK
- Click OK
- Press Restart to restart the server.

13.2 Installing WhizBase on Windows with Apache

This section contains notes and hints specific to Apache Web Server. To download Apache Web Server installation package please visit http://httpd.apache.org/download.cgi .

To set up WhizBase to work with Apache on Windows you need to stop the Apache server, and edit your srm.conf or httpd.conf to configure Apache to work with WBSP. Although there can be a few variations of configuring WBSP under Apache, these are simple enough to be used by the newcomer. Please consult the Apache Docs for further configuration directives.

To install WBSP insert these lines to your conf file (assuming that you have installed WBSP in c:\wbsp\):

```
<Directory "c:/wbsp/">
    AllowOverride all
    Options None
    Order allow,deny
    Allow from all
</Directory>
ScriptAlias /wbsp/ "c:/wbsp/"
AddType application/x-httpd-wbsp .wbsp
Action application/x-httpd-wbsp "/wbsp/wbsp.exe"
```

You must repeat from last two lines for each extension you want associated with WBSP scripts. (.wbsp,.sr,.aut,.ic and .inc are recommended.)

Please note that we have done our best to disable calling WBSP directly:
http://servername/wbsp/wbsp.exe?.....
so please do not change the .exe extension on wbsp.exe file.

As a further precaution, we recommend you change the "/wbsp/" ScriptAlias to something more random, to prevent any attempts to call your binary (like the Code Red scripts) for returning a response other than 404.

Remember when you have finished to restart the server, for example,
NET STOP APACHE
followed by
NET START APACHE

13.3 Windows NT/2000/2003 and IIS 4, 5, 6 or newer and PWS 4 on NT Workstation or W2K non server editions

To install WhizBase on an NT/2000/2003 Server running IIS 4, 5, 6 follow these instructions. Start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

Next, do the following:

- Under 'Home Directory', 'Virtual Directory', or 'Directory', click on the 'Configuration' button, and then enter the App Mappings tab.
- Click Add, and in the Executable box , type:
  c:\wbsp\wbsp.exe (assuming that you have installed WBSP in c:\wbsp\).
- In the Extension box , type the file name extension you want associated with WBSP scripts.
  Leave 'Method exclusions' blank, and check the Script engine checkbox. You do not need to check the 'check that file exists' box - because of a performance effects, WBSP itself will check that the script file exists and sort out authentication before processing the request.
  This means that you will get sensible 404 style error messages anyway but checking the 'check that file exists' box will slow the server a bit.
- You must repeat from 'Click Add...' for each extension you want associated with WBSP scripts. (.wbsp,.sr,.aut,.ic and .inc are recommended.)
- **For Windows 2003 Server and IIS 6 only:** Set web extension status to "Allowed" by opening "Web Service Extensions" and selecting "Add a new Web service extension...". Name the service (e.g. "WBSP Extension"), add full path to wbsp.exe to the "Required files" list, and check the "Set extension status to Allowed " checkbox. This step has to be done on W2003/IIS 6 even if you want to use WBSP as a CGI.

13.4 IIS 7

Please read file **installingiis.pdf**.

13.5 Installing WhizBase on Windows with Netscape servers.

To Install WhizBase:

- Make a file association from the command line (type the 2 following lines, assuming that you have installed WBSP in c:\wbsp\)
  assoc .wbsp=WBScript
  ftype WBScript=c:\wbsp\wbsp.exe %1 %*
- In the Netscape Enterprise Administration Server create a dummy shellcgi directory and remove it just after (this step creates 5 important lines in obj.conf and allow the web server to handle shellcgi scripts)
- In the Netscape Enterprise Administration Server create a new mime type (Category:type,Content-Type:magnus-internal/shellcgi,File Suffix:wbsp)

Do it for each web server instance you want WBSP to run

13.6 Installing WhizBase on Windows with OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd 2.0b1 and up for Windows.

- Install OmniHTTPd server.

- Right click on the blue OmniHTTPd icon in the system tray and select 'Properties'
- Click on 'Web Server Global Settings'
- On the 'External ' tab, enter:
  virtual = .wbsp | actual = c:\wbsp\wbsp.exe (assuming that you have installed WBSP in c:\wbsp\)
  and use the Add button.
- On the Mime tab , enter:
  virtual = wwwserver/stdcgi | actual = .wbsp
  and use the Add button.
- Click 'OK'

Repeat steps 4,5 and 6 for each extension you want to associate with WBSP (.wbsp,.sr,.aut,.ic and .inc are recommended).

13.7 Installing WhizBase on Windows with Xitami

This section contains notes and hints specific to Xitami - powerful multiplatform Open Source web server. To download Xitami installation package please visit http://www.xitami.com/download.htm.

- Make sure the webserver is running, and point your browser to xitamis admin console (usually http://127.0.0.1/admin), and click on Configuration.
- Navigate to the Filters, and put the extension which wbsp should parse (i.e. .wbsp) into the field File extensions (.xxx).
- In Filter command or script put the path and name of your wbsp executable i.e. c:\wbsp\wbsp.exe (assuming that you have installed WBSP in c:\wbsp\).
- Press the 'Save' icon.
- Repeat last three steps for each extension you want associated with WBSP scripts (.wbsp,.sr,.aut,.ic and .inc are recommended).

To add WBSP filter manually please follow these steps:

- In directory where Xitami is installed create file defaults.cfg
- Add following lines to the file:
  ```
  [Server]
  priority=1

  [Server]
  Default3=default.wbsp
  [Filter]
  #assuming that WBSP.exe is located in c:\wbsp\
  .wbsp=c:\WBSP\wbsp.exe
  .sr=c:\WBSP\wbsp.exe
  .aut=c:\WBSP\wbsp.exe
  .ic=c:\WBSP\wbsp.exe
  .inc=c:\WBSP\wbsp.exe
  ```
- Restart the server

14. WBSP Server Side Configuration (wbsp.ssc)

WBSP server side configuration file is text file located in the same directory with engine file wbsp.exe. Both these files should not be accessible using HTTP protocol (they should be located above wwwroot directory of the web server or, if WSBP is used as CGI, in a directory without read access).
WBSP.SSC file has three main sections and a separate configuration section for every virtual host on the server:

- Registration
- Servers (virtual hosts)
- Default

14.1 Default section

Default section contains configuration that will be applied to any virtual host that does not have its own configuration section . Here's an example:

```
[default]
DisableWB=F
AbsolutePath=Off
Upload=Off
FileCommands=Off
Execute=JavaScript
HiddenEnvVars=PATH,COMP*,System*,Win*
TimeOutSec=10
DefaultDocument=index.wbsp
ADOConnectionString=F
ActivateCGIByExt=wbsp
CGISecurityString=LhBv3KP
MaxInstances=30
HideDocuments=sr,ic,inc,aut,mdb,dbf
```

The next example will disable usage of WBSP on all virtual hosts, but those that are defined in Servers section:

```
[default]
DisableWB=T
```

14.2 Registration section

The registration section always contains a single value of registration key for entire server (server license - ServerKey variable). If valid server registration key exists, virtual hosts do not need their own registration keys.

[Registration]
ServerKey=SXHZSNKIKDIJSHIJKL

14.3 Server configuration section

This section contains specific server side configuration for all aliases of a single virtual host defined in Servers section. It has to be created for every virtual host that needs separate configuration different from default settings. Here's an example:

```
[wbsp.com]
DisableWB=F
AbsolutePath=On
Upload=On
FileCommands=On
Execute=JavaScript,VBScript,JScript
HiddenEnvVars=PATH,COMP*,System*
TimeOutSec=20
DefaultDocument=index.wbsp
ADOConnectionString=T
ActivateCGIByExt=wbsp
CGISecurityString=LhBv3KP
MaxInstances=0
HideDocuments=sr,aut,inc,ic,mdb
ScriptTimeOutSec=60

[dsd.ba]
DisableWB=F
AbsolutePath=Off
Upload=On
FileCommands=Off
Execute=VBScript
HiddenEnvVars=PATH,COMP*,System*,Win*
TimeOutSec=10
DefaultDocument=default.wbsp
ADOConnectionString=F
ActivateCGIByExt=wbsp
CGISecurityString=LhBv3KP
MaxInstances=10
```

14.4 Servers (virtual hosts) section

This section contains definition of aliases used to access single virtual host. Format of the definition is

```
virtualhostname=alias1,alias2,alias3,...,aliasN
```

If some virtual hosts do not have any aliases then they will be listed only by virtualhostname. The purpose of this section is to make WBSP engine use the same configuration section for a single virtual host regardless of an alias used to access it. Aliases are needed only if virtual host has more than one domain pointing to the same web, but they can be used also for single domain sites.

```
[Servers]
whizbase.com=wbsp.com
myfirstclient.com=myfirstclient.net,myfirstclient.org,myfirstclient.ca
dsd.ba=www.dsd.ba,members.dsd.ba
dws.ba
```

The value marked with blue in example above is not required (because all aliases point to the same domain - dsd.ba), but it will not produce error of any kind.
The red colored values in example are used to define the server side configuration section for specific virtual host. As you can see, if virtual host has only one domain used to access it, then simply add that domain to the list, without specifying any aliases (as it is the case with domain dws.ba in example above).

14.5 Server configuration variables

The purpose of server side configuration is to provide more control of the hosted sites to the server owner. All these variables are used to increase security of the server, by limiting developers access rights to certain server data (environmental variables, paths above virtual hosts wwwroot directory, writing files during run-time), or by limiting the ability to use some WBSP functionalities (scripting host, file upload, full definition of ADO connection string, limiting number of instances of the WBSP engine that can be used simultaneously by single virtual host or even disabling WBSP on certain virtual hosts) or by strictly defining the rules by which WBSP can be used in CGI mode.

Server-side configuration variables :
AbsolutePath
ActivateCGIByExt
ADOConnectionString
CGISecurityString
DefaultDocument
Developer
DisableWB
Execute
FileCommands
HideDocuments
HiddenEnvVars
MaxInstances
RegCode
ScriptTimeOutSec
SessionFile
SessionIdleTime
TimeOutSec
Upload

14.5.1 AbsolutePath

```
AbsolutePath=On/Off
```

If this variable is set to True, the WBSP engine will allow owner of the site (author, user that creates WBSP files and uploads them to server) to use absolute path names, i.e. to access files outside his wwwroot directory, files located on other disks or even other network computers.
Setting this value to True can have great advantages on intranet (or if you use entire server exclusively) **but it is not wise to set this to True on shared web server with many users**. **Default value is False**.

14.5.2 ActivateCGIByExt

```
ActivateCGIByExt=fileextension
```

This variable has no effect on the WhizBase it is installed as a scripting engine. It defines the file extension required for activating the WhizBase in the CGI mode. The **default value is wbsp** (the WhizBase can be activated in the CGI mode only by a file with .wbsp extension), and if, for any reason, there is a need for disabling this security measure set this variable to $empty$.

14.5.3 ADOConnectionString

```
ADOConnectionString=On/Off
```

If this variable is set to True then WBSP will accept any valid ADO connection string in WB_DBObject . If it is set to False then WB_DBObject will accept only predefined values D35, D36, A35, A40 and A07. **Default value is off**.

14.5.4 AllowedPaths

```
AllowedPaths=path list
```

This variable contains a comma-separated list of absolute paths that are allowed for use by the site. It is similar to setting AbsolutePath to true, but limited only to specified paths. Path list can contain paths outside site's wwwroot directory, paths located on other disks or even other network computers.
It is very useful on shared servers where web site needs an access to files outside wwwroot directory but you can not set AbsolutePath to true due to security reasons.

14.5.5 CacheDir

```
CacheDir= path to cache directory
```

This variable contains a reference to the directory where WhizBase stores files generated by function $WBCACHE.

Example:

```
CacheDir=/cache/
```

14.5.6 CGISecurityString

```
CGISecurityString=anystring
```

This variable has no effect on the WhizBase when it is installed as a scripting engine. It contains the administrator-defined case-sensitive string that must match the beginning of the file used to activate the WhizBase in the CGI mode. It is used when for some reason it is not suitable to use ActivateCGIByExt. **There is no default value** for this variable.

Example:

```
CGISecurityString=LnTW34FxD

Valid WBSP file:
LnTW34FxD
[FormFields]
wb_command=...

Invalid WBSP file:
[FormFields]
wb_command=...
```

14.5.7 DefaultDocument

```
DefaultDocument=filename
```

This variable contains the file name of default WBSP file defined in your server's configuration (e.g. index.html, index.htm, index.wbsp). Some web server software can have problems recognizing the default document. If that is the case with your web server software than setting this variable will solve the problem. **Default value is default.wbsp**.

14.5.8 Developer

```
Developer=registered developer name
```

This variable contains the name of the registered developer used in combination with RegCode variable. If valid combination of these two variables exists, virtual hosts do not need their own registration keys.

14.5.9 DisableWB

```
DisableWB=T/F
```

If this variable is set to True, the WBSP engine will not process WBSP files located on that specific virtual host, and error will be generated. **Default value is False**.

14.5.10 Execute

```
Execute=languagelist
```

This variable contains the comma-separated list of scripting languages that are allowed to be used on the virtual server. If this variable is empty then server-side scripting functionality of WBSP engine (using $wbrun function) will be disabled on that virtual host. **There is no default value** for this variable, so if administrator does not set the list, scripting will be disabled.

14.5.11 FileCommands

```
FileCommands=On/Off
```

If this variable is set to True, the WBSP will process files that include writing and/or deleting files on server. It is strongly recommended not to set this variable to True together with AbsolutePath set to true, unless webmaster of virtual server is experienced WBSP developer. **Default value is Off**.

14.5.12 HiddenEnvVars

```
HiddenEnvVars=envlist
```

This variable contains the comma-separated list of environment variables that will be hidden, i.e. author will not be able to use $wbe function with those variables. **Default value is ***, what means that all environment variables will be hidden. However, some of the environment variables are very important for developing advanced WBSP-powered sites, so it is good idea to hide only those environment variables that can be abused (e.g. PATH, COMP*, SYSTEM*, etc.)

14.5.13 HideDocuments

```
HideDocuments=extlist
```

This variable contains the comma-separated list of file extensions that will be hidden, i.e. if you configure server to handle files with those extensions by WBSP, WBSP will return HTTP error 404 (file not found). **Default value is sr,aut,ic,inc**, what means that include files (both inc and ic), subreports and authorization files will be hidden, as long as your server is configured to handle them with WBSP.exe same as files with wbsp extension.

14.5.14 MaxInstances

```
MaxInstances=number
```

This variable defines the maximum number of concurrent instances of the engine for the specific virtual server. If this number is exceeded the error will be generated. The **default value is 0** (unlimited number of instances).

14.5.15 RegCode

```
RegCode=Registration Key
```

This variable contains the registration code used in combination with Developer variable. If valid combination of these two variables exists, virtual hosts do not need their own registration keys.

14.5.16 ScriptTimeOutSec

```
ScriptTimeOutSec=seconds
```

This variable contains the default value for WB_TimeOut variable for specific virtual host. **Default value is 30**.

14.5.17 SessionFile

```
SessionFile=/DVKOTWF/sessionfile.ssc
```

This variable contains the full path to the file where WhizBase will save all session-related data. **Default value is /sessions.ssc**.

14.5.18 SessionIdleTime

```
SessionIdleTime=3600
```

This variable contains the number of seconds of continuous idle time that must pass before WhizBase clears the session due to inactivity. **Default value is 1800**.

14.5.19 TimeOutSec

```
TimeOutSec=seconds
```

This variable contains the number of seconds that WBSP engine will wait for server-side script ($wbrun) to execute before it terminates the execution. **Default value is 10**.

14.5.20 Upload

```
Upload=On/Off
```

If this variable is set to True, the WBSP engine will process WBSP files that upload files to server. To learn more about this, please read the explanation for usage of WB_AllowMultipart variable in section FormFields and explanation for Upload section of WBSP file. WBSP enables author to control the type, size and location of uploaded files. However, server administrator can disable file-uploading by setting this variable off, even when author of the site has created proper WBSP files. **Default value is Off**.

14.5.21 UseServerKey

```
UseServerKey=On/Off
```

If this variable is set to False, the WBSP engine will require separate registration key for WhizBase instance used by the site even if computer has valid WhizBase server license. Useful in situations where owner of the server wants to limit usage of existing WhizBase server license (e.g. to web hosting clients that pay additional fee to the server's owner for being able to use WhizBase). **Default value is True**.

14.5.22 VirtualDirHomeRef

```
VirtualDirHomeRef=path to wwwroot directory
```

This variable contains a reference to wwwroot (home) directory of the site, the value that will be returned by function $WBVDHR. It is used to provide reference to site's wwwroot directory for files located in virtual directory.

15. Getting started

Although WBSP on your server can be configured to process files with any extension, to make usage of this tutorial easier we will assume that .wbsp extension is used. For development purposes you will probably install WBSP locally, and this manual has installation instructions for many existing web servers including some very good freeware. If you have any problems with installing WBSP, please do not hesitate to contact us .

15.1 Your first WBSP page

Create file hello.wbsp and place it in document root of your web server with following content:

```
<html>
    <body>
    Hello visitor from $wbe[remote_addr]!<br>
    Your language is $wbe[HTTP_ACCEPT_LANGUAGE].
    </body>
</html>
```

Access the file with your web server's URL followed by /hello.wbsp. On your local web server (located on same computer as your browser) the address should be http://localhost/hello.wbsp or http://127.0.0.1/hello.wbsp. If WBSP is installed correctly and the server is configured correctly, your browser should receive something like this:

```
<html>
    <body>
    Hello visitor from 127.0.0.1!<br>
    Your language is en-us.
    </body>
</html>
```

and it would be displayed like this:

```
Hello visitor from 127.0.0.1!
Your language is en-us.
```

If this example did not output anything when you tried it or if it started the download or displayed the file unchanged (with $wb elements included), then the server probably is not configured properly.

15.2 Something useful

This is a simple solution for common task - determining the visitor's preferred language and redirecting him to proper content:

```
[FormFields]
WB_Command=R
WB_Redirect=http://$wbe{server_name}/$wbif{$wbindof{$wbe{HTTP_ACCEPT_LA
NGUAGE}|de}>0|ger|eng}
```

If visitor's browser is configured to accept content in German language the WBSP will redirect visitor to directory ger (with content in German language) in servers root, and if it does not then visitor will be redirected to directory eng (with content in English language).

15.3 Simple database example

What makes WBSP really different from programming languages and other hypertext preprocessors is it's way of dealing with databases. WBSP can work with wide range of databases that are supported by MS Jet Engine, DAO and ADO objects and ODBC. In our examples we use MS Access database biblio.mdb that can be downloaded from our server.

Download file bibliomdb.zip from our server and unpack it in your document root of your web server and create file biblio.wbsp in same directory with following content:

```
[FormFields]
WB_Command=q
WB_Basename=biblio.mdb
WB_Rcdset=titles
WB_TempName=$default$
```

Access the file with your web server's URL followed by /biblio.wbsp. On your local web server (located on same computer as your browser) the address should be http://localhost/biblio.wbsp or http://127.0.0.1/biblio.wbsp. If WBSP is installed correctly and the server is configured correctly, your browser should receive something like this:

Title	Year Published	ISBN	PubID	AU_ID
McGraw-Hill's Encyclopedia of Networking & Telecommunications	2001	0072120053	10	10
Microsoft SMS Installer	2000	0072124474	10	9
Windows 2000 Iis 5.0 : A Beginner's Guide	2001	0072133724	9	9
Windows Nt Security Handbook	1996	0078822408	10	11
Microsoft Internet Information Server 4: the Complete Reference	1998	0078824575	10	10
Non-Designer's Scan and Print Book, The	1999	0201353946	1	2
Real World Adobe InDesign 1.5	2000	0201354780	1	1
HTML 4 for the World Wide Web: Visual Quickstart Guide	2000	0201354934	1	6
Real World Freehand 7	1997	0201688875	1	1
Netscape 3 for Macintosh Visual Quickstart Guide	1996	0201694085	1	6
Kai's Power Tools 3 for Windows Visual Quickstart Guide	1997	0201696681	1	2
InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide	2000	0201710366	1	2
Fireworks 4 for Windows and Macintosh Visual	2001	0201731339	1	2

Quickstart Guide			
Macromedia FreeHand 10 for Windows and Macintosh: Visual QuickStart Guide	2001	0201749653 1	2
Real World FreeHand 5.0/5.5	1996	0201883600 4	1
Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours	2000	0672318830 12	13
Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours	2000	0672320428 12	13
Photoshop 6 Photo-Retouching Secrets	2001	0735711461 3	3
www.color	2000	0823058573 8	7
Www.Layout : Effective Design and Layout for the World Wide Web	2001	0823058581 8	8

1 2
First page Next page Last page

As you can see WBSP has created a default template for displaying all fields from table Titles in biblio.mdb. Since we did not define maximum number of records per page (WB_MaxRec) it used the default value of 20 records, and generated links to pages containing further records. It also displayed the WBSP logo at the bottom of the page.

Well, it is nice and easy solution but is not very useful, isn't it?

To make really useful example we will need an ordinary HTML page like this:

```
<html>
    <head>
    <title>Simple database example</title>
    </head>
    <body>

    </body>
</html>
```

The second step would be to add some WBSP lines (marked red):

```
[FormFields]
WB_Command=q
WB_Basename=biblio.mdb
WB_Rcdset=titles
wb_showlogo=F
<!--WB_BeginTemplate-->
<html>
    <head>
    <title>Simple database example</title>
    </head>
    <body>
    $wbdetail[t]
    </body>
</html>
```

If you open this page with your browser by typing it's address on your local server, you will see very little changes compared to first example - WBSP logo is not included in a page because we set WB_ShowLogo to false.

Now the third and final step for this example - setting the style for generated table (marked with blue):

```
[FormFields]
WB_Command=q
WB_Basename=biblio.mdb
WB_Rcdset=titles
wb_showlogo=F
[MsgAndLbl]
WB_Style=font-family:verdana;font-size:12px;color:#CC0000;
<!--WB_BeginTemplate-->
<html>
    <head>
    <style>
    .wbspttbl{
    border:1px solid #000000;
    font-family:verdana;
    font-size:12px;
    border-collapse:collapse;
    border-spacing:0px;
    }
    .wbspthdr{
    background-color:#CC0000;
    border:1px solid #000000;
    color:#C0C0C0;
    }
    .wbsptrow{
    background-color:#FFCC00;
    border:1px solid #000000;
    color:#0000CC;
    }
    </style>
    <title>Simple database example</title>
    </head>
    <body>
    $wbdetail[t]
    </body>
</html>
```

The result in the browser now should look something like this:

Title	Year Published	ISBN	PubID	AU_ID
McGraw-Hill's Encyclopedia of Networking & Telecommunications	2001	0072120053	10	10
Microsoft SMS Installer	2000	0072124474	10	9
Windows 2000 Iis 5.0 : A Beginner's Guide	2001	0072133724	9	9
Windows Nt Security Handbook	1996	0078822408	10	11
Microsoft Internet Information Server 4: the	1998	0078824575	10	10

Complete Reference				
Non-Designer's Scan and Print Book, The	1999	0201353946	1	2
Real World Adobe InDesign 1.5	2000	0201354780	1	1
HTML 4 for the World Wide Web: Visual Quickstart Guide	2000	0201354934	1	6
Real World Freehand 7	1997	0201688875	1	1
Netscape 3 for Macintosh Visual Quickstart Guide	1996	0201694085	1	6
Kai's Power Tools 3 for Windows Visual Quickstart Guide	1997	0201696681	1	2
InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide	2000	0201710366	1	2
Fireworks 4 for Windows and Macintosh Visual Quickstart Guide	2001	0201731339	1	2
Macromedia FreeHand 10 for Windows and Macintosh: Visual QuickStart Guide	2001	0201749653	1	2
Real World FreeHand 5.0/5.5	1996	0201883600	4	1
Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours	2000	0672318830	12	13
Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours	2000	0672320428	12	13
Photoshop 6 Photo-Retouching Secrets	2001	0735711461	3	3
www.color	2000	0823058573	8	7
Www.Layout : Effective Design and Layout for the World Wide Web	2001	0823058581	8	8

1 2
First page Next page Last page

As you can see we used css classes wbsptbl, wbspthdr and wbsptrow to format the table and for formatting navigation links we used WBSP variable WB_Style.

16. How to...

This section contains step-by-step instructions on how to complete basic tasks using WhizBase.

Displaying records from database table
Uploading files

16.1 The simplest database example

1. Open new file in your favorite text editor
2. Type the following code

```
[FormFields]
wb_basename=biblio.mdb
wb_rcdset=titles
```

```
wb_command=q
wb_tempname=$default$
```

3. Save page as wbsp file (some_name.wbsp) and upload it to the server together with biblio.mdb file

16.2 1-2-3 example

1. Open your plain HTML file in your favorite editor and position the cursor where you want to display your database contents
2. Type the following code

```
#*
[FormFields]
wb_basename=biblio.mdb
wb_rcdset=titles
wb_command=q
*#
$wbdetail[T]
```

3. Save page as wbsp file (some_name.wbsp) and upload it to the server together with biblio.mdb file

16.3 Displaying formatted records from database table

This simple example uses few basic WhizBase variables and functions, but it includes HTML tag **STYLE** to define classes wbspttbl, wbspthdr and wbsptrow which are used to format table containing records from database.

```
[FormFields]
wb_basename=biblio.mdb
wb_rcdset=Titles
WB_Command=Q
WB_MaxRec=10
<!--WB_BeginTemplate-->
<html>
<head>
<style>
.wbspttbl{
border:1px solid #000000;
font-family:verdana;
font-size:12px;
border-collapse:collapse;
border-spacing:0px;
}
.wbspthdr{
background-color:#CC0000;
border:1px solid #000000;
color:#C0C0C0;
}
.wbsptrow{
background-color:#FFCC00;
border:1px solid #000000;
color:#0000CC;
}
</style>
```

```
<title>Simple database example</title>
</head>
<body>
$wbdetail[t]
</body>
</html>
```

16.4 Displaying records from joined tables

Table "Titles" does not contain neither author's nor publisher's name but their respective ID numbers. In this example we joined table "Titles" with tables "Authors" and "Publishers" (using WhizBase variable WB_RcdSet) and used WB_DBFlds to define fields we want to include in resulting recordset. We also sorted records by year in descending order.

```
[FormFields]
wb_basename=biblio.mdb
WB_RcdSet=(Authors inner join titles on authors.au_id=titles.au_id)
inner join publishers on publishers.pubid=titles.pubid
WB_DBFlds=Title,[Year Published],ISBN,Authors.Name as
Author,publishers.Name as Publisher
WB_Command=Q
WB_MaxRec=10
WB_Order=[Year published] desc
<!--WB_BeginTemplate-->
<html>
<head>
<style>
.wbspttbl{
border:1px solid #000000;
font-family:verdana;
font-size:12px;
border-collapse:collapse;
border-spacing:0px;
}
.wbspthdr{
background-color:#CC0000;
border:1px solid #000000;
color:#C0C0C0;
}
.wbsptrow{
background-color:#FFCC00;
border:1px solid #000000;
color:#0000CC;
}
</style>
<title>Simple database example</title>
</head>
<body>
$wbdetail[t]
</body>
</html>
```

16.5 Creating search form

To enable search though the recordset we will add the search form and use joined tables so our visitor can search for author's and publisher's name instead of their ID. Please note that form fields with name starting with WBF_ are used to carry a value for specific database field. The name of the recordset field is specified **after** WBF_ prefix and it has to be **exactly the same as in the database**.

```
[FormFields]
wb_basename=biblio.mdb
WB_RcdSet=(select Title,[Year Published],ISBN,Authors.Name as
Author,publishers.Name as Publisher,Authors.Name,publishers.Name from
(Authors inner join titles on authors.au_id=titles.au_id) inner join
publishers on publishers.pubid=titles.pubid)
WB_Command=Q
WB_MaxRec=10
WB_Order=[Year published] desc
<!--WB_BeginTemplate-->
<html>
<head>
<style>
.wbspttbl{
border:1px solid #000000;
font-family:verdana;
font-size:12px;
border-collapse:collapse;
border-spacing:0px;
}
.wbspthdr{
background-color:#CC0000;
border:1px solid #000000;
color:#C0C0C0;
}
.wbsptrow{
background-color:#FFCC00;
border:1px solid #000000;
color:#0000CC;
}
input, select, label{
width:150px;
font-family:verdana;
font-size:11px;
}
</style>
<title>Simple database example</title>
</head>
<body>
<form action="$wbe[script_name]"> #*comment: $wbe[script_name] will
return the name of current wbsp file*#
<label for="year">Select year of publishing: </label>
<select name="WBF_Year published" size="1" id="year">#*comment:
WBF_Year published form field will carry the search string for
recordset field Year published*#
<option value="">ignore</option>
<option value="1996">1996</option>
<option value="1997">1997</option>
<option value="1998">1998</option>
<option value="1999">1999</option>
```

```
<option value="2000">2000</option>
<option value="2001">2001</option>
</select><br>
<label for="title">Title: </label>
<input type="text" size="20" id="title" name="wbf_title"><br>#*comment:
WBF_title form field will carry the search string for recordset field
title*#
<label for="author">Author: </label>
<input type="text" size="20" id="author"
name="wbf_author"><br>#*comment: WBF_author form field will carry the
search string for recordset field author*#
<label for="publisher">Publisher: </label>
<input type="text" size="20" id="publisher"
name="wbf_publisher"><br>#*comment: WBF_publisher form field will carry
the search string for recordset field publisher*#
<label for="AO">Return records that: </label>
<select name="WB_AndOr" size="1" id="AO">
<option value="AND">match all conditions</option>
<option value="OR">match any of the conditions</option>
</select><br>#*comment: Value of WB_AndOr form field will define if
WhizBase will search for records that meet all conditions (And) or any
condition (Or)*#
<input type="submit" name="sButt" value="Search">
</form>
$wbdetail[t]
</body>
</html>
```

16.6 Displaying records in XML format

In other examples we used $WBDetail function, but in some cases there is a need for placing the database field in a specific location on your report, or even not using HTML at all. In that case we do not use $WBDetail function but some (or all) of database field functions $WBF, $WBFF, $WBFC, $WBFU and $WBRF. We also use <!--WB_BeginDetail--> and <!--WB_EndDetail--> comments to define the detail section (section that will be repeated for every record). In this example we will modify the code so it will return recordset in XML format:

```
[FormFields]
wb_basename=biblio.mdb
WB_RcdSet=(Authors inner join titles on authors.au_id=titles.au_id)
inner join publishers on publishers.pubid=titles.pubid
WB_DBFlds=Title,[Year Published],ISBN,Authors.Name as
Author,publishers.Name as Publisher
WB_Command=Q
WB_MaxRec=$all$
WB_Order=[Year published] desc
WB_ContentType=text/xml
<!--WB_BeginTemplate--><?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
generated="2008-10-02T12:24:19"><!--WB_BeginDetail-->
<Titles>
<Title>$wbmrepl[$wbf[Title]|',&|&apos;,&amp;]</Title>
<Year_x0020_Published>$wbf[year published]</Year_x0020_Published>
<ISBN>$wbf[isbn]</ISBN>
<Author>$wbf[Author]</Author>
```

```
<Publisher>$wbf[Publisher]</Publisher>
</Titles><!--WB_EndDetail-->
</dataroot>
```

We modified WB_MaxRec to special value all (this will show all records from recordset in a single report page) and set the WB_ContentType variable to text/xml. Note that we also had to use $WBMREPL function to replace apostrophe and ampersand characters in field "Title" due to syntax rules of XML.

16.7 Simple upload example

The upload process functions like this – visitor sends the file to the server using multipart form (enctype="multipart/form-data"). WhizBase first checks if file upload is enabled in current virtual host setting (file wbsp.ssc). If it is not, WBSP will generate error and terminate processing of the WBSP file. If file upload is enabled, it checks if the WB_AllowMultipart variable is set to True in the current WBSP file, and if it is, WhizBase reads settings from Upload section of same file and process uploaded files (every multipart form can upload more than one file at the time). When WhizBase receives all files it saves them following the instructions from Upload section and replace values of the multipart fields (fields that had contained the uploaded files) with URL of saved file, and then it process the WBSP file as any other. Multipart form must use **POST method**.

Here is a very simple upload example (please read the comments in source code below):

```
[FormFields]
WB_AllowMultipart=T #*Allow this wbsp file to accept uploaded files*#
WB_Command=R
[Upload]
WB_Disallow=![jpg,gif] #*disallow uploading of any file type but jpg
and gif*#
WB_UploadDir=/upload/ #*destination directory name (where uploaded
files will be saved)*#
WB_BaseURL=/upload/ #*path that will be added to file name to reference
the URL of the uploaded file*#
WB_Overwrite=F #*if set to false WhizBase will generate a unique name
for newly uploaded file if file with same name already exists in upload
directory*#
WB_MaxFSize=102400 #*Maximum size in bytes for single file that will be
accepted by WhizBase*#
WB_UploadLog=upload.log #*name of log file where WhizBase will keep
track of all uploads received by this file*#
<!--WB_BeginTemplate-->
<html>
<head>
<title>$wbif["$wbv[image]"=""|Upload file|File uploaded]</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
$wbif["$wbv[image]"="" #*check if file has been uploaded*#
|
#*If not ($wbv[image] is empty) show the upload form*#
<form action="$wbe[script_name]" method="post" ENCTYPE="multipart/form-
data">
Select file (*.jpg;*.gif - max. 100KB): <input type="file" name="image"
size="20"> <input type="submit" name="sBtt" value="Upload">
</form>
|
#*If file is uploaded ($wbv[image] contains URL to file) then display a
link to uploaded file*#
Open uploaded image<br><a target="_blank">$wbv[image]</a>
]
</body>
</html>
```

16.8 Advanced upload example

This example adds the record to "Titles" table including a reference to uploaded cover page image. Here is the code for page containing the form:

```
<!--
[FormFields]
WB_Command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Add Titles</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
<meta http-equiv="Content-Language" content="en-us">
</head>
<body>
<form action="titlesAdd.wbsp" method="POST" align="center" id="addForm"
ENCTYPE="multipart/form-data">
<table border="0" cellpadding="0" cellspacing="0">
<tr>
<td>Title</td>
<td align="right"><input type="text" name="WBF_Title" value
size="20"></td>
</tr>
<tr>
<td>Year Published&nbsp; </td>
<td align="right"><input type="text" name="WBF_Year Published" value
size="20"></td>
</tr>
<tr>
<td>ISBN</td>
<td align="right"><input type="text" name="WBF_ISBN" value
size="20"></td>
</tr>
<tr>
<td>Publisher </td>
<td align="right"><select name="WBF_PubID"
size="1">$wbsr[selpublisher.sr]</select></td>
</tr>
<tr>
<td>Author</td>
<td align="right"><select name="WBF_AU_ID"
size="1">$wbsr[selauthor.sr]</select></td>
</tr>
<tr>
```

```
<td>Book cover (image)</td>
<td align="right"><input type="file" name="WBF_imageURL"
size="20"></td>
</tr>
<tr>
<td>&nbsp;</td>
<td align="right"><input type="submit" name="sButt" value="Add new
title"></td>
</tr>
</table>
</form>
</body>
</html>
```

and titlesAdd.wbsp file should look something like this:

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_AllowMultipart=T
WB_Command=A
WB_Redirect=$wbe[http_referer]$wbif[$wbindof[$wbe[http_referer]|?]>0||?
wb_startrec=$wbv[sr]]
WB_RcdSet=Titles
[Upload]
WB_Disallow=![jpg,gif]
WB_UploadDir=/images/
WB_BaseURL=/images/
WB_Overwrite=F
WB_MaxFSize=24576
WB_UploadLog=upload.log
```

In this example, when visitor enters the values in all form fields (including the image file to be uploaded using WBF_ImageUrl) and submits the form, WBSP will accept the data and:

1. Check the Upload variable for current virtual host in wbsp.ssc file
2. If uploading is enabled WBSP engine will read the WB_AllowMultipart variable from WBSP file titlesAdd.wbsp
3. Since it exists and is set to True WBSP engine will process the uploaded file (e.g. mycover.gif)
4. First WBSP checks if the file size is less then or equal to the number specified in variable WB_MaxFSize (24576 bytes in this example)
5. If file size is OK, WBSP saves the file in directory specified in variable WB_UploadDir (directory images in servers document root)
6. If file with same name already exists it will generate unique file name (because WB_Overwrite is set to False) for uploaded file
7. Next WBSP changes the value of form variable WBF_ImageURL to WB_BaseURL+name of the saved file (in this example /images/mycover.gif)
8. WBSP writes the log record in log file specified in variable WB_UploadLog (in this example file upload.log located in same dir with file titlesAdd.wbsp)
9. After this WBSP engine will continue processing the WBSP file to complete the Add record command (specified by WB_Command=A) as it was ordinary form, with the exception of the value of **table field imageurl** which will have value **/images/mycover.gif** (relative URL of the uploaded file, and not the file itself).

17. WhizBase Report Template (WBSP file)

Unlike the previous versions of WhizBase when report template had to be invoked using EXE CGIprogram(e.g.<a>) with WhizBase Server Pages any file with extension .wbsp will start WBSP engine which will open the file and start processing it. WhizBase report templates (.wbsp files) have the same structure as ordinary documents ot selected type (HTML, XML, XHML, RTF, JS, TXT, etc.) extended with WhizBase configuration variables and WhizBase report tags and functions ("placeholders" - reserved words starting with $WB).

Depending of its type, WBSP file can be divided in up to six (6) sections – Configuration section, Top section, Header, Detail and Footer sections and Bottom section, but it can also be ordinary document with no sections at all. The number of sections depends upon type of operation that WBSP page is created for.

For example, this is a WBSP file divided in four sections (section dividers are marked red):

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=publishers
WB_Command=Q
wb_showlogo=F
wb_order=name
<!--WB_BeginTemplate-->
<html>
<head>
<title>Publishers</title>
</head>
<body>
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:14px;font-
weight:bold;color:#0066cc;">$wbf[Name]</span><br>
$wbsr[titles.sr]<br>
<!--WB_EndDetail-->
</body>
</html>
```

and this one has no sections at all:

```
<html>
<head>
<title>Publishers</title>
</head>
<body>
Hello!
Today is $wbfn[Weekdays], $wbfn[fdt(dd.mmm.yyyy)]!
</body>
</html>
```

Here is a brief explanation for all 6 sections:

```
<!--Start of Configuration section-->
<!-- comment tag to prevent WYSIWYG editors from destroying the file
structure
[FormFields]
#this subsection contains most of the WB variables needed by WBSP to
run properly
[MsgAndLbl]
```

```
#this subsection contains custom messages and labels
[Upload]
#this subsection contains variables needed for processing multipart
form data
[ErrorMessages]
#this subsection contains custom error messages
[Include]
#this subsection contains paths to include files with WB variables
[Referrer Check]
#this subsection defines valid referrers for current page
[UserData]
#this subsection defines custom, user-defined variables
-->
<!--WB_BeginTemplate-->
<!--End of Configuration section-->
<!--Start of Top section-->
    <html>
    <body>
<!--End of Top section-->
    <!--Start of Detail section-->
    <!--WB_BeginDetail-->
        <!--Start of Header section-->
        <!--WB_BeginHeader-->
        <!--this is where WB report tags and functions for header
section should be placed-->
        <!--WB_EndHeader-->
        <!--End of Header section-->
    <!--this is where WB report tags and functions for detail section
should be placed-->
        <!--Start of Footer section-->
        <!--WB_BeginFooter-->
        <!--this is where WB report tags and functions for footer
section should be placed-->
        <!--WB_EndFooter-->
        <!--End of Footer section-->
    <!--WB_EndDetail-->
    <!--End of Detail section-->
<!--Start of Bottom section-->
    </body>
    </html>
<!--End of Bottom section-->
```

17.1 Configuration section

This section should be placed on top of the WBSP file and separated from rest of the file with **<!--WB_BeginTemplate-->** comment. This way WBSP engine will not send anything from this section to the client (e.g. visitor's browser), and all variables and file paths will stay hidden from visitor. Configuration section contains definitions for WBSP variables needed for a specific task. It can have one to seven subsections, depending on purpose of the WBSP page – Include, FormFields, Upload, MsgAndLbl, Referrer check, ErrorMessages and UserData . To learn more about these sections please read the explanations for each section in chapter "Configuration section subsections".

Here's an example (configuration section is marked blue)

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

17.2 Top section

Everything from top of the Wbsp file (or from <!--WB_BeginTemplate--> comment, if it exists) to <!--WB_BeginDetail--> comment is considered to be a Top section. It will be processed by WBSP engine and sent to client, before WBSP processes the Detail section . It can contain DB related functions , but it will ignore WB_StartRec variable (i.e. for all $wbf functions it will return the value of the first record, and not the record at the position defined in WB_StartRec).

Here's an example (top section is marked blue)

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
```

```
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

17.3 Detail section

Detail section is everything located between <!--WB_BeginDetail--> and <!--WB_EndDetail--> comments. When WBSP start processing the detail section, it moves the recrdset to the record defined in WB_StartRec variable, processes all tags and functions found in detail section, sends the resulting content to the client, moves the recordset to the next record and repeat the process for every record in record range (starting with record number WB_StartRec and ending with record number WB_StartRec + WB_MaxRec). It means that detail section will be repeated as many times as there is records in the range. It is very important to place <!--WB_BeginDetail--> and <!--WB_EndDetail--> comments properly, because misplacing them can produce unwanted results.

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
```

```
</body>
</html>
```

When processed this WBSP page will produce a table with 3 columns and 20 rows containing filed values for Year published, Title and ISBN fields from table Titles.

However, if we misplace the <!--WB_BeginDetail--> and <!--WB_EndDetail--> comments by putting hem inside the <tr> </tr> structure, like this:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<tr>
<!--WB_BeginDetail-->
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
<!--WB_EndDetail-->
</tr>
</table>
<center>$wbnavigator</center>
</body>
</html>
```

we'll get a two rows table with **3 columns** in first row and **60 columns** in second row (3 columns for every record displayed).

17.4 Header and Footer sections

Header and footer are subsections of detail section. When used they must be placed on the very beginning (header) and very end (footer) of the detail section. This means that nothing (HTML code, RTF tags, plain text, etc.) can not be placed between <!--WB_BeginDetail--> and <!--WB_BeginHeader--> and/or between <!--WB_EndFooter--> and <!--WB_EndDetail-->. This is an example of correctly placed header and footer (marked blue):

```
<!--
[FormFields]
wb_command=q
wb_basename=biblio.mdb
wb_rcdset=titles
wb_changeHFon=[Year published]
```

```
wb_order=[Year published]
wb_maxrec=$all$
-->
<!--WB_BeginTemplate-->
<html>
<body>
$wbsetv[countbooks|0]
<!--WB_BeginDetail-->
<!--WB_BeginHeader-->
<strong>This is the list of titles for year $wbf[year
published]</strong><br>
<!--WB_EndHeader-->
$wbf[title] - $wbf[isbn]<br>
$wbsetv[countbooks|$wbcalc[$wbgetv[countbooks]+1]]
<!--WB_BeginFooter-->
<hr>
Total books in $wbf[year published]: <strong>
$wbgetv[countbooks]</strong><hr><br>
$wbsetv[countbooks|0]
<!--WB_EndFooter-->
<!--WB_EndDetail-->
</body>
</html>
```

Header and Footer sections will be processed and sent to the client before and after Detail section whenever any of referent fields change it's value (to learn more about referent fields please read the explanation for input variable WB_ChangeHFOn). Header and Footer sections will also be shown at the beginning of the Detail section for first record and at the end of the Detail section for last record on the report.

17.5 Bottom section

Everything from <!--WB_EndDetail--> comment to the end of the report template is Bottom section. It will be processed by WBSP engine and sent to client, after WBSP processes the Detail section . If it can contains DB related functions , they will not produce error, but they also will not return proper value (eg. for all $wbf functions it will return empty string). To display proper result of these functions in bottom section, place the result in a variable using $wbsetv function, and retrieve them in bottom section using $wbgetv function.

Here's an example (bottom section is marked blue)

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
```

```
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

17.6 WhizBase SubReports

Due to its simplicity and unique way of working with databases, single WBSP file can work with single recordset regardless of recordsets complexity. However, in most real-life situations there is the need for using more than one recordset for a single task (in a single wbsp file). For that purpose WhizBase uses subreports - a report that is inserted in another report. When you combine two or more reports, one of them must serve as the main report (main WBSP file).
To include subreport in main WBSP file use either $WBSR or $WBSRQ functions. Although subreport file looks very similar to ordinary WBSP file, there are few differences:

- Subreport can only execute Q command (i.e. it is not possible to add, delete or modify data using subreport)
- It reads only a subset of WBSP variables - mostly those related to database recordset:
  FormFields section
  WB_BaseName, WB_ShowEmpty, WB_LCID , WB_DBObject, WB_WC, WB_InsBR, WB_StartRec, WB_MaxRec, WB_RcdSet, WB_TempName, WB_Query, WB_SetADOCompatible, WB_ChangeHFOn, WB_Order, WB_Exclusive, WB_ReadOnly, WB_Connect, WB_Last, WB_Usr, WB_Pass, WB_System, WB_Predicate, WB_Group, WB_Having, WB_DBFlds
  MsgAndLbl section
  WBM_NoMatch
  Including any other WBSP variable in subreport will not generate the error, but the additional variables will be ignored.
- WB_Query variable in subreport file has different behavior compared to same variable in ordinary WBSP file - it is possible to include $wbf, $wbrf, $wbfc, $wbfu and $wbff functions in wb_query variable and when any of these are included they should be written using report syntax (with square brackets) and not input syntax (with braces). Any occurrence of these functions will be replaced with proper content from superior recordset (one defined in main WBSP file)
- Subreport ignores all code that is above <body...> and bellow </body> tags, including the tags itself.

Subreports recordset can, but does not have to, be related to the data in the main report. For example, you can use the main report to list publishers and subreport to list titles published by them.

This is the code for main WBSP file (subreport line is marked red):

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=publishers
WB_Command=Q
wb_showlogo=F
wb_order=name
<!--WB_BeginTemplate-->
<html>
<head>
<title>Publishers</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:14px;font-
weight:bold;color:#0066cc;">$wbf[Name]</span><br>
$wbsr[titles.sr]<br>
<!--WB_EndDetail-->
</body>
</html>
```

The code for subreport file (titles.sr) looks like this (line that connects two recordsets is marked blue):

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_query=PubID=$wbf[PubID]
wb_order=[Year published]
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:11px;">$wbf[Year published]-
$wbf[title] <strong>(ISBN:$wbf[ISBN])</strong></span><br>
<!--WB_EndDetail-->
</body>
</html>
```

The resulting page would look something like this:

Addison-Wesley Pub Co
1996-Real World FreeHand 5.0/5.5 **(ISBN:0201883600)**

Apress
2001-C# and the .NET Platform **(ISBN:1893115593)**

McGraw-Hill Professional Publishing
1996-Windows Nt Security Handbook **(ISBN:0078822408)**
1998-Microsoft Internet Information Server 4: the Complete Reference **(ISBN:0078824575)**
2000-Microsoft SMS Installer **(ISBN:0072124474)**
2001-McGraw-Hill's Encyclopedia of Networking & Telecommunications **(ISBN:0072120053)**

NAPP Publishing, Inc.
2000-Photoshop 6 Down and Dirty Tricks **(ISBN:0967985307)**

New Riders Publishing
2001-Photoshop 6 Photo-Retouching Secrets **(ISBN:0735711461)**

Osborne McGraw-Hill
2001-Windows 2000 Iis 5.0 : A Beginner's Guide **(ISBN:0072133724)**

Peachpit Press
1996-Netscape 3 for Macintosh Visual Quickstart Guide **(ISBN:0201694085)**
1997-Kai's Power Tools 3 for Windows Visual Quickstart Guide **(ISBN:0201696681)**
1997-Real World Freehand 7 **(ISBN:0201688875)**
1999-Non-Designer's Scan and Print Book, The **(ISBN:0201353946)**
2000-InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide **(ISBN:0201710366)**
2000-HTML 4 for the World Wide Web: Visual Quickstart Guide **(ISBN:0201354934)**
2000-Real World Adobe InDesign 1.5 **(ISBN:0201354780)**
2001-Macromedia FreeHand 10 for Windows and Macintosh: Visual QuickStart Guide **(ISBN:0201749653)**
2001-Fireworks 4 for Windows and Macintosh Visual Quickstart Guide **(ISBN:0201731339)**

Sams
2000-Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours **(ISBN:0672320428)**
2000-Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours **(ISBN:0672318830)**

The Coriolis Group
1999-Apache Server for Windows Little Black Book: The Indispensable Guide to Day-to-Day Apache Server Tips and Techniques **(ISBN:1576103919)**

Watson-Guptill Pubns
2000-www.type: Effective Typographic Design for the World Wide Web **(ISBN:0823058603)**
2000-www.color **(ISBN:0823058573)**
2001-Www.Layout : Effective Design and Layout for the World Wide Web **(ISBN:0823058581)**

Wordware Publishing
2000-Developer's Workshop to COM and ATL 3.0 **(ISBN:1556227043)**

Wrox Press Inc.
1997-Beginning C **(ISBN:1861001142)**
1998-Beginning Visual C++ 6 **(ISBN:186100088X)**
1999-Beginning Java 2 **(ISBN:1861002238)**

In this example main report uses table "Publishers" to list publishers in detail section and then uses subreport to show titles published by current publisher. Unique identification for every record in table "Publishers" is field PubID. In table "Titles" publisher is identified with field PubID (also in every record). Setting WB_Query to this value will generate subreport for every record in table "Publishers" and every

subreport will contain only those records from table "Titles" where PubID field has the same value as PubID field of publishers recordset.
Subreport files can have any extension you want. The .sr extension in the example is used to separate subreport from main WBSP file.

For security reasons it is good policy to name subreport files with extension .sr. It is because WBSP server-side configuration variable HideDocuments in default value include .sr extension, which means that WBSP will return HTTP error 404 (File not found) whenever someone tries to execute any .sr file directly (e.g. by typing its URL in browsers address bar).

18. Configuraton section subsections

Configuration section contains definitions for WBSP variables needed for a specific task. It can have one to seven subsections, depending on purpose of the WBSP page – **Include**, **FormFields**, **Upload**, **MsgAndLbl**, **Referrer check**, **ErrorMessages** and **UserData**.

18.1 Subsection [Include]

This subsection contains a list of files (respecting the WBSP path rules) that WBSP engine will check for input variables if they are not found in original WBSP file. Include files are processed backwards – from end to top of the list. WBSP will stop searching as soon as it finds first occurrence of the variable it search for.
Include files will be processed only if WBSP engine does not find the variable in WBSP file, but before WBSP engine tries to read variable from HTML form input fields. WBSP engine will read include files backwards (starting from the last file in the include section).

Here is an example of Include section (marked blue):

```
<!--
[Include]
db.inc
[FormFields]
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
```

```
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

And include file db.inc looks like this:

```
[FormFields]
WB_Basename=biblio.mdb
```

Special case are include files named default.inc.

18.1.1 Default.inc

Special case of include file are files named **default.inc**. These files will always be included in search, even if there is no Include section defined in WBSP file. WBSP engine will try to read default.inc file located in same directory with WBSP file and, if that file does not exist or does not contain specified variable, it will proceed reading all default.inc files located in directories above current directory until wwwroot directory of virtual host is reached. However, none of the files will be processed if WBSP engine finds the variable either in WBSP file itself or in other include files.

18.2 Subsection [FormFields]

This subsection contains the variables that are essential for processing WBSP file. Here you put information about the database, recordset, template, error template, log file, redirection, etc.
These are the variables that can be stored in FormFields subsection (in alphabetic order):

WB_AddCookie	WB_Debug	WB_Order
WB_AddJoker	WB_Defaults	WB_Pass
WB_AllowMultipart	WB_Destination	WB_PID
WB_AndOr	WB_ErrFile	WB_Predicate
WB_AppendMode	WB_ErrMail	WB_Query
WB_Attach	WB_ExactCount	WB_RcdSet
WB_AttachField	WB_Exclusive	WB_ReadOnly
WB_BaseName	WB_Execute	WB_Redirect
WB_BCC	WB_FileName	WB_Required
WB_BCCField	WB_Forced	WB_Section
WB_CC	WB_From	WB_Separator
WB_CDate	WB_FULID	WB_SetADOCompatible
WB_ChangeHFOn	WB_Group	WB_ShowEmpty
WB_Command	WB_Having	WB_ShowLogo
WB_Connect	WB_HideLogin	WB_StartRec
WB_ContentType	WB_HTTPHeader	WB_Subject
WB_DBAddData	WB_InsBR	WB_System
WB_DBAdmin	WB_KeyName	WB_SysVarByForm
WB_DBDelData	WB_KeyValue	WB_TempName
WB_DBEditData	WB_LCID	WB_TimeOut
WB_DBFlds	WB_Log	WB_To

WB_DBGroup	WB_LogData	WB_ToField
WB_DBLock	WB_LogTemp	WB_UID
WB_DBModDes	WB_MailPort	WB_Unicode
WB_DBNewPass	WB_MailServer	WB_UniFTS
WB_DBNPassCh	WB_MatchCase	WB_UserData
WB_DBObject	WB_MaxPages	WB_Usr
WB_DBOldPass	WB_MaxRec	WB_ValDelimiter
WB_DBReadData	WB_MQ	WB_WC
WB_DBReadDes	WB_Null	WB_WholeWord
WB_DBUser		

Here's an example of FormFields section (marked blue):

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=publishers
WB_Command=Q
wb_showlogo=F
wb_order=name
<!--WB_BeginTemplate-->
<html>
<head>
<title>Publishers</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:14px;font-
weight:bold;color:#0066cc;">$wbf[Name]</span><br>
$wbsr[titles.sr]<br>
<!--WB_EndDetail-->
</body>
</html>
```

18.3 Subsection [Upload]

This subsection contains variables needed to control file upload using WBSP page. Using these variables, the developer can control where uploaded files will be saved, what is maximum size of a single file, what file types can not be uploaded, weather existing file will be overwritten or not, where to store log data and what prefix to add to URL of uploaded file. The variables that can be stored in this subsection are (in alphabetic order):
WB_BaseURL
WB_Disallow
WB_MaxFSize
WB_Overwrite
WB_UploadDir
WB_UploadLog

Here's an example of upload subsection (marked with blue):

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_AllowMultipart=T
WB_Command=A
WB_UID=ISBN
WB_Redirect=titlesQ.wbsp
```

```
WB_RcdSet=Titles
[Upload]
WB_Disallow=![jpg,gif]
WB_UploadDir=images/
WB_Overwrite=T
WB_MaxFSize=24576
WB_UploadLog=upload.log
```

To learn more about uploading files using WBSP read the "Uploading files using WBSP" page.

18.4 Subsection [MsgAndLbl]

This subsection contains the variables that define labels and style for navigation links (for previous, next, first and last page) and messages (empty recordset and deleted records). They can contain any valid HTML code including WhizBase report tags and functions both in input and output syntax.
It can also contain variables for formatting navigation links WB_DigitDir and WB_Style .

These are variables that can be stored in this subsection (in alphabetic order):
WB_DigitDir
WB_Style
WBL_FirstPage
WBL_LastPage
WBL_NextPage
WBL_PrevPage
WBM_Deleted
WBM_NoMatch

Here is the example of MsgAndLbl section (marked blue):

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
[MsgAndLbl]
WB_DigitDir=digits/
WB_Style=color:#CC0000; text-decoration: none
wbl_nextpage=<img src="images/rightarrow.gif" border="0">
wbl_prevpage=<img src="images/leftarrow.gif" border="0">
wbl_lastpage=<img src="images/last.gif" border="0">
wbl_firstpage=<img src="images/first.gif" border="0">
wbm_nomatch=Sorry!<br>Your search returned no records!<br><a >Please
try again with other parameters</a>
wbm_deleted=Operation completed!<br><b>$WBDeleted</b> record(s)
successfully removed from your database!
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
```

```
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

To test this example, create images rightarrow.gif, leftarrow.gif, first.gif and last.gif and place them in directory images located in same directory with this wbsp file. Also create images of the digits 0 to 9 (name them 0.gif, 1.gif, 2.gif, ..., 9.gif) and place them in directory digits located in same directory with this wbsp file.

18.5 Section [UserData]

This section contains a list of custom, user-defined variables that can be used in WBSP files, in form:
UserVariable=AnyValue
These variables can be read using functions $WBRV and $WBRRV.
The user-defined variables in UserData section can contain WhizBase tags and functions, same as any WBSP system variable (variables starting with WB_).

Here is an example of UserData section in default.inc file in virtual hosts document root:

```
[UserData]
AdminEmail=admin@somemailserver.com
MyMailServer=mail.somemailserver.com
```

Later these variables can be accessed from your WBSP files like this:

```
[FormFields]
wb_mailserver=$wbrv{MyMailServer}
wb_bcc=$wbrv{AdminEmail}
```

Or like this:

```
<html>
<body>
some html code ...

Please send your comments/questions to
<strong>$wbrv[AdminEmail]</strong><br>
Thank you!
</body>
</html>
```

18.6 Section [ErrorMessages]

This section contains a list of custom error messages in form:
ErrorNumber=ErrorMessage
ErrorNumber is standard number reported by WBSP engine. It can be either WhizBase error number or Jet or any other error number. It is not recommended to put any Whizbase tags, functions, etc. in the ErrorMessage, because it can produce unwanted results. If you need to place such content in ErrorMessage, it is strongly recommended to test all possible situations before publishing the page to the real world. The exception to this is a reserved word **$WBERRRED:** followed by the URL of the web content where the WBSP engine will redirect visitor in the case of a specific error occurrence. This will work only if error occurs before WBSP sends HTTP header (e.g. if error occurs before WBSP starts report rendering).
This is useful for translating the error messages from English or for using IMG HTML tag to display an image instead of error message, or to use <script></script> block to process certain errors.

Here is an example of ErrorMessages section:

```
[ErrorMessages]
5034=<script>alert("You MUST provide values for all form
fields!");history.back();</script>
3030=<script>alert("Invalid login!");history.back();</script>
429=<img src="noActiveX.gif" border="0">
5040=$WBERRRED:/requiredMissing.wbsp
```

18.7 Referrer Check Section

Section Referrer Check defines what commands will require validation of referring page and what domains will be accepted as valid referrers.
It has a variable Referrer that contains a comma-separated list of servers (domains) that will be accepted by WBSP engine as valid hosts for referring pages.
If this variable does not exist (or it is empty) then WBSP engine will not try to validate referrer. If you want to allow referring page to be located only on same server as WBSP file set this variable to $self$.
The main change from previous versions is that this section is part of the WBSP file while in previous versions it was part of WhizBase.ssc server-side configuration file.

Here is an example of Referrer check section:

```
[Referrer Check]
Referrer=whizbase.com,wbsp.com
```

19. Update prefixes

As you could see from examples for updating records using WBSP, when new values for database table fields are received, the engine will replace old values with new ones. However in some cases there is a need to store new value that is related to old one (e.g. increased by some value, decreased by some value, etc.), or to delete field value completely by setting new value to empty string (""). Some of these tasks can

not be done at all (without these prefixes), and those that can be done require a lot of work to accomplish. Therefore WBSP has defined **update prefixes**, special values that should be added at the beginning of the WBF_ form fields, before the actual value. The prefixes are:
$WBNULL$
$WBP$
$WB-$
$WB/$
$WB*$
$WBA$
$WBR$

19.1 $WBNULL$ - delete value

Availability
UPDATE command

Syntax
$wbnull$

Description
Since WBSP engine ignores empty WBF_*tablefieldname* form fields, it is not possible to delete content of non-numeric field by sending empty string as a WBF_ field value. Therefore WBSP uses $WBNULL$ as a value for WBF_ form field(s) to define which database table fields in current (updated) record should be set to empty string. It does not apply to the numeric fields (to clear them set their value to 0).

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_ImageURL" value="$wbnull$">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
<img src="images/$wbf[imageurl]"><br>
<input type="submit" name="sClrImg" value="Remove image">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
</body>
</html>
```

19.2 $WB-$ - subtract from value

Availability
UPDATE command

Syntax
$WB-$*NumericValueToBeSubtracted*

Description
This is a form field value prefix for doing mathematical operation of subtraction with values of fields of numeric type. When WBSP engine receives a WBF_*tablefieldname* form field value with this prefix for database table field of numeric type, instead of replacing the old value with new one, it will subtract the new value from the existing one (so new value stored in the database is OldDatabaseTableFieldValue-NewFormValue)

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_Qty" value="$wb-$10">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
Current quantity: $wbf[Qty]<br>
<input type="submit" name="sUpd" value="Take 10 books">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
</body>
</html>
```

After running this example (and pressing the "Take 10 books" button), field **Qty** in updated record of table **Titles** will be decreased by 10.

19.3 $WB*$ - multiply value by

Availability
UPDATE command

Syntax
$WB*$*NumericValueToBeMultipliedBy*

Description
This is a form field value prefix for doing mathematical operation of multiplication with values of fields of numeric type. When WBSP engine receives a

WBF_*tablefieldname* form field value with this prefix for database table field of numeric type, instead of replacing the old value with new one, it will multiply existing value with the new value (so new value stored in the database is OldDatabaseTableFieldValue*NewFormValue)

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_Price" value="$wb*$2">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
Current price: $wbf[Price]<br>
<input type="submit" name="sUpd" value="Double the price">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
</body>
</html>
```

After running this example (and pressing the "Double the price" button), field **Price** in updated record of table **Titles** will be multiplied by 2.

19.4 $WB/$ - divide value with

Availability
UPDATE command

Syntax
$WB/$*NumericValueToBeDividedBy*

Description
This is a form field value prefix for doing mathematical operation of division with values of fields of numeric type. When WBSP engine receives a WBF_*tablefieldname* form field value with this prefix for database table field of numeric type, instead of replacing the old value with new one, it will divide existing value with the new value (so new value stored in the database is OldDatabaseTableFieldValue/NewFormValue)

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
```

```
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_Price" value="$wb/$2">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
Current price: $wbf[Price]<br>
<input type="submit" name="sUpd" value="Reduce the price by half">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
</body>
</html>
```

After running this example (and pressing the "Reduce the price by half" button), field **Price** in updated record of table **Titles** will be divided by 2.

19.5 $WBA$ - append text to value

Availability
UPDATE command

Syntax
$WBA$*TextToBeAppended*

Description
This is a form field value prefix for appending text to fields of string types (text, memo, etc.). When WBSP engine receives a WBF_*tablefieldname* form field value with this prefix for database table field of string type, instead of replacing the old value with new one, it will append the new value to the existing one (so new value stored in the database is OldDatabaseTableFieldValueNewFormValue)

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_Title" value="$wba$ - second edition">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
<input type="submit" name="sUpd" value="Second edition">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
```

```
</body>
</html>
```

After running this example (and pressing the "Second edition" button), text **- second edition** will be appended to the field **Title** in updated record of table **Titles**.

19.6 $WBP$ - add to value

Availability
UPDATE command

Syntax
$WBP$*NumericValueToBeAdded*

Description
This is a form field value prefix for doing mathematical operation of addition with values of fields of numeric type. When WBSP engine receives a WBF_*tablefieldname* form field value with this prefix for database table field of numeric type, instead of replacing the old value with new one, it will add new value to the existing one (so new value stored in the database is NewFormValue+OldDatabaseTableFieldValue)

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_Qty" value="$wbp$10">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
Current quantity: $wbf[Qty]<br>
<input type="submit" name="sUpd" value="Add 10 books">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
</body>
</html>
```

After running this example (and pressing the "Add 10 books" button), field **Qty** in updated record of table **Titles** will be increased by 10.

19.7 $WBR$ - remove text from value

Availability
UPDATE command

Syntax
$WBA$*TextToBeRemoved*

Description
This is a form field value prefix for removing text from fields of string types (text, memo, etc.). When WBSP engine receives a WBF_*tablefieldname* form field value with this prefix for database table field of string type, instead of replacing the old value with new one, it will search the existing value for value sent by form (all text after $wbr$ prefix) and if it founds it, remove it from the database field (so new value stored in the database is OldDatabaseTableFieldValue with removed all occurrences of NewFormValue)

Here's an example:

```
<!--
[FormFields]
WB_BaseName=biblioA.mdb
WB_RcdSet=Titles
WB_Command=Q
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
<form action="TitlesUpdate.wbsp" method="POST">
<input type="hidden" name="WBF_Title" value="$wbr$ - second edition">
<input type="hidden" name="wbf_isbn" value="$wbf[isbn]">
$wbf[Title]<br>
<input type="submit" name="sUpd" value="Remove second edition">
</form>
<!--WB_EndDetail-->
<center>$wbnavigator</center>
</body>
</html>
```

After running this example (and pressing the "Remove second edition" button), if text **- second edition** exists in field **Title** of updated record of table **Titles**, it will be removed. If the text does not exist (not found), the database field will not be changed.

20. Report tags

Report tags are WBSP placeholders that do not have any arguments, and therefore their syntax does not necessarily include neither square brackets nor braces. However, if for any reason you need to pass them as arguments to any WBSP function you have to add empty square brackets [] or braces {} at the end.

$WBADMIN
$WBFILEREPORT
$WBFULID
$WBDOCROOT
$WBCURRDIR
$WBTIMER

Database related tags
$WBQUERY
$WBCQUERY
$WBDELETED
$WBRECORDBREAK

Error message tags
$WBERRDESC
$WBERRMSG
$WBERRNUM
$WBERRMAIL

Navigation tags
$WBNAVIGATOR
$WBPAGENUMS
$WBPREVPAGE
$WBNEXTPAGE
$WBFIRSTPAGE
$WBLASTPAGE

20.1 $WBAdmin

Availability
$WBAdmin is available for use with following WBSP commands:
Change Password
Add User or Group
Delete User or Group
Add User to Group
Delete User from Group
Set user's permissions
Read user's permissions
Compact database

Syntax
$WBAdmin
$WBAdmin[]

Returns
Report of executed database administration command.

Here's an example:

```
<!--
[FormFields]
wb_basename=biblioA.mdb
wb_command=CD
-->
<!--WB_BeginTemplate-->
<html>
<body>
$wbadmin
</body>
</html>
```

After running this example database biblioA.mdb will be compacted and repaired (if needed), and default message will be displayed:

```
Database file biblioA.mdb successfully repaired and compacted!
```

20.2 $WBCurrDir - current directory

Availability
$WBCurrDir is available for use with all WBSP commands.

Syntax
$WBCurrDir
$WBCurrDir[]

Returns
Path to the directory of current WBSP file (directory where it is located) relative to the $WBDocRoot.

Here's an example:

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<body>
The currenet script directory is:<br>
<b>$wbcurrdir</b>
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

```
The current script directory is:
/wbsp/basic/
```

20.3 $WBCurrDirA - current directory absolute path

Availability
$WBCurrDirA is available for use with all WBSP commands.

Syntax
$WBCurrDirA
$WBCurrDirA[]

Returns
Physical path to the directory of current WBSP file (directory where it is located).

Here's an example:

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<body>
The currenet script directory is:<br>
<b>$WBCurrDirA</b>
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

```
The current script directory is:
E:\WebSrv\webpages\wbsp\basic\
```

20.4 $WBDocRoot - root directory of virtual host

Availability
$WBDocRoot is available for use with all WBSP commands.

Syntax
$WBDocRoot
$WBDocRoot[]

Returns
Physical path to the root directory of the current virtual host.

Here's an example:

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<body>
The document root directory is:<br>
```

```
<b>$wbdocroot</b>
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

The document root directory is:
E:\WebSrv\webpages\

20.5 $WBFileReport

Availability
$WBFileReport is available for use with WBSP file commands:
Delete file
Write to file

Syntax
$WBFileReport
$WBFileReport[]

Returns
Report of executed file command.

Here's an example:

```
<!--
[FormFields]
WB_FileName =/default.inc
WB_Section =UserData
WB_KeyName =AdminName
WB_KeyValue =John Doe
wb_command=WF
-->
<!--WB_BeginTemplate-->
<html>
<body>
$wbfilereport
</body>
</html>
```

After running this example, user variable AdminName will be set to new value **John Doe**, and default message will be displayed:

Key value for AdminName sucessfully changed in file /default.inc!

20.6 $WBFULID - upload form ID

Availability
$WBFULID is available for use with all WBSP commands, but it requires WB_FULID variable to be set to TRUE in the configuration section of the same WBSP document.

Syntax
$WBFULID
$WBFULID[]

Returns
Server-side unique form upload ID generated by WhizBase.

Here's an example:

```
[FormFields]
WB_AllowMultipart=T
WB_Command=R
wb_showlogo=f
wb_timeout=300
WB_FULID=$wbif{"$wbv{image}"=""|t|f}
[Upload]
WB_Disallow=![jpg,gif]
WB_UploadDir=/
WB_BaseURL=/
WB_Overwrite=F
WB_MaxFSize=10485760
WB_UploadLog=upload.log
<!--WB_BeginTemplate-->
<html>
<head>
<title>$wbif["$wbv[image]"=""|Upload file|File uploaded]</title>
</head>
<body>
$wbif["$wbv[image]"=""|< br><form action="$wbe[script_name]"
method="post" ENCTYPE="multipart/form-
data"onsubmit="document.getElementById('ifrprogress').src='ru.wbsp?fid=
$wbfulid[]';">
Select file (*.jpg;*.gif - max. 10MB): <input type="file" name="image"
size="20"> <input type="submit" name="sButt" value="Upload">
</form>
<iframe src="" width="300" height="100" id="ifrprogress"></iframe>
|
Open uploaded image<br><a target="_blank">$wbv[image]</a><br>
]
</body>
</html>
```

For this example you'll need file ru.wbsp that can be found here.
After running this example, WhizBase will upload selected image to the server displaying the progress in an IFRAME.

20.7 $WBTimer - system timer

Availability
$WBTimer is available for use with all WBSP commands.

Syntax
$WBTimer
$WBTimer[]

Returns
Number of seconds elapsed since midnight.

Here's an example:

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=publishers
WB_Command=Q
wb_showlogo=F
wb_order=name
wb_userdata=$wbtimer
<!--WB_BeginTemplate-->
<html>
<head>
<title>Publishers</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:14px;font-
weight:bold;color:#0066cc;">$wbf[Name]</span><br>
$wbsr[titles.sr]<br>
<!--WB_EndDetail-->
Processing time (seconds):$wbcalc[$wbtimer[]-$wbfn[userdata]]
</body>
</html>
```

After running this example, total processing time will be displayed at the bottom of the resulting page (actual value may wary, depending on the system performance):

```
Processing time (seconds):0,690000000002328
```

20.8 Database related tags

Database related tags are WBSP keywords related to database operations.
$WBQUERY
$WBCQUERY
$WBDELETED
$WBRECORDBREAK

20.8.1 $WBCQuery - URL encoded query expression

Availability
$WBCQuery is available for use with following recordset related WBSP commands:
UPDATE
DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail

Syntax
$WBCQuery
$WBCQuery[]

Returns
WHERE clause of the SQL expression used to generate current recordset in url-encoded format.

Here's an example:

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_Command=D
WB_UID=ISBN
WB_RcdSet=Titles
wb_tempname=$default$
[MsgAndLbl]
WBM_Deleted=Operation completed! $WBDeleted record(s) removed from your
database!<br>Recordset generated upon following
condition(s):<br>$WBCQuery
```

After running this example by typing its URL followed by ?wbf_isbn=*isbnvalue* (**e.g. http://localhost/delete.wbsp?wbf_isbn=0823058573**), WBSP will display delete report with detailed search condition information:

```
Operation completed! 1 record(s) removed from your database!
Recordset generated upon following condition(s):
isbn%20%3D%20%270823058573%27
```

20.8.2 $WBDeleted - number of deleted records

Availability
DELETE command.

Syntax
$WBDeleted
$WBDeleted[]

Returns
Number of deleted records.

Here's an example:

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_Command=D
WB_UID=ISBN
WB_RcdSet=Titles
wb_tempname=$default$
[MsgAndLbl]
WBM_Deleted=Operation completed! $WBDeleted record(s) removed from your
database!
```

After running this example by typing its URL followed by ?wbf_isbn=*isbnvalue* (**e.g. http://localhost/delete.wbsp?wbf_isbn=0823058573**), WBSP will display delete report as defined in WBM_Deleted variable:

```
Operation completed! 1 record(s) removed from your database!
```

20.8.3 $WBQuery - query expression

Availability
$WBQuery is available for use with following recordset related WBSP commands:
UPDATE
DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail

Syntax
$WBQuery
$WBQuery[]

Returns
WHERE clause of the SQL expression used to generate current recordset.

Here's an example:

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_Command=D
WB_UID=ISBN
WB_RcdSet=Titles
wb_tempname=$default$
[MsgAndLbl]
WBM_Deleted=Operation completed! $WBDeleted record(s) removed from your
database!<br>Recordset generated upon following
condition(s):<br>$WBQuery
```

After running this example by typing its URL followed by ?wbf_isbn=*isbnvalue* (**e.g. http://localhost/delete.wbsp?wbf_isbn=0823058573**), WBSP will display delete report with detailed search condition information:

```
Operation completed! 1 record(s) removed from your database!
Recordset generated upon following condition(s):
isbn = '0823058573'
```

20.8.4 $WBRecordBreak - force next record

Availability
$WBRecordBreak is available for use with following recordset related WBSP commands:
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail

Syntax
$WBRecordBreak
$WBRecordBreak[]

Returns
Nothing.

Description
This tag is not a placeholder and will not be replaced by anything (it will not be shown on the report either). Its purpose is to move recordset one record forward.

Important: When $WBRecordBreak is used, WBSP will display **WB_maxrec * (Number of $WBRecordBreak tags +1)**
For example - if WB_MaxRec=10 and there is only one $WBRecordBreak tag on the page, WBSP will display 20 records per page.< /p>

In the following example we used $WBRecordBreak to make different background color for odd and even records:

```
<!--
[FormFields]
WB_basename=biblioa.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
$wbrecordbreak
<tr bgcolor="#c0c0c0">
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

After running this example, resulting page received by browser should look something like this:

Year published	Title	ISBN
2001	McGraw-Hill's Encyclopedia of Networking & Telecommunications	0072120053
2000	Microsoft SMS Installer	0072124474
2001	Windows 2000 Iis 5.0 : A Beginner's Guide	0072133724
1996	Windows Nt Security Handbook	0078822408
1998	Microsoft Internet Information Server 4: the Complete Reference	0078824575
1999	Non-Designer's Scan and Print Book, The	0201353946
2000	Real World Adobe InDesign 1.5	0201354780
2000	HTML 4 for the World Wide Web: Visual Quickstart Guide	0201354934
1997	Real World Freehand 7	0201688875
1996	Netscape 3 for Macintosh Visual Quickstart Guide	0201694085

20.9 Error message tags

Error message tags are WBSP placeholders for various parts of an error message. They are not intended to be used on the ordinary WBSP pages nor in normal circumstances, but in error template file (defined by WB_ErrFile variable and sent to client when error occurs). Their purpose is to format and/or hide some of the error details. Also, when output type (Content-type) is not **text/html** they can be used to generate proper error report (e.g. XML, XHTML, WML, RTF, etc.)
$WBERRDESC
$WBERRMSG
$WBERRNUM
$WBERRMAIL

20.9.1 $WBErrDesc - full error description

Availability
$WBErrDesc is available for use with error template file, processed by WBSP **in case of a run-time error**.

Syntax
$WBErrDesc
$WBErrDesc[]

Returns
Full error description including error message and error number.

Here's an example of an error template file:

```
<html>
<head>
<title>Sorry!</title>
</head>
<body>
$wberrdesc
```

```
</body>
</html>
```

When error occurs, instead of showing the default error report:

Error in /wbsp/basic/delete.wbsp

The following internal error has occurred:
Illegal unique identifier! Query returned more than one record!
Error Number = 5014

Please note what you were doing when this problem occurred, so we can identify and correct it. Write down the Web page you were using, any data you may have entered into a form or search box, and anything else that may help us duplicate the problem.Then contact the administrator of this service:

<webmaster@localhost>

WBSP will display following:

Illegal unique identifier! Query returned more than one record!
Error Number = 5014

20.9.2 $WBErrMail - email address shown in error report

Availability
$WBErrMail is available for use with error template file, processed by WBSP **in case of a run-time error**.

Syntax
$WBErrMail
$WBErrMail[]

Returns
Email address specified in WB_ErrMail form field for current WBSP file.

Here's an example of an error template file:

```
<html>
<head>
<title>Sorry!</title>
</head>
<body>
Sorry!<br>
The page you requested produced an error!<br>
Please contact the administrator of this service at <a
drkred">$wberrmail">$wberrmail</a>
</body>
</html>
```

When error occurs, instead of showing the default error report:

Error in /wbsp/basic/delete.wbsp

The following internal error has occurred:
Illegal unique identifier! Query returned more than one record!
Error Number = 5014

Please note what you were doing when this problem occurred, so we can identify and correct it. Write down the Web page you were using, any data you may have entered into a form or search box, and anything else that may help us duplicate the problem.Then contact the administrator of this service:

<webmaster@localhost>

WBSP will display following:

Sorry!
The page you requested produced an error!
Please contact the administrator of this service at webmaster@localhost

20.9.3 $WBErrMsg - error description (text only)

Availability
$WBErrMsg is available for use with error template file, processed by WBSP **in case of a run-time error**.

Syntax
$WBErrMsg
$WBErrMsg[]

Returns
Text-only error description without error number.

Here's an example of an error template file:

```
<html>
<head>
<title>Sorry!</title>
</head>
<body>
$wberrMsg
</body>
</html>
```

When error occurs, instead of showing the default error report:

Error in /wbsp/basic/delete.wbsp

The following internal error has occurred:
Illegal unique identifier! Query returned more than one record!
Error Number = 5014

Please note what you were doing when this problem occurred, so we can identify and correct it. Write down the Web page you were using, any data you may have entered into a form or search box, and anything else that may help us duplicate the problem.Then contact the administrator of this service:

<webmaster@localhost>

WBSP will display following:

Illegal unique identifier! Query returned more than one record!

20.9.4 $WBErrNum - error number

Availability
$WBErrNum is available for use with error template file, processed by WBSP **in case of a run-time error**.

Syntax
$WBErrNum
$WBErrNum[]

Returns
Error number, without any text message.

Here's an example of an error template file:

```
<html>
<head>
<title>Sorry!</title>
</head>
<body>
Sorry, the requested page produced the error number $wberrNum
</body>
</html>
```

When error occurs, instead of showing the default error report:

Error in /wbsp/basic/delete.wbsp

The following internal error has occurred:
Illegal unique identifier! Query returned more than one record!
Error Number = 5014

Please note what you were doing when this problem occurred, so we can identify and correct it. Write down the Web page you were using, any data you may have entered into a form or search box, and anything else that may help us duplicate the problem.Then contact the administrator of this service:

<webmaster@localhost>

WBSP will display following:

Sorry, the requested page produced the error number 5014

20.10 Navigation tags

Navigation tags are WBSP placeholders for various parts of a report navigation block (links used to navigate through report pages).
$WBNAVIGATOR
$WBPAGENUMS
$WBPREVPAGE
$WBNEXTPAGE
$WBFIRSTPAGE
$WBLASTPAGE

20.10.1 $WBFirstPage - navigation link to first report page

Availability
QUERY command

Syntax
$WBFirstPage
$WBFirstPage[]

Returns
Link to the first page of the report.

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
wb_startrec=$wbif{"$wbv{wb_startrec}"=""|16|$wbv{wb_startrec}}
-->
<!--WB_BeginTemplate-->
<html>
```

```
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>Back to $WBFirstPage</center>
</body>
</html>
```

After running this example, WBSP will display the report with link to first page only:

Back to First page

20.10.2 $WBLastPage - navigation link to last report page

Availability
QUERY command

Syntax
$WBLastPage
$WBLastPage[]

Returns
Link to the last page of the report.

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
wb_startrec=$wbif{"$wbv{wb_startrec}"=""|16|$wbv{wb_startrec}}
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
```

```
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>Go to $WBLastPage</center>
</body>
</html>
```

After running this example, WBSP will display the report with link to last page only:

Go to Last page

20.10.3 $WBNavigator - full set of report navigation links

Availability
QUERY command

Syntax
$WBNavigator
$WBNavigator[]

Returns
Full set of report navigation links (page-number links and links to first, last, previous and next page).

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
```

```
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$WBNavigator</center>
</body>
</html>
```

After running this example, WBSP will display the report with all navigation links at the bottom, separate links to every page (page-number links) and links to first, last, previous and next page:

```
                          1 2 3 4 5 6
          First page Previous page Next page Last page
```

20.10.4 $WBNextPage - navigation link to next report page

Availability
QUERY command

Syntax
$WBNextPage
$WBNextPage[]

Returns
Link to the next page of the report.

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
wb_startrec=$wbif{"$wbv{wb_startrec}"=""|16|$wbv{wb_startrec}}
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
```

```
<center>Go to $WBPrevPage or $WBNextPage</center>
</body>
</html>
```

After running this example, WBSP will display the report with links to previous and next pages:

Go to Previous page or Next page

20.10.5 $WBPageNums - links to separate report pages

Availability
QUERY command

Syntax
$WBPageNums
$WBPageNums[]

Returns
Separate links to every page of the report.

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$WBPageNums</center>
</body>
</html>
```

After running this example, WBSP will display the report with separate links to every page of the report:

1 2 3 4 5 6

20.10.6 $WBPrevPage - navigation link to previous report page

Availability
QUERY command

Syntax
$WBPrevPage
$WBPrevPage[]

Returns
Link to the previous page of the report.

Here's an example:

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
wb_startrec=$wbif{"$wbv{wb_startrec}"=""|16|$wbv{wb_startrec}}
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>Go to $WBPrevPage or $WBNextPage</center>
</body>
</html>
```

After running this example, WBSP will display the report with links to previous and next pages:

Go to Previous page or Next page

20.11 Session tags

20.11.1 $WBACTSES - active sessions

Availability
$WBACTSES is available for use with all WBSP commands.

Syntax
$WBACTSES
$WBACTSES[]

Returns
Number of active sessions.

Here's an example:

```
<!--
[FormFields]
wb_command=R
wb_usesessions=T
-->
<!--WB_BeginTemplate-->
<html>
<body>
There are $wbactses users on-line.
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

```
There are 27 users on-line.
```

21. Functions

Functions are WBSP placeholders with one or more arguments, separated with vertical line (|), and enclosed in square brackets or braces depending on WBSP page section in which they are located and processing phase in which they should be processed. Based on this, functions use report syntax ($wbfnname[arg]) and input syntax ($wbfnname{arg}). To learn more about differences between these two syntax models, please read the next topic - Difference between report and input functions . All functions that have more than one argument (separated with |) can be written in multi-line form. However, none of the arguments can be broken in separate lines, unless it is also WhizBase function.

This is a list of WhizBase functions in alphabetical order:

$WBAADD	$WBFSIZE	$WBRF
$WBACHG	$WBFTIME	$WBRIGHT
$WBALEN	$WBFU	$WBRINC
$WBALIDX	$WBFUP	$WBRRV
$WBASRC	$WBFUT	$WBRUN
$WBBAND	$WBGC	$WBRV

$WBBOR	$WBGETURL	$WBSETV
$WBBXOR	$WBGETV	$WBSPLIT
$WBCALC	$WBGS	$WBSR
$WBCASE	$WBGV	$WBSRQ
$WBCNL	$WBHE	$WBTRIM
$WBCSTR	$WBIF	$WBUNESC
$WBDIR	$WBINC	$WBUNTIL
$WBE	$WBINDOF	$WBURL
$WBERR	$WBLEFT	$WBV
$WBESC	$WBLEN	$WBVA
$WBF	$WBLINDOF	$WBVC
$WBFC	$WBMID	$WBVR
$WBFF	$WBMREPL	$WBVS
$WBFN	$WBP	$WBVSC
$WBFORMAT	$WBREPL	$WBWHILE

21.1 Difference between report and input functions

WhizBase input functions are similar to the report functions. They have different syntax (input functions use general syntax $wbfnname {arg} instead of report functions - $wbfnname[arg]), but the main difference is in way of using them. Inserting server's date/time in the database as record update time, for example, without input functions requires some serious work including unnecessary redirections and resource consuming.

Example:

If you put hidden field in your report like this

```
<input type="hidden" name="WBF_LastUpdated" value="$WBFN[fdt(dd-mmm-
yyyy hh:mm:ss)]">
```

then, when WBSP engine process this report, this field has value of server's date/time when the page was sent to browser:

```
<input type="hidden" name="WBF_LastUpdated" value="03-Apr-2008
13:22:47">
```

This field will have the same value even if the form is submitted hours later, what means that it is completely useless for tracking record update time.

However if you use same function in input syntax WhizBase will ignore it during report processing. To use same function as above, but this time in input syntax simply change the value as follows:

```
<input type="hidden" name="WBF_LastUpdated" value="$WBFN{fdt(dd-mmm-
yyyy hh:mm:ss)}">
```

WBSP will ignore this function during report processing and visitor's browser will receive exactly the same code as you wrote it in report:

```
<input type="hidden" name="WBF_LastUpdated" value="$WBFN{fdt(dd-mmm-
yyyy hh:mm:ss)}">
```

As an opposite from previous case it is of no importance how long you will wait

before submitting this form. Once you submit the form WBSP will receive
$WBFN{fdt(dd-mmm-yyyy hh:mm:ss)}
as a value for WBF_LastUpdated form field, process it and write exact update time to DB field named LastUpdated.

Because input functions are processed before opening the database it is not possible to use DB related functions as input functions (except $WBSR and $WBSRQ). Using these functions in input syntax will result with empty string.

Any WB form field can have input function in its value and they can be used both in HTML forms and WBSP files:

```
WB_Query=$wbif{"$wbv{pg}"=""|PageID=1|PageID=$wbv{pg}}
```
Or
```
<input type="hidden" name="WB_TempName" value="$WBFN{HTUser}.htm">
```

21.2 $WBAADD - add element to array

Important: $WBAADD function has effect on entire WBSP page and all it's sub elements (configuration section, included files, sub reports, etc.). All instances of $WBAADD function that use the same array name will add new element to the same array regardless of their location inside WBSP page.

Availability
$WBAADD is available for use with all WBSP commands.

Syntax
$WBAADD{*arrayname*|*varvalue*|***showvar***}
$WBAADD[*arrayname*|*varvalue*|***showvar***]

Parameters
arrayname - the name of the array to which new element will be added
varvalue - the value of the new element that will be added
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the new length of resulting array

Returns
New number of elements in resulting array if showvar parameter is set to true, otherwise it returns nothing, just adds the element.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBAAdd</title>
</head>
<body>
$wbsetv[filecount|0]
```

```
$wbsplit[$wbdir[|||d]|dirlist|,|f]
$wbwhile[$wbgetv[filecount]<=$wbsplit[$wbdir[]|fillist|,|t]
|
$wbaadd[dirlist|$wbgetv[fillist($wbgetv[filecount])]|f]
$wbsetv[filecount|$wbcalc[$wbgetv[filecount]+1]]
]
$wbsetv[filecount|0]$wbwhile[$wbgetv[filecount]<=$wbalen[dirlist]
|
$wbgetv[dirlist($wbgetv[filecount])]<br>
$wbsetv[filecount|$wbcalc[$wbgetv[filecount]+1]]
]
</body>
</html>
```

After running this example, the resulting page in browser should contain the list of all directories and files in the directory where current WBSP file is located.

21.3 $WBACHG - change value of array element

Availability
$WBACHG is available for use with all WBSP commands.

Syntax
$WBACHG{*arrayname*|*index*|*varvalue*|***show***}
$WBACHG[*arrayname*|*index*|*varvalue*|***show***]

Parameters
arrayname - the name of the array
index - index of the element
varvalue - the new value
show - optional parameter - if set to true (T,ON,1) WhizBase will return 1 if function was successful and 0 if not (element not found)

Returns
1 if function was successful and 0 if not (element not found).

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBAChg</title>
</head>
<body>
$wbsplit[Mon,Tue,Wed,thu,Fri,Sat,Sun|days|,|f]
$wbgetv[days(3)]<br>
$wbachg[days|3|Thu]
$wbgetv[days(3)]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
thu
Thu
```

21.4 $WBALEN - array length

Availability
$WBALEN is available for use with all WBSP commands.

Syntax
$WBALEN{*arrayname*}
$WBALEN[*arrayname*]

Parameters
arrayname - the name of the array

Returns
Number of elements in requested WBSP array

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBSplit</title>
</head>
<body>
$wbsetv[filecount|0]
$wbsplit[$wbdir[]|dirlist|,|f]
$wbwhile[$wbgetv[filecount]<=$wbalen[dirlist]
|
$wbgetv[dirlist($wbgetv[filecount])]<br>
$wbsetv[filecount|$wbcalc[$wbgetv[filecount]+1]]
]
</body>
</html>
```

After running this example, the resulting page in browser should contain the list of all files in the directory where current WBSP file is located.

21.5 $WBALIDX - last array index

Availability
$WBALIDX is available for use with all WBSP commands.

Syntax
$WBALIDX{*arrayname*}
$WBALIDX[*arrayname*]

Parameters
arrayname - the name of the array

Returns
Last index value in requested WBSP array

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBSplit</title>
</head>
<body>
$wbsetv[filecount|0]
$wbsplit[$wbdir[]|dirlist|,|f]
$wbwhile[$wbgetv[filecount]<=$WBALIDX[dirlist]
|
$wbgetv[dirlist($wbgetv[filecount])]<br>
$wbsetv[filecount|$wbcalc[$wbgetv[filecount]+1]]
]
</body>
</html>
```

After running this example, the resulting page in browser should contain the list of all files in the directory where current WBSP file is located.

21.6 $WBASRC - search array elements for specified value

Availability
$WBASRC is available for use with all WBSP commands.

Syntax
$WBASRC{*arrayname*|*varvalue*}
$WBASRC[*arrayname*|*varvalue*]

Parameters
arrayname - the name of the array to search
varvalue - the value to search for

Returns
The array index of the first matching element. If varvalue is not found the function returns -1.

Example

```
<!--
[FormFields]
wb_command=R
-->
```

```
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBASrc</title>
</head>
<body>
$wbsplit[Mon,Tue,Wed,Thu,Fri,Sat,Sun|days|,|f]
$wbasrc[days|Thu]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
3
```

21.7 $WBAPRN - concatenate elements of array

Availability
$WBAPRN is available for use with all WBSP commands.

Syntax
$WBAPRN{*arrayname*|***delimiter***}
$WBAPRN[*arrayname*|***delimiter***]

Parameters
arrayname - the name of the array to search
delimiter - optional parameter - defines the delimiter for array elements. Default value is comma (,).

Returns
All elements of the array delimited by delimiter.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBAPrn</title>
</head>
<body>
$wbsplit[Mon,Tue,Wed,Thu,Fri,Sat,Sun|days|,|f]
$wbaprn[days|<br>]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
Mon
Tue
Wed
Thu
Fri
Sat
Sun
```

21.8 $WBB64DEC - Base64 decode

Availability
$WBB64DEC is available for use with all WBSP commands:

Syntax
$WBB64DEC{*Base64EncodedString*}
$WBB64DEC[*Base64EncodedString*]

Parameters
Base64EncodedString - any string encoded using $WBB64ENC
Returns
Decoded value of *Base64EncodedString*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBB64DEC example</title>
</head>
<body>
Encoded text: 5N/84g==<br>
WBB64DEC:($WBB64DEC[5N/84g==])<br>
</body>
</html>
```

After running this example the HTML code of resulting page in browser, may look like this:

```
<html>
<head>
<title>WBB64DEC example</title>
</head>
<body>
Encoded text: 5N/84g==<br>
WBB64DEC:(äßüâ)<br>
</body>
</html>
```

21.9 $WBB64ENC - Base64 encode

Availability
$WBB64ENC is available for use with all WBSP commands:

Syntax
$WBB64ENC{*anystring*}
$WBB64ENC[*anystring*]

Parameters
anystring - any textual value that can be either WhizBase tag, function, plain text or any combination of those

Returns
Base64-encoded value of *anystring*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBB64ENC example</title>
</head>
<body>
Original text: äßüâ<br>
WBB64ENC:($WBB64ENC[äßüâ])<br>
</body>
</html>
```

After running this example the HTML code of resulting page in browser, may look like this:

```
<html>
<head>
<title>WBB64ENC example</title>
</head>
<body>
Original text: äßüâ<br>
WBB64ENC:(5N/84g==)<br>
</body>
</html>
```

21.10 $WBBAND - binary AND

Availability
$WBBAND is available for use with all WBSP commands.

Syntax
$WBBAND{*num1*|*num2*}
$WBBAND[*num1*|*num2*]

Parameters
num1, num2 - any two numbers in range +/-79,228,162,514,264,337,593,543,950,335

Returns
A result of bitwise comparison num1 AND num2.

Example

```
<html>
    <body>
    2 and 6 = $WBBAND[2|6]
    </body>
</html>
```

Result:

```
2 and 6 = 2
```

21.11 $WBBOR - binary OR

Availability
$WBBOR is available for use with all WBSP commands.

Syntax
$WBBOR{*num1*|*num2*}
$WBBOR[*num1*|*num2*]

Parameters
num1, num2 - any two numbers in range +/-79,228,162,514,264,337,593,543,950,335

Returns
A result of bitwise comparison num1 OR num2.

Example

```
<html>
    <body>
    2 or 6 = $WBBOR[2|6]
    </body>
</html>
```

Result:

```
2 or 6 = 6
```

21.12 $WBBXOR - binary XOR

Availability
$WBBXOR is available for use with all WBSP commands.

Syntax
$WBBXOR{*num1*|*num2*}
$WBBXOR[*num1*|*num2*]

Parameters
num1, num2 - any two numbers in range +/-
79,228,162,514,264,337,593,543,950,335

Returns
A result of bitwise comparison num1 XOR num2.

Example

```
<html>
    <body>
    2 xor 6 = $WBBXOR[2|6]
    </body>
</html>
```

Result:

```
2 xor 6 = 4
```

21.13 $WBCACHE - cache content

Availability
$WBCACHE is available for use with all WBSP commands.

Syntax
$WBCACHE{code|time|uniqueID}
$WBCACHE[code|time|uniqueID]

Parameters
code - WhizBase code snippet that should be cached
time - period of validity of cached content in seconds
uniqueID - part of *code* that uniquely identifies the snippet (see example below)

Returns
Cached content if it is not expired (older than specified *time*) or executed *code* if it is.

Example

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=publishers
WB_Command=Q
wb_showlogo=F
wb_order=name
<!--WB_BeginTemplate-->
<html>
<head>
<title>Publishers</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:14px;font-
weight:bold;color:#0066cc;">$wbf[Name]</span><br>
$wbcache[$wbsrq[titles.sr|PubID=$wbf[PubID]]|300|$wbf[PubID]]<br>
<!--WB_EndDetail-->
</body>
</html>
```

When opened for the first time, this script will run sub report titles.sr and generate cache for every record. During next 300 seconds, any request for this script will return cached content, instead of running sub report titles.sr. We used $wbf[PubID] as unique identifier for cached content, because WhizBase would return the same cached content for all records if we did not.

21.14 $WBCALC - calculate math expression

Availability
$WBCalc is available for use with all WBSP commands.

Syntax
$WBCalc{*mathexp*}
$WBCalc[*mathexp*]

Parameters
mathexp - any valid mathematical expression.
Valid operators are:

Operator	Syntax	Operation
+	a + b	sum of two numbers
-	a - b	difference between two numbers
-	- a	negative value of a number
*	a * b	multiply two numbers
/	a / b	a divided by b
\	a \ b	integer part of a divided
%	a % b	the remainder of a division
mod	a mod b	the remainder of a division (same as %)

Valid mathematical functions are:

Operator	Syntax	Returns
sin	sin(a)	the sine of an angle
cos	cos(a)	the cosine of an angle
tan	tan(a)	the tangent of an angle
exp	exp(a)	e (the base of natural logarithms) raised to a power
log	log(a)	the logarithm to the base 10 of a number
ln	ln(a)	the natural logarithm of a number
atn	atn(a)	the arctangent of a number
abs	abs(a)	the absolute value of a number
sgn	sgn(a)	the sign of a number
sqr	sqr(a)	the square root of a number

Returns
The result of a mathematical expression.

Example
```
<!--
[FormFields]
wb_command=q
wb_basename=biblio.mdb
wb_rcdset=titles
wb_changeHFon=[Year published]
wb_order=[Year published]
wb_maxrec=$all$
-->
<!--WB_BeginTemplate-->
<html>
<body>
$wbsetv[countbooks|0]
<!--WB_BeginDetail-->
<!--WB_BeginHeader-->
<strong>This is the list of titles for year $wbf[year
published]</strong><br>
<!--WB_EndHeader-->
$wbf[title] - $wbf[isbn]<br>
$wbsetv[
countbooks
|
$wbcalc[$wbgetv[countbooks]+1]
]
<!--WB_BeginFooter-->
<hr>
Total books in $wbf[year published]:
<strong>$wbgetv[countbooks]</strong><hr><br>
$wbsetv[countbooks|0]
<!--WB_EndFooter-->
<!--WB_EndDetail-->
</body>
</html>
```

After running this example, total number of books will be displayed at the bottom of every year:

This is the list of titles for year 1998
Beginning Visual C++ 6 - 186100088X
Microsoft Internet Information Server 4: the Complete Reference - 0078824575

Total books in 1998: **2**

This is the list of titles for year 1999
Beginning Java 2 - 1861002238
Apache Server for Windows Little Black Book: The Indispensable Guide to Day-to-Day Apache Server Tips and Techniques - 1576103919
Non-Designer's Scan and Print Book, The - 0201353946

Total books in 1999: **3**

This is the list of titles for year 2000
Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours - 0672320428
HTML 4 for the World Wide Web: Visual Quickstart Guide - 0201354934
InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide - 0201710366
Real World Adobe InDesign 1.5 - 0201354780
Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours - 0672318830
www.color - 0823058573
Microsoft SMS Installer - 0072124474
www.type: Effective Typographic Design for the World Wide Web - 0823058603
Photoshop 6 Down and Dirty Tricks - 0967985307
Developer's Workshop to COM and ATL 3.0 - 1556227043

Total books in 2000: **10**

21.15 $WBCAPTCHA - show text as captcha

Availability
$WBCAPTCHA is available for use with all WBSP commands.

Syntax
$WBCAPTCHA{*SourceText*|***TableBorder***|***ImageOn***|***ImageOff***|***CellWidth***|***CellColor***|***TableStyle***|***BorderColor***|***BackgroundColor***}
$WBCAPTCHA[*SourceText*|***TableBorder***|***ImageOn***|***ImageOff***|***CellWidth***|***CellColor***|***TableStyle***|***BorderColor***|***BackgroundColor***]

Parameters
SourceText - a string that should be transformed to CAPTCHA
TableBorder - optional parameter - border thickness in pixels
ImageOn - optional parameter - URL of the image that will be placed in cells that form the character (contain the dot)
ImageOff - optional parameter - URL of the image that will be placed in blank cells
CellWidth - optional parameter - the width of a single cell in pixels

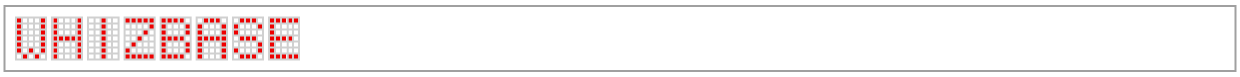
CellColor - optional parameter - the background color for cells that form the character (contain the dot)
TableStyle - optional parameter - style for tables forming the characters
BorderColor - optional parameter - the color of the border
BackgroundColor - optional parameter - the background color for blank cells

Returns
SourceText transformed into CAPTCHA.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBCaptcha</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
$wbcaptcha[whizbase|1|dot.gif|dot.gif|2|#EE0000|border-
collapse:collapse;|#d0d0d0|#FFFFFF]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:



21.16 $WBCASE - select case (switch)

Availability
$WBCASE is available for use with all WBSP commands.

Syntax
$WBCASE{*separator*|*value*|*conditionlist*|*resultlist*|*default*}
$WBCASE[*separator*|*value*|*conditionlist*|*resultlist*|*default*]

Parameters
separator - a character used to separate elements of *conditionlist* and *resultlist*. Default value is comma (,)
value - value that will be compared
conditionlist - list of conditions in form operator value (e.g. =3,>5,<>7), separated with *separator*
resultlist - list of return values for each condition, separated by *separator*
default - value that will be returned if none of the conditions is true

Returns
Code contained in true part or false part depending on result of evaluated expression.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
Random number $WBCASE[,|$wbfn[rnd(50)]|
=30,=10,=25,>=40,<10|
is 30,is 10,is 25,is greater or equal to 40, is less than 10|
is neither 10, 25, 30 nor it is greater than 40 or less than 10]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Random number is greater or equal to 40
```

Refresh the page few times and watch the changes of the result.

21.17 $WBCID - mail content ID

Availability
$WBCID is available for use with mail-related WBSP commands (P and L):

Syntax
$WBCID{*filename*}
$WBCID[*filename*]

Parameters
filename - the name of the file embedded to the mail message (using WB_Embed variable).

Returns
The CID (content id) value of *filename* generated during the embedding process.
Please note that *filename* argument of $WBCID function must be exactly the same as file name used in the WB_Embed variable.
If you change $wbcid[/poster.jpg] to $wbcid[poster.jpg] in the following example, the function will not return proper CID even if "/poster.jpg" and "poster.jpg" are both referencing the same file (e.g. both WBSP file and poster.jpg are located in the document root).

```
<!--
[FormFields]
wb_command=P
wb_mailserver=mail.whizbase.com
wb_to=info@whizbase.com
wb_from=info@whizbase.com
wb_subject=Test for WBCid
wb_embed=/poster.jpg,/alphabet.gif
-->
<!--WB_BeginTemplate-->
```

```
<html>
<head>
<title>WBCID</title>
</head>
<body>

<table border="0" cellpadding="5" cellspacing="0">
<tr>
<td><img border="0" src="$wbcid[/poster.jpg]" width="320"
height="422"></td>
<td><img border="0" src="$wbcid[/alphabet.gif]" width="300"
height="300"></td>
</tr>
</table>

</body>
</html>
```

21.18 $WBDCALC - calculate date

Availability
$WBDCALC is available for use with all WBSP commands.

Syntax
$WBDCALC{*startdate*|*offset*|*interval*|*operator*}
$WBDCALC[*startdate*|*offset*|*interval*|*operator*]

Parameters
startdate - starting date/time value
offset - number of intervals to be added to or subtracted from startdate
interval - interval of time to be added to or subtracted from startdate
Valid values are:
S - seconds
M - minutes
H - hours
D - days
operator - for addition set operator to + (plus sign), for subtraction set it to - (minus sign)
Returns
New date value created by adding or subtracting the *offset* number of *intervals* to/from *startdate*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Calculate date</title>
</head>
<body>
$wbdcalc[01.03.2009|1|D|-]<br>
$wbdcalc[01.03.2009 00:00:00|3600|S|-]
```

```
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
28.2.2009
28.2.2009 23:00:00
```

21.19 $WBDIR - list directory

Availability
$WBDIR is available for use with all WBSP commands.

Syntax
$WBDIR{*path*|*file*|*separator*|*attributes*}
$WBDIR[*path*|*file*|*separator*|*attributes*]

Parameters
path - directory or folder, and drive if AbsolutePath is enabled. Default value is current script directory
file - file name or comma-separated list of file names, including multiple-character (*) and single-character (?) wildcards to specify multiple files. Default value is *.*
separator - the character(s) used to separate files. Default value is comma (,)
attributes - any combination of the following values H-hidden,S-system, D-directory, F-normal files. Default value is F
Returns
List of *path* content that matches *file* pattern and any of the *attributes*. Empty string if no files match the file name.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Directory list</title>
</head>
<body>
Directory $wbcurrdir contents:<br>
$WBDIR[$wbcurrdir[]||<br>|FDSH]
</body>
</html>
```

After running this example, the resulting page in browser will list all the files and folders located in same directory as this WBSP file.

21.20 $WBDV - decrement value

Important: $WBDV function has effect on entire WBSP page and all it's sub elements (configuration section, included files, sub reports, etc.). All instances of $WBDV function that use the same variable name will set same variable regardless of their location inside WBSP page (e.g. $WBDV[somevar] in main WBSP page and $WBDV[somevar] in subreport will change (decrement by 1) the value of the same variable named **somevar**.

Availability
$WBDV is available for use with all WBSP commands.

Syntax
$WBDV{*varname*|***decrement***|***showvar***}
$WBDV[*varname*|***decrement***|***showvar***]

Parameters
varname - the name of global WBSP variable to be decremented
decrement - optional parameter - the value that will be subtracted from the existing value of *varname*. Default value is 1.
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the assigned value

Returns
New (decremented) value if showvar parameter is set to true, otherwise it returns nothing, just sets the value of a variable.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBDV</title>
</head>
<body>
$WBSETV[loopcounter|10]
$WBWHILE[$wbgetv[loopcounter]>0|
The loopcounter value is:$wbgetv[loopcounter]<br>
$WBDV[loopcounter]
]
Loop ended, loopcounter value is $wbgetv[loopcounter]!
</body>
</html>
```

After running this example, the resulting page in browser may look like this:

```
The loopcounter value is:10
The loopcounter value is:9
The loopcounter value is:8
The loopcounter value is:7
The loopcounter value is:6
The loopcounter value is:5
The loopcounter value is:4
The loopcounter value is:3
The loopcounter value is:2
The loopcounter value is:1
Loop ended, loopcounter value is 0!
```

21.21 $WBE - environment variable

Availability
$WBE is available for use with all WBSP commands.

Syntax
$WBE{*envstring*}
$WBE[*envstring*]

Parameters
envstring - string expression containing the name of an environment variable.
Some of the environmental variables generated by most web servers are:
CONTENT_LENGTH, CONTENT_TYPE, HTTP_COOKIE, GATEWAY_INTERFACE, QUERY_STRING, HTTP_REFERER, REMOTE_ADDR, REMOTE_HOST, REMOTE_IDENT, REMOTE_USER, REQUEST_METHOD, SCRIPT_NAME, SERVER_SOFTWARE, SERVER_NAME, SERVER_PORT, SERVER_PROTOCOL, HTTP_USER_AGENT

Returns
The value of an operating system environment variable.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<body>
Your IP address is $WBE[REMOTE_ADDR]!
</body>
</html>
```

After running this example, visitors IP address will be shown:

```
Your IP address is 127.0.0.1!
```

21.22 $WBERR - simulates an error

Availability
$WBERR is available for use with all WBSP commands.

Syntax
$WBERR{*errnum*|*errmsg*}
$WBERR[*errnum*|*errmsg*]

Parameters
errnum - error number in range from 6000 to 65535
errmsg - optional parameter containing textual description of an error. It should not be omitted if custom error message is not defined in ErrorMessages section of the current WBSP file or in include files.

Returns
Nothing.
When processing this function WhizBase will generate an error with specified number, and process it as any other error (stop the page processing and return an error report, or perform an error redirection).

Example

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_AllowMultipart=T
WB_Command=A
WB_Redirect=$wbe[http_referer]$wbif[$wbindof[$wbe[http_referer]|?]>0||?
wb_startrec=$wbv[sr]]
WB_RcdSet=Titles
wb_UserData=$wbif{$wbindof{$wbgv{system.cfg|AppGroups|$wbe{remote_addr}
}|;add record;}>0||$wberr{6001|Sorry!, you do not have the access
rights!}}
[Upload]
WB_Disallow=![jpg,gif]
WB_UploadDir=images/
WB_Overwrite=T
WB_MaxFSize=24576
WB_UploadLog=upload.log
```

After running this example (together with titleslist.wbsp), any access to this file from IP address other than 127.0.0.1 will generate an error message:

Error in /wbsp/basic/titlesAdd.wbsp

The following internal error has occurred:

```
Sorry!, you do not have the access rights!
Error Number =  6001
```

Please *note what you were doing when this problem occurred, so we can identify and correct it. Write down the Web page you were using, any data you may have entered into a form or search box, and anything else that may help us duplicate the problem. Then contact the administrator of this service: <webmaster@officeserver>*

21.23 $WBESC - URL encode string

Availability
$WBESC is available for use with all WBSP commands:

Syntax
$WBESC{*anystring*}
$WBESC[*anystring*]

Parameters
anystring - any textual value that can be either WhizBase tag, function, plain text or any combination of those

Returns
url-encoded value of *anystring*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBESC example</title>
</head>
<body>
Original text: äßüâ<br>
WBESC:($WBESC[äßüâ])<br>
</body>
</html>
```

After running this example the HTML code of resulting page in browser, may look like this:

```
<html>
<head>
<title>WBESC example</title>
</head>
<body>
Original text: äßüâ<br>
WBESC:(%E4%DF%FC%E2)<br>
</body>
</html>
```

21.24 $WBFN

Availability
$WBFN is available for use with all WBSP commands.

Syntax
$WBFN{*function(argument)*}
$WBFN[*function(arguments)*]

Parameters
function - valid function name (see table below)
argument - optional parameter. Contains valid function argument where needed (see explanation for each function).

Functions without arguments	Functions with arguments
DATE	ASC
DAY	BIN
HTPASS	CHR
HTUSER	FDT
MONTH	HEX
SECONDS	INT
TIME	LCS
USER	OCT
USERDATA	RND
WEEKDAYN	SQR
WEEKDAYS	UCS
YEAR	UTF

Returns
Function result, depending on function used.

Example

```
<html>
    <body>
    Hello!
    Your IP address is <b>$wbe[REMOTE_ADDR]</b>.<br>
    Today is $wbfn[weekdays], $wbfn[fdt(dd-mmm-yyyy)].
    </body>
</html>
```

Result:

Hello! Your IP address is **127.0.0.1**.
Today is Monday, 14-Apr-2008.

21.24.1 ASC - character's ASCII code

Availability
ASC is available for use with all WBSP commands.

Syntax
$WBFN{*ASC(character)*}
$WBFN[*ASC(character)*]

Arguments
character - any ASCII character.

Returns
An ASCII code of the character passed as an argument.

Example

```
<html>
<body>
ASCII code of letter A is $wbfn[asc(A)]!
</body>
</html>
```

Result

```
ASCII code of letter A is 65!
```

21.24.2 BIN - convert decimal number to binary

Availability
BIN is available for use with all WBSP commands.

Syntax
$WBFN{*BIN(decnumber)*}
$WBFN[*BIN(decnumber)*]

Arguments
decnumber - any integer in range 0 to 2147483647.

Returns
A string representing the binary value of the number passed as an argument.

Example

```
<html>
<body>
Binary value of letter A is $wbfn[bin(65)]!
</body>
</html>
```

Result

```
Binary value of letter A is 1000001!
```

21.24.3 CHR - print character with specified ASCII code

Availability
CHR is available for use with all WBSP commands.

Syntax
$WBFN{*CHR(decnumber)*}
$WBFN[*CHR(decnumber)*]

Arguments
decnumber - any integer in range 0 to 255.

Returns
A character with ASCII code *decnumber*.

Example

```
<html>
<body>
Character with ASCII code 65 is $wbfn[chr(65)]!
</body>
</html>
```

Result

```
Character with ASCII code 65 is A!
```

21.24.4 DATE - current system date

Availability
DATE is available for use with all WBSP commands.

Syntax
$WBFN{*DATE*}
$WBFN[*DATE*]

Arguments
DATE function does not have any arguments.

Returns
Current server date in short date format depending on server's regional settings.

21.24.5 DAY - current day of the month

Availability
DAY is available for use with all WBSP commands.

Syntax
$WBFN{*DAY*}
$WBFN[*DAY*]

Arguments
DAY function does not have any arguments.

Returns
A whole number between 1 and 31, inclusive, representing the day of the month for current server date.

21.24.6 FDT - date and time in specified format

Availability
FDT is available for use with all WBSP commands.

Syntax
$WBFN{*FDT(formatstring)*}
$WBFN[*FDT(formatstring)*]

Arguments
formatstring - any valid named or user-defined format string.

Returns
Current server date or time (depending on value of formatstring argument) formatted using formatstring.

Example

```
<html>
<body>
Today is $wbfn[fdt(dd-mmmm-yyyy)]!
</body>
</html>
```

Result

```
Today is 07-july-2008!
```

21.24.7 HEX - convert decimal number to hexadecimal

Availability
HEX is available for use with all WBSP commands.

Syntax
$WBFN{*HEX(decnumber)*}
$WBFN[*HEX(decnumber)*]

Arguments
decnumber - any integer in range 0 to 2147483647.

Returns
A string representing the hexadecimal value of the number passed as an argument.

Example

```
<html>
<body>
Hexadecimal value of ASCII code for letter A is $wbfn[HEX(65)]!
</body>
</html>
```

Result

```
Hexadecimal value of ASCII code of letter A is 41!
```

21.24.8 HTPASS - password used for authentication

Availability
HTPASS is available for use with all WBSP commands when authentication is used.

Syntax
$WBFN{*HTPASS*}
$WBFN[*HTPASS*]

Arguments
HTPASS function does not have any arguments.

Returns
Password used for authentication (WB_HTPass).

21.24.9 HTUSER - user name used for authentication

Availability
HTUSER is available for use with all WBSP commands when authentication is used.

Syntax
$WBFN{*HTUSER*}
$WBFN[*HTUSER*]

Arguments
HTUSER function does not have any arguments.

Returns
User name used for authentication (WB_HTUsr).

21.24.10 INT - integer portion of number

Availability
INT is available for use with all WBSP commands.

Syntax
$WBFN{*INT(decnumber)*}
$WBFN[*INT(decnumber)*]

Arguments
decnumber - any decimal number.

Returns
The integer portion of the number passed as an argument.

Example

```
<html>
<body>
Integer part of timer is $wbfn[INT($wbtimer[])]!
</body>
</html>
```

Result

```
Integer part of timer is 53021!
```

21.24.11 LCS - to lowercase

Availability
LCS is available for use with all WBSP commands.

Syntax
$WBFN{*LCS(anystring)*}
$WBFN[*LCS(anystring)*]

Arguments
anystring - any string value.

Returns
The string passed as an argument converted to lowercase characters.

Example

```
<html>
<body>
Lowercase of Hello World is $wbfn[LCS(Hello World)]!
</body>
</html>
```

Result

```
Lowercase of Hello World is hello world!
```

21.24.12 MONTH - current month

Availability
MONTH is available for use with all WBSP commands.

Syntax
$WBFN{*MONTH*}
$WBFN[*MONTH*]

Arguments
MONTH function does not have any arguments.

Returns
A whole number between 1 and 12, inclusive, representing the month of the year for current server date.

21.24.13 OCT - convert decimal number to octal

Availability
OCT is available for use with all WBSP commands.

Syntax
$WBFN{*OCT(decnumber)*}
$WBFN[*OCT(decnumber)*]

Arguments
decnumber - any integer in range 0 to 2147483647.

Returns
A string representing the octal value of the number passed as an argument.

Example

```
<html>
<body>
Octal value of letter A is $wbfn[OCT(65)]!
</body>
</html>
```

Result

```
Octal value of letter A is 101!
```

21.24.14 RND - random number

Availability
RND is available for use with all WBSP commands.

Syntax
$WBFN{*RND(decnumber)*}
$WBFN[*RND(decnumber)*]

Arguments
decnumber - any integer in range 1 to 2147483647.

Returns
The randomly generated number in range from 1 to the number passed as an argument.

Example

```
<html>
<body>
Random number:$wbfn[RND(1000)]!
</body>
</html>
```

Result

```
Random number:351!
```

21.24.15 SECONDS - seconds elapsed since midnight

Availability
SECONDS is available for use with all WBSP commands.

Syntax
$WBFN{*SECONDS*}
$WBFN[*SECONDS*]

Arguments
SECONDS function does not have any arguments.

Returns
A single-precision number representing the number of seconds elapsed since midnight (server time).

21.24.16 SQR - square root

Availability
SQR is available for use with all WBSP commands.

Syntax
$WBFN{*SQR(decnumber)*}
$WBFN[*SQR(decnumber)*]

Arguments
decnumber - any number in range from 0 to 2147483647.

Returns
The square root of the number passed as an argument.

Example

```
<html>
<body>
Square root of number 3.0625 is $wbfn[SQR(3.0625)]!
</body>
</html>
```

Result

```
Square root of number 3.0625 is 1.75!
```

21.24.17 TIME - current system time

Availability
TIME is available for use with all WBSP commands.

Syntax
$WBFN{*TIME*}
$WBFN[*TIME*]

Arguments
TIME function does not have any arguments.

Returns
Current server time in short format depending on server's regional settings.

21.24.18 UCS - to uppercase

Availability
UCS is available for use with all WBSP commands.

Syntax
$WBFN{*UCS(anystring)*}
$WBFN[*UCS(anystring)*]

Arguments
anystring - any string value.

Returns
The string passed as an argument converted to uppercase characters.

Example

```
<html>
<body>
Uppercase of Hello World is $wbfn[UCS(Hello World)]!
</body>
</html>
```

Result

```
Uppercase of Hello World is HELLO WORLD!
```

21.24.19 UNI - convert UTF-8 text to Unicode

Availability
UTF is available for use with all WBSP commands.

Syntax
$WBFN{*UNI(UTF-8 string)*}
$WBFN[*UNI(UTF-8 string)*]

Arguments
UTF-8 string - any string value in UTF-8 format.

Returns
The string passed as an argument converted to Unicode charset.

Example

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
This is original character in UTF-8 format:Ăź<br>
This is a converted character $wbfn[UNI(Ăź)]!
</body>
</html>
```

Result

```
This is original character in UTF-8 format: ß
This is a converted character: �!
```

21.24.20 USER - current database user name

Availability
USER is available for use with database related WBSP commands.

Syntax
$WBFN{*USER*}
$WBFN[*USER*]

Arguments
USER function does not have any arguments.

Returns
User name used to connect to the database.

21.24.21 USERDATA - retrieve value of WB_UserData

Availability
USERDATA is available for use with all WBSP commands when WB_UserData variable is defined.

Syntax
$WBFN{*USERDATA*}
$WBFN[*USERDATA*]

Arguments
USERDATA function does not have any arguments.

Returns
A processed value of WBSP system variable WB_UserData.

21.24.22 UTF - covert text to UTF-8

Availability
UTF is available for use with all WBSP commands.

Syntax
$WBFN{*UTF(anystring)*}
$WBFN[*UTF(anystring)*]

Arguments
anystring - any string value.

Returns
The string passed as an argument converted to UTF-8 charset.

Example

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
This is original character:ß<br>
This is a converted character $wbfn[UTF(ß)]!
</body>
</html>
```

Result

```
This is original character: □
This is a converted character: ß!
```

21.24.23 WEEKDAYN - day of the week (numeric value)

Availability
WEEKDAYN is available for use with all WBSP commands.

Syntax
$WBFN{*WEEKDAYN*}
$WBFN[*WEEKDAYN*]

Arguments
WEEKDAYN function does not have any arguments.

Returns
A whole number between 1 and 7, inclusive, representing the day of the week starting with Sunday (Sunday=1, Saturday=7) for current server date.

21.24.24 WEEKDAYS - day of the week (string value)

Availability
WEEKDAYS is available for use with all WBSP commands.

Syntax
$WBFN{*WEEKDAYS*}
$WBFN[*WEEKDAYS*]

Arguments
WEEKDAYS function does not have any arguments.

Returns
The name of the day of the week in English, for current server date.

21.24.25 YEAR - current year

Availability
YEAR is available for use with all WBSP commands.

Syntax
$WBFN{*YEAR*}
$WBFN[*YEAR*]

Arguments
YEAR function does not have any arguments.

Returns
A whole number representing the year for current server date.

21.25 $WBFOR - unconditional (for...next) loop

Availability
$WBFOR is available for use with all WBSP commands.

Syntax
$WBFOR{VarName|Start|End|***Step***|Content}
$WBFOR[VarName|Start|End|***Step***|Content]

Parameters
VarName - the name of the variable used as a loop counter. The variable can be accessed using $WBGETV function (see the example)
Start - initial value of loop counter.
End - final value of loop counter.
Step - optional parameter - amount counter is changed each time through the loop. It can be either positive (when start < end) or negative (when start > end). If not specified, step defaults to 1 (one).
Content - block of code (including WhizBase elements) that will be repeated through the loop

Returns
Code contained in *content* repeated through the loop.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
  <title>WBFOR</title>
</head>
<body>
$WBFOR[LCnt|20|2|-2|Counter value: $wbgetv[LCnt]<br>]
</body>
</html>
```

After running this example, the resulting page in browser, should look like this:

```
Counter value: 20
Counter value: 18
Counter value: 16
Counter value: 14
Counter value: 12
Counter value: 10
Counter value: 8
Counter value: 6
Counter value: 4
Counter value: 2
```

21.26 $WBFOREACH - loop through array elements

Availability
$WBFOREACH is available for use with all WBSP commands.

Syntax
$WBFOREACH{VarName|ArrayName|Content}
$WBFOREACH[VarName|ArrayName|Content]

Parameters
VarName - the name of the variable used to iterate through the elements of the *ArrayName* . The variable can be accessed using $WBGETV function (see the example)
ArrayName- the name of an array.
Content - block of code (including WhizBase elements) that will be repeated through the loop

Returns
Code contained in *content* repeated through the loop.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
  <title>WBFOREACH</title>
</head>
<body>
$WBSPLIT[Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday|Days|
,]
$wbforeach[Day|Days|$wbfn[ucs($wbgetv[Day])]<br>]
</body>
</html>
```

After running this example, the resulting page in browser, should look like this:

```
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
```

21.27 $WBFSIZE - file size

Availability
$WBFSIZE is available for use with all WBSP commands.

Syntax
$WBFSIZE{*filename*}
$WBFSIZE[*filename*]

Parameters
filename - any valid file name respecting WhizBase path rules

Returns
The size of requested file in bytes.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>File Size</title>
</head>
<body>
This file contains $wbfsize[$wbe[script_name]] bytes of code.
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
This file contains 203 bytes of code.
```

21.28 $WBFTIME - file time

Availability
$WBFTIME is available for use with all WBSP commands.

Syntax
$WBFTIME{*filename*}
$WBFTIME[*filename*]

Parameters
filename - any valid file name respecting WhizBase path rules

Returns
Time stamp of requested file

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>File date and time</title>
</head>
<body>
This file was last modified at $WBFTIME[$wbe[script_name]].
</body>
</html>
```

After running this example, the resulting page in browser should contain the date and time when a file was created or last modified.

21.29 $WBFUP - bytes uploaded

Availability
$WBFUP is available for use with all WBSP commands.

Syntax
$WBFUP{FULID}
$WBFUP[FULID]

Parameters
FULID - server-side generated unique form upload ID generated using WB_FULID variable and retrieved with $WBFULID tag.

Returns
Number of bytes that already have been uploaded by form identified with FULID.

Here's an example:

```
[FormFields]
wb_command=r
wb_showlogo=f
<!--WB_BeginTemplate-->
<html>
<head>
<meta http-equiv="refresh" content="1">
</head>
<body>
Uploaded:
$wbformat[$wbcalc[$wbfup[$wbv[fid]]/$wbfut[$wbv[fid]]*100]|0.00] %
```

```
</body>
</html>
```

After running this example (together with upload.wbsp file that can be found here), WhizBase will display the upload progress.

21.30 $WBFUT - bytes total

Availability
$WBFUT is available for use with all WBSP commands.

Syntax
$WBFUT{FULID}
$WBFUT[FULID]

Parameters
FULID - server-side generated unique form upload ID generated using WB_FULID variable and retrieved with $WBFULID tag.

Returns
Total bytes that should be uploaded by form identified with FULID.

Here's an example:

```
[FormFields]
wb_command=r
wb_showlogo=f
<!--WB_BeginTemplate-->
<html>
<head>
<meta http-equiv="refresh" content="1">
</head>
<body>
Uploaded:
$wbformat[$wbcalc[$wbfup[$wbv[fid]]/$wbfut[$wbv[fid]]*100]|0.00] %
</body>
</html>
```

After running this example (together with upload.wbsp file that can be found here), WhizBase will display the upload progress.

21.31 $WBGETATOM - get Atom feed

Availability
$WBGETATOM is available for use with all WBSP commands.

Syntax
$WBGETATOM{*URL*|*ArrName*|***limit***|***show***}
$WBGETATOM[*URL*|*ArrName*|***limit***|***show***]

Parameters
URL - The Internet address of the Atom feed you want to include in your page. It can be either absolute (containing entire address including protocol, server, path and resource) or relative (to document root of virtual server if it starts with slash /

character, or to the directory where the WBSP file is located).
ArrName - name that will be used as prefix for all variables where Atom feed content will be stored.
limit - optional parameter. Contains the number of items to be included in arrays
show - optional parameter - if set to true (T,ON,1) WhizBase will show the last index (length-1) of resulting array

Returns
Set of five (5) arrays:
ArrName_title() - containing values of all <title> nodes
ArrName_summary() - containing values of all <summary> nodes
ArrName_updated() - containing values of all <updated> nodes
ArrName_content() - containing values of all <content> nodes
ArrName_link - set of four (4) sub arrays:
ArrName_link_replies() - containing values of all <link> nodes with rel atribute set to "replies"
ArrName_link_edit() - containing values of all <link> nodes with rel atribute set to "edit"
ArrName_link_self() - containing values of all <link> nodes with rel atribute set to "self" or empty ("")
ArrName_link_alternate() - containing values of all <link> nodes with rel atribute set to "alternate"

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
  <title>Atom</title>
</head>
<body>
$wbsetv[lastAtom|$wbgetatom[http://www.blogger.com/feeds/64695940303292
84280/posts/default|AtomFeed|5|T]]
$wbsetv[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=$wbgetv[lastAtom]|
<a target="_blank">$wbgetv[AtomFeed_Title($wbgetv[loopcounter])]</a>
$wbgetv[AtomFeed_Updated($wbgetv[loopcounter])]<br>
$wbsetv[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
</body>
</html>
```

After running this example (retrieving posts from blog.whizbase.com), the resulting page in browser may look like this:

Sending an email with embedded images using WhizBase 2011-08-11T11:39:48.072+02:00
How to collect data from web visitors and save them in your database 2011-05-25T09:53:37.552+02:00
Whiz your database to the web, now as a SaaS! 2011-05-18T14:57:12.996+02:00
Create an AJAX supported registration form with WhizBase 2011-03-30T10:43:08.184+02:00
Whizbase fast tutorial 2011-03-23T11:40:35.143+01:00

21.32 $WBGC - get cookie

Availability
$WBGC is available for use with all WBSP commands.

Syntax
$WBGC{*cookiename*}
$WBGC[*cookiename*]

Parameters
cookiename - name of the cookie

Returns
Value of the HTTP cookie specified with *cookiename*.

Example

File GetCookies.wbsp

```
[FormFields]
wb_command=R
wb_showlogo=F
<!--WB_BeginTemplate-->
<html>
<head><br><LINK$wbgc[css]"=""|style1.css|$wbgc[css]]" type=text/css
rel=stylesheet>
<title>Cookie test</title>
</head>
<body>
Cookievalue:($wbgc[css])<br><br><ahref= "setcookie.wbsp?css=
style1.css">Style1</a><br><br><a>Style 2</a>
</body>
</html>
```

File SetCookie.wbsp

```
[FormFields]
wb_command=R
wb_redirect=$wbe[http_referer]
wb_addcookie=CSS=$wbv{CSS}
```

File style1.css

```
body{
font-family:Verdana;
font-size:12px;
color:#333333;
background-color:#efefef;
}
a{
text-decoration:none;
color:#0065b7;
font-weight:bold;
}
a:hover{
color:#cc0000;
}
```

File style2.css

```
body{
font-family:Verdana;
font-size:12px;
color:#0000cc;
background-color:#ffffff;
}
a{
text-decoration:none;
color:#6500b7;
font-weight:bold;
}
a:hover{
color:#00cc00;
}
```

After running this example (file getcookies.wbsp) the page will change its appearance depending on value of cookie named CSS (changed by clicking on links "Style 1" and "Style 2").

21.33 $WBGETRSS - get RSS feed

Availability
$WBGETRSS is available for use with all WBSP commands.

Syntax
$WBGETRSS{*URL*|*ArrName*|***limit***|***show***}
$WBGETRSS[*URL*|*ArrName*|***limit***|***show***]

Parameters
URL - The Internet address of the RSS feed you want to include in your page. It can be either absolute (containing entire address including protocol, server, path and resource) or relative (to document root of virtual server if it starts with slash / character, or to the directory where the WBSP file is located).
ArrName - name that will be used as prefix for all variables where RSS feed content will be stored.
limit - optional parameter. Contains the number of items to be included in arrays
show - optional parameter - if set to true (T,ON,1) WhizBase will show the last index (length-1) of resulting array

Returns
Set of four (4) arrays:
ArrName_title() - containing values of all <title> nodes
ArrName_link() - containing values of all <link> nodes
ArrName_description() - containing values of all <description> nodes
ArrName_pubdate() - containing values of all <pubDate> nodes

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
  <title>RSS</title>
</head>
<body>
$wbsetv[lastRSS|$wbgetrss[rss.xml|RssFeed|5|T]]
$wbsetv[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=$wbgetv[lastRSS]|
<a target="_blank">$wbgetv[RssFeed_Title($wbgetv[loopcounter])]</a>
$wbgetv[RssFeed_pubdate($wbgetv[loopcounter])]<br>
$wbsetv[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
</body>
</html>
```

After running this example, the resulting page in browser may look like this:

```
18.12.2009 - Version 5.0.15 released 18.12.2009 20:30:00 CET
22.09.2009 - Version 5.0.14 released 22.09.2009 14:34:58 CET
15.07.2009 - Version 5.0.13 released 15.07.2009 12:16:44 CET
08.07.2009 - PowerPack Wizards version 1.0.3 released 08.07.2009 10:11:43 CET
07.07.2009 - Version 5.0.12 released 07.07.2009 14:50:00 CET
```

21.34 $WBGETURL - get data from URL (GET method)

Availability
$WBGETURL is available for use with all WBSP commands.

Syntax
$WBGETURL{*URL*|***FullPage***}
$WBGETURL[*URL*|***FullPage***]

Parameters
URL - The Internet address of the resource (page, file) you want to include in your page. It can be either absolute (containing entire address including protocol, server, path and resource) or relative (to document root of virtual server if it starts with slash / character, or to the directory where the WBSP file is located).
FullPage - optional parameter. If set to true (T) function will return entire page code without truncating code outside <body> and </body> tags.

Returns
Source code of the received page/file. If source code contains <body> and </body> tags and FullPage is not set to true, function will include only the source code between these tags.

Example
File GetCookiesA.wbsp

```
[FormFields]
<!--WB_BeginTemplate-->
<html>
<head>
<LINK $wbgc[css]"=""
|
$wbif["$wbrv[style]"=""|style1.css|$wbrv[style]]
|
$wbif["$wbv[css]"=""|style1.css|$wbgeturl[changecss.wbsp?css=$wbv[css]]
]
]" type=text/css rel=stylesheet>
<title>Cookie test</title>
</head>
<body>
Cookie value:($wbgc[css])<br>
Variablevalue:($wbrv[style])<br><br><a >Style1</a><br><br><a>Style
2</a>
</body>
</html>
```

File SetCookieA.wbsp

```
[FormFields]
wb_command=R
wb_redirect=http://$wbe[server_name]$wbv[sp]?css=$wbv[css]
wb_addcookie=CSS=$wbv{CSS}
```

File ChangeCss.wbsp

```
<!--
[FormFields]
wb_filename=/default.inc
wb_command=wf
wb_keyname=Style
wb_keyvalue=$wbv{css}
wb_section=Userdata
-->
<!--WB_BeginTemplate-->$wbv[css]
```

File style1.css

```
body{
font-family:Verdana;
font-size:12px;
color:#333333;
background-color:#efefef;
}
a{
text-decoration:none;
color:#0065b7;
font-weight:bold;
}
a:hover{
```

```
color:#cc0000;
}
```

File style2.css

```
body{
font-family:Verdana;
font-size:12px;
color:#0000cc;
background-color:#ffffff;
}
a{
text-decoration:none;
color:#6500b7;
font-weight:bold;
}
a:hover{
color:#00cc00;
}
```

After running this example (file getcookiesa.wbsp) the page will change its appearance depending on value of cookie named CSS (if it exists) or depending on a value of user defined variable Style in file /default.inc, or if neither of those two exists, it will use file style1.css. When you click the either of links ("Style 1" and "Style 2") it will set the cookie to selected value, call the file changecss.wbsp (with selected value as parameter named CSS) using $WBGETURL function, and write new value to file /default.inc as user-defined variable "Style".

21.35 $WBGETV - get value of WB variable or array element

Availability
$WBGETV is available for use with all WBSP commands.

Syntax
$WBGETV{*varname*}
$WBGETV[*varname*]

Parameters
varname - the name of the variable

Returns
Value of requested WBSP variable

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
```

```
$WBSETV[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=10|
The loopcounter value is:$wbgetv[loopcounter]<br>
$WBSETV[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
Loop ended, loopcounter value is $wbgetv[loopcounter]!
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The loopcounter value is:0
The loopcounter value is:1
The loopcounter value is:2
The loopcounter value is:3
The loopcounter value is:4
The loopcounter value is:5
The loopcounter value is:6
The loopcounter value is:7
The loopcounter value is:8
The loopcounter value is:9
The loopcounter value is:10
Loop ended, loopcounter value is 11!
```

21.36 $WBGETXML - get XML

Availability
$WBGETXML is available for use with all WBSP commands.

Syntax
$WBGETXML{*Source*|*ArrName*|*NodeName*|***limit***|***show***}
$WBGETXML[*Source*|*ArrName*|*NodeName*|***limit***|***show***]

Parameters
Source - Either Internet address of the XML file you want to include in your page (URL) or a string value containing XML code. URL can be either absolute (containing entire address including protocol, server, path and resource) or relative (to document root of virtual server if it starts with slash / character, or to the directory where the WBSP file is located). XML source code must start with **"<?xml"** (without quotations)
ArrName - name that will be used as prefix for all variables where XML node values will be stored.
NodeName - Name of the parent node for which all child nodes will be retrieved
limit - optional parameter. Contains the number of items to be included in arrays
show - optional parameter - if set to true (T,ON,1) WhizBase will show the last index (length-1) of resulting array

Returns
Set of arrays containing values of all child nodes of *NodeName* of as well as array containing all the parent node (*NodeName*) values (all child nodes in one line separated by space). Arrays are named following this convention: ArrName_NodeName() and ArrName_NodeName_ChildNodeName(). See the example.

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
  <title>GetXML</title>
</head>
<body>
$wbsetv[xmlcnt|$wbgetxml[XMLSample.xml|conditions|condition|5|T]]
$wbsetv[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=$wbgetv[xmlcnt]|
No.: $wbgetv[loopcounter]<br/>
Condition (full parent node):
<b>$wbgetv[conditions_condition($wbgetv[loopcounter])]</b><br/>
Code: $wbgetv[conditions_condition_code($wbgetv[loopcounter])]<br/>
Day_Icon:
$wbgetv[conditions_condition_day_icon($wbgetv[loopcounter])]<br/>
Night_Icon:
$wbgetv[conditions_condition_night_icon($wbgetv[loopcounter])]<br/>
Description:
$wbgetv[conditions_condition_description($wbgetv[loopcounter])]<br/>
$wbsetv[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
</body>
</html>
```

File XMLSample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<codes>
  <condition>
    <code>XXX</code>
    <description>XXX description</description>
    <day_icon>XXXDayIcon.png</day_icon>
    <night_icon>XXXNightIcon.png</night_icon>
  </condition>
  <condition>
    <code>YYY</code>
    <description>YYY description</description>
    <day_icon>DayYYYIcon.png</day_icon>
    <night_icon>NightYYYIcon.png</night_icon>
  </condition>
  <condition>
    <code>ZZZ</code>
    <description>ZZZ description</description>
    <day_icon>DayIconZZZ.png</day_icon>
    <night_icon>NightIconZZZ.png</night_icon>
  </condition>
</codes>
```

After running this example, the resulting page in browser should look like this:

No.: 0
Condition (full parent node): **XXX XXX description XXXDayIcon.png XXXNightIcon.png**
Code: XXX
Day_Icon: XXXDayIcon.png
Night_Icon: XXXNightIcon.png
Description: XXX description
No.: 1
Condition (full parent node): **YYY YYY description DayYYYIcon.png NightYYYIcon.png**
Code: YYY
Day_Icon: DayYYYIcon.png
Night_Icon: NightYYYIcon.png
Description: YYY description
No.: 2
Condition (full parent node): **ZZZ ZZZ description DayIconZZZ.png NightIconZZZ.png**
Code: ZZZ
Day_Icon: DayIconZZZ.png
Night_Icon: NightIconZZZ.png
Description: ZZZ description

21.37 $WBIF - conditionally execute statements

Availability
$WBIF is available for use with all WBSP commands.

Syntax
$WBIF{*expression*|*true part*|*false part*}
$WBIF[*expression*|*true part*|*false part*]

Parameters
expression - expression you want to evaluate. If it contains more conditions connected with keyword AND or keyword OR, single condition has to be enclosed in brackets - (condition 1) or (condition 2)
true part - value that will be returned if expression is True
false part - value that will be returned if expression is False

Returns
Code contained in true part or false part depending on result of evaluated expression.

Example

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
-->
<!--WB_BeginTemplate-->
<html>
```

```
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr
bgcolor="#$wbif[$wbcalc[$wbp[rn]/2]=$wbcalc[$wbp[rn]\2]|c0c0c0|ff6699]"
>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Year published	Title	ISBN
2001	McGraw-Hill's Encyclopedia of Networking & Telecommunications	0072120053
2000	Microsoft SMS Installer	0072124474
2001	Windows 2000 Iis 5.0 : A Beginner's Guide	0072133724
1996	Windows Nt Security Handbook	0078822408
1998	Microsoft Internet Information Server 4: the Complete Reference	0078824575

21.38 $WBINC - include file

Important: $WBINC function is processed before any other function, tag or section. Therefore it can not accept other WhizBase functions as arguments, nor it can be used as an argument for other WhizBase functions. For situations where it is required (e.g. to use Whizbase functions in a file name, or to use include file as a part of $WBIF or $WBCASE functions, etc.) use $WBRINC.

Availability
$WBINC is available for use with all WBSP commands.

Syntax
$WBINC{*includefilename*}
$WBINC[*includefilename*]

Parameters
includefilename - full path and file name of the file that should be included. Included file **can** contain <!--WB_BeginDetail--> and <!--WB_EndDetail--> comments.

Returns
Source code of the file *includefilename*. If source code contains <body> and </body> tags, function will include only the source code between these tags.

Example

```
[FormFields]
wb_command=R
wb_showlogo=F
<!--WB_BeginTemplate-->
<html>
<body>
$WBINC[menu.ic]
</body>
</html>
```

File menu.ic

```
<html>
<body>
<a >Link 1</a><br>
<a >Link 2</a><br>
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

```
Link 1
Link 2
```

21.39 $WBIRUN - execute inline script

Important: $WBIRUN function uses safe subset which means that scripting engine is only allowed to create or use objects that are marked safe for scripting. In order to start this function WBSP file must have WB_AllowInlineFunc set to TRUE.

Availability
$WBIRUN is available for use with all WBSP commands in scripting languages listed in Execute variable of WBSP.ssc file.

Syntax
$WBIRUN{*Language*|*function(fnargs)*|*ScriptCode*}
$WBIRUN[*Language*|*function(fnargs)*|*ScriptCode*]

Parameters
language - name of the scripting language used in script (e.g. JavaScript, VBScript, JScript, etc.)
function - name of the function that should be executed - fnargs are arguments required by the script function, not by the WBSP
scriptcode - actual script code that should be executed

Returns
Result returned by script function.

Example

```
[FormFields]
wb_command=R
wb_showlogo=F
wb_allowinlinefunc=T
<!--WB_BeginTemplate-->
<html>
<body>
This server uses $wbirun[VBScript|GetScriptEngineInfo|
Function GetScriptEngineInfo
   s = ""
   s = ScriptEngine & " Version "
   s = s & ScriptEngineMajorVersion & "."
   s = s & ScriptEngineMinorVersion & "."
   s = s & ScriptEngineBuildVersion
   GetScriptEngineInfo = s
End Function
]
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

```
This server uses VBScript Version 5.8.16990
```

21.40 $WBIV - increment value

Important: $WBIV function has effect on entire WBSP page and all it's sub elements (configuration section, included files, sub reports, etc.). All instances of $WBIV function that use the same variable name will set same variable regardless of their location inside WBSP page (e.g. $WBIV[somevar] in main WBSP page and $WBIV[somevar] in subreport will change (increment by 1) the value of the same variable named **somevar**.

Availability
$WBIV is available for use with all WBSP commands.

Syntax
$WBIV{*varname*|***increment***|***showvar***}
$WBIV[*varname*|***increment***|***showvar***]

Parameters
varname - the name of global WBSP variable to be incremented
increment - optional parameter - the value that will be added to the existing value of *varname*. Default value is 1.
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the assigned value

Returns
New (incremented) value if showvar parameter is set to true, otherwise it returns nothing, just sets the value of a variable.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBIV</title>
</head>
<body>
$WBSETV[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=10|
The loopcounter value is:$wbgetv[loopcounter]<br>
$WBIV[loopcounter]
]
Loop ended, loopcounter value is $wbgetv[loopcounter]!
</body>
</html>
```

After running this example, the resulting page in browser may look like this:

```
The loopcounter value is:0
The loopcounter value is:1
The loopcounter value is:2
The loopcounter value is:3
The loopcounter value is:4
The loopcounter value is:5
The loopcounter value is:6
The loopcounter value is:7
The loopcounter value is:8
The loopcounter value is:9
The loopcounter value is:10
Loop ended, loopcounter value is 11!
```

21.41 $WBJSON - get value of JSON object

Availability
$WBJSON is available for use with all WBSP commands.

Syntax
$WBJSON{JSONSource|JSONObjectName}
$WBJSON[JSONSource|JSONObjectName]

Parameters
JSONSource - String value containing JSON to be searched. Note that this is not an URL of the JSON file but JSON source itself.
JSONObjectName - Any valid object in given JSON code including array elements. When accessing array elements use brackets () instead of square brackets []. **Note**

that JSON object names are case sensitive.

Returns
Value of JSONObjectName.

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
  <title>WBJSON</title>
</head>
<body>
$wbsetv[JSON|$wbrinc[sample.json]]
Glossary title: $wbjson[$wbgetv[json]|glossary.title]<br>
See also:
$wbjson[$wbgetv[json]|glossary.GlossDiv.GlossList.GlossEntry.GlossDef.G
lossSeeAlso(0)],
$wbjson[$wbgetv[json]|glossary.GlossDiv.GlossList.GlossEntry.GlossDef.G
lossSeeAlso(1)]
</body>
</html>
```

File Sample.json

```
{
    "glossary":
     {
      "title": "example glossary",
      "GlossDiv":
       {
        "title": "S",
        "GlossList":
         {
          "GlossEntry":
           {
            "ID": "SGML",
            "SortAs": "SGML",
            "GlossTerm": "Standard Generalized Markup Language",
            "Acronym": "SGML",
            "Abbrev": "ISO 8879:1986",
            "GlossDef":
             {
              "para": "A meta-markup language, used to create markup
languages such as DocBook.",
              "GlossSeeAlso": ["GML", "XML"]
             },
            "GlossSee": "markup"
           }
         }
       }
     }
}
```

After running this example, the resulting page in browser should look like this:

```
Glossary title: example glossary
See also: GML, XML
```

21.42 $WBJSONELEM - get element names of JSON object

Availability
$WBJSONELEM is available for use with all WBSP commands.

Syntax
$WBJSONELEM{JSONSource|JSONObjectName}
$WBJSONELEM[JSONSource|JSONObjectName]

Parameters
JSONSource - String value containing JSON to be searched. Note that this is not an URL of the JSON file but JSON source itself.
JSONObjectName - Any valid object in given JSON code including array elements. When accessing array elements use brackets () instead of square brackets []. **Note that JSON object names are case sensitive.**

Returns
Comma-separated list of element names of JSONObjectName.

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
  <title>WBJSONELEM</title>
</head>
<body>
$wbsetv[JSON|$wbrinc[sample.json]]
Elements of GlossEntry:
$wbjsonelem[$wbgetv[json]|glossary.GlossDiv.GlossList.GlossEntry]
</body>
</html>
```

File Sample.json

```
{
    "glossary":
     {
      "title": "example glossary",
      "GlossDiv":
       {
        "title": "S",
        "GlossList":
         {
          "GlossEntry":
           {
            "ID": "SGML",
            "SortAs": "SGML",
            "GlossTerm": "Standard Generalized Markup Language",
            "Acronym": "SGML",
            "Abbrev": "ISO 8879:1986",
            "GlossDef":
             {
              "para": "A meta-markup language, used to create markup
languages such as DocBook.",
              "GlossSeeAlso": ["GML", "XML"]
             },
            "GlossSee": "markup"
           }
         }
       }
     }
}
```

After running this example, the resulting page in browser should look like this:

```
Elements of GlossEntry: ID,SortAs,GlossTerm,Acronym,Abbrev,GlossDef,GlossSee
```

21.43 $WBJSONLEN - length of JSON object

Availability
$WBJSONLEN is available for use with all WBSP commands.

Syntax
$WBJSONLEN{JSONSource|JSONArrayName}
$WBJSONLEN[JSONSource|JSONArrayName]

Parameters
JSONSource - String value containing JSON to be searched. Note that this is not an URL of the JSON file but JSON source itself.
JSONArrayName - The name of the array. Do not use brackets () or square brackets [] in array name. **Note that JSON array names are case sensitive.**

Returns
Number of elements of JSONArrayName.

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
  <title>WBJSONLEN</title>
</head>
<body>
$wbsetv[JSON|$wbrinc[sample.json]]
Number of elements in "See also" array:
$wbjsonlen[$wbgetv[json]|glossary.GlossDiv.GlossList.GlossEntry.GlossDe
f.GlossSeeAlso]
```

```
</body>
</html>
```

File Sample.json

```
{
    "glossary":
     {
      "title": "example glossary",
      "GlossDiv":
       {
        "title": "S",
        "GlossList":
         {
          "GlossEntry":
           {
            "ID": "SGML",
            "SortAs": "SGML",
            "GlossTerm": "Standard Generalized Markup Language",
            "Acronym": "SGML",
            "Abbrev": "ISO 8879:1986",
            "GlossDef":
             {
              "para": "A meta-markup language, used to create markup
languages such as DocBook.",
              "GlossSeeAlso": ["GML", "XML"]
             },
            "GlossSee": "markup"
           }
         }
       }
     }
}
```

After running this example, the resulting page in browser should look like this:

```
Number of elements in "See also" array: 2
```

21.44 $WBPOSTURL - get data from URL (POST method)

Availability
$WBPOSTURL is available for use with all WBSP commands.

Syntax
$WBPOSTURL{*URL*|*PostData*|***FullPage***|***ContentType***|***SoapAction***|***Options***|***Headers***}
$WBPOSTURL[*URL*|*PostData*|***FullPage***|***ContentType***|***SoapAction***|***Options***|***Headers***]

Parameters
URL - The Internet address of the resource (page, file) you want to include in your page. It can be either absolute (containing entire address including protocol, server, path and resource) or relative (to document root of virtual server if it starts with slash / character, or to the directory where the WBSP file is located).
PostData - data to be posted in format var1=value 1&var2= value 2&...&varn= value

n for ordinary POST or as XML for SOAP.
FullPage - optional parameter. If set to true (T) function will return entire page code without truncating code outside <body> and </body> tags.
ContentType - optional parameter. Sets the Content-Type clause in HTTP request header. If omitted WhizBase will send the default value **application/x-www-form-urlencoded**
SoapAction - optional parameter. Sets the SOAPAction clause in HTTP request header for web services. If omitted no default value will be sent.
Options - optional parameter. Contains comma-separated list of ServerXMLHTTP options in form optionnumber=optionvalue.
Headers - optional parameter. Contains comma-separated list of ServerXMLHTTP request headers in form headername= headervalue

Returns
Source code of the received page/file. If source code contains <body> and </body> tags and FullPage is not set to true, function will include only the source code between these tags.

Example
File PostCookiesA.wbsp

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
<LINK $wbgc[css]"=""
|
$wbif["$wbrv[style]"=""|style1.css|$wbrv[style]]
|
$wbif["$wbv[css]"=""|style1.css|$wbposturl[changecss.wbsp|css=$wbv[css]
]]
]" type=text/css rel=stylesheet>
<title>Cookie test</title>
</head>
<body>
Cookie value:($wbgc[css])<br>
Variable value:($wbrv[style])<br><br>
<form action="setcookieA.wbsp" method="post">
<input type="hidden" name="css" value="style1.css">
<input type="hidden" name="sp" value="$wbe[script_name]">
<input type="submit" value="Style 1">
</form>
<form action="setcookieA.wbsp" method="post">
<input type="hidden" name="css" value="style2.css">
<input type="hidden" name="sp" value="$wbe[script_name]">
<input type="submit" value="Style 2">
</form>
</body>
</html>
```

File SetCookieA.wbsp

```
[FormFields]
wb_command=R
wb_redirect=http://$wbe[server_name]$wbv[sp]?css=$wbv[css]
wb_addcookie=CSS=$wbv{CSS}
```

File ChangeCss.wbsp

```
<!--
[FormFields]
wb_filename=/default.inc
wb_command=wf
wb_keyname=Style
wb_keyvalue=$wbv{css}
wb_section=Userdata
-->
<!--WB_BeginTemplate-->$wbv[css]
```

File style1.css

```
body{
font-family:Verdana;
font-size:12px;
color:#333333;
background-color:#efefef;
}
a{
text-decoration:none;
color:#0065b7;
font-weight:bold;
}
a:hover{
color:#cc0000;
}
```

File style2.css

```
body{
font-family:Verdana;
font-size:12px;
color:#0000cc;
background-color:#ffffff;
}
a{
text-decoration:none;
color:#6500b7;
font-weight:bold;
}
a:hover{
color:#00cc00;
}
```

After running this example (file postcookiesa.wbsp) the page will change its appearance depending on value of cookie named CSS (if it exists) or depending on a value of user defined variable Style in file /default.inc, or if neither of those two exists, it will use file style1.css. When you click the either of buttons ("Style 1" and "Style 2") it will set the cookie to selected value, call the file changecss.wbsp (with selected value as parameter named CSS) using $WBPOSTURL function, and write new value to file /default.inc as user-defined variable "Style".

21.45 $WBRENDER - process WhizBase code

Availability
$WBRENDER is available for use with all WBSP commands:

Syntax
$WBRENDER{*anystring*}
$WBRENDER[*anystring*]

Parameters
anystring - any string containing WhizBase tag(s) and function(s).

Returns
The rendered value of *anystring*. This means that WhizBase will process any tag and/or function contained in *anystring* before it sends it to the client

<BEXAMPLE< B>

```
<!--
[FormFields]
wb_command=q
wb_basename=biblioA.mdb
wb_rcdset=titles
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBRENDER example</title>
</head>
<body>
$WBRENDER[$wbsrq[getrep.sr|id=$wbfn[HTUser]]]
</body>
</html>
```

If we assume that there is another table in the database that contains different WhizBase and HTML code for the body of the report for every user then getrep.sr subreport should return a **WhizBase** code for displaying the contents of the "titles" table in "biblioA.mdb" database. So, calling subreport alone will return **unprocessed** WhizBase code, but using $WBRF in subreport will produce the error (Item not found in this collection). So, subreport **MUST NOT** process the code, but return it to the main report. There we use $WBRENDER to instruct the WhizBase to process the code in main report. This is just one of many possible usages of $WBRENDER function.

21.46 $WBRINC - include file

Availability
$WBRINC is available for use with all WBSP commands.

Syntax
$WBRINC{*includefilename*}
$WBRINC[*includefilename*]

Parameters
includefilename - full path and file name of the file that should be included. Included file **can NOT** contain <!--WB_BeginDetail--> and <!--WB_EndDetail--> comments.

Returns
Source code of the file *includefilename*. If source code contains <body> and </body> tags, function will include only the source code between these tags.

Example

```
[FormFields]
wb_command=R
wb_showlogo=F
<!--WB_BeginTemplate-->
<html>
<body>
$WBRINC[menu.ic]
</body>
</html>
```

File menu.ic

```
<html>
<body>
<a >Link 1</a><br>
<a >Link 2</a><br>
</body>
</html>
```

After running this example, the resulting page in web browser may look like this:

```
Link 1
Link 2
```

21.47 $WBRNDSTR - randomly generated string

Availability
$WBRNDSTR is available for use with all WBSP commands.

Syntax
$WBRNDSTR{*length*|***type***}
$WBRNDSTR[*length*|***type***]

Parameters
length - the length of a generated random string in characters
type - optional parameter - defines which characters will be used:
A - Alphanumeric uppercase (A-Z and 0-9)
AM - Alphanumeric (A-Z a-z and 0-9)
AL - Alphanumeric lowercase (a-z and 0-9)
L - Letters uppercase (A-Z)

LM - Letters (A-Z a-z)
LL - Letters lowercase (a-z)
N - Numbers (0-9)

Returns
Randomly generated string.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBRNDSTR</title>
</head>
<body>
$WBRNDSTR[20|A]<br>
$WBRNDSTR[20|L]<br>
$WBRNDSTR[20|N]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
V081XDSMKE21TE4VGABZ
ZQTLBHOJSDQJBJRLQKJD
33863882287748758754
```

21.48 $WBROUND - rounds number value to specified number of decimal places

Availability
$WBROUND is available for use with all WBSP commands.

Syntax
$WBROUND{*number*|***decimals***}
$WBROUND[*number*|***decimals***]

Parameters
number - the number to be rounded
decimals - optional parameter containing number of decimal places. Default value is 0 (round to integer).

Returns
Number value rounded to specified decimal places.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBRound</title>
</head>
<body>
Original number: $wbsetv[number|41488.53617|T]<br>
Integer: $wbround[$wbgetv[number]]<br>
Two decimal places: $wbround[$wbgetv[number]|2]<br>
Three decimal places: $wbround[$wbgetv[number]|3]<br>
Four decimal places: $wbround[$wbgetv[number]|4]
</body>
</html>
```

After running this example you should get the following:

```
Original number: 41488.53617
Integer: 41489
Two decimal places: 41488.54
Three decimal places: 41488.536
Four decimal places: 41488.5362
```

21.49 $WBRRV - read and render configuration variable

Availability
$WBRRV is available for use with all WBSP commands:

Syntax
$WBRRV{*varname*}
$WBRRV[*varname*]

Parameters
varname - the name of requested variable defined in configuration section. It can be either WhizBase variable or any variable defined in any Userdata section (either in WBSP file itself or in include files)

Returns
The value of the variable *varname* with all WhizBase functions and tags processed

Example

```
<!--
[FormFields]
wb_command=q
wb_basename=biblioA.mdb
wb_rcdset=titles
wb_query=id=$wbfn{rnd(28)}
-->
```

```
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBRRV example</title>
</head>
<body>
WBRV:$WBRV[wb_query]<br>
WBRRV:$WBRRV[wb_query]<br>
WBV:$wbv[wb_query]<br>
WBQuery:$wbquery
$wbdetail[f]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
WBRV:id=$wbfn{rnd(28)}
WBRRV:id=5
WBV:
WBQuery:id=13



Title          Fireworks 4 for Windows and Macintosh Visual Quickstart Guide
Year Published 2001
ISBN           0201731339
PubID          1
AU_ID          2
imageURL       0201731339.jpg
Qty            100
Price          100
ID             13
```

In this example we can see that $WBRV will return exact content of the variable, as it is written in file. $WBV function will return an empty string, because WB_Query is not sent as form variable, but defined in file. $WBRRV will return processed value of the variable, **but the processing will take place in the moment of executing $WBRRV function**, so returned value for $wbfn{rnd(28)} will not be the same as used to select the record ($WBQuery contains correct value).

21.50 $WBRUN - execute external script

Availability
$WBRUN is available for use with all WBSP commands in scripting languages listed in Execute variable of WBSP.ssc file.

Syntax
$WBRUN{*scriptfilename*|*Language*|*function(fnargs)*}
$WBRUN[*scriptfilename*|*Language*|*function(fnargs)*]

Parameters
scriptfilename - full path and file name of the file where script code is stored
language - name of the scripting language used in script (e.g. JavaScript, VBScript, JScript, etc.)
function - name of the function that should be executed - fnargs are arguments required by the script function, not by the WBSP

Returns
Result returned by script function.

Example

```
[FormFields]
wb_command=R
wb_showlogo=F
<!--WB_BeginTemplate-->
<html>
<body>
File size for biblio.mdb:
$wbrun[getfilesize.vbs|VBScript|getFSize("$wbcurrdir[]biblio.mdb")]
</body>
</html>
```

File getfilesize.vbs

```
Function getFSize(fname)
Dim fso
Dim fil
Set fso = CreateObject("Scripting.FileSystemObject")
set fil = fso.GetFile(fname)
sizebytes=fil.size
if sizebytes<1024 then getFSize=sizebytes & " bytes": exit function
if sizebytes<1024^2 then getFSize=formatnumber(sizebytes/1024,2) & "
KB": exit function
getFSize=formatnumber(sizebytes/1024^2,2) & " MB"
end function
```

After running this example, the resulting page in web browser may look like this:

```
File size for biblio.mdb: 112,00 KB
```

21.51 $WBRV - read configuration variable

Availability
$WBRV is available for use with all WBSP commands:

Syntax
$WBRV{*varname*}
$WBRV[*varname*]

Parameters
varname - the name of requested variable defined in configuration section. It can be either WhizBase variable or any variable defined in any Userdata section (either in

WBSP file itself or in include files)

Returns
The value of the variable *varname*

Example

```
<!--
[FormFields]
wb_command=q
wb_basename=biblioA.mdb
wb_rcdset=titles
wb_query=id=$wbfn{rnd(28)}
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBRV example</title>
</head>
<body>
WBRV:$wbrv[wb_query]<br>
WBRRV:$wbrrv[wb_query]<br>
WBV:$wbv[wb_query]<br>
WBQuery:$wbquery
$wbdetail[f]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
WBRV:id=$wbfn{rnd(28)}
WBRRV:id=5
WBV:
WBQuery:id=13



Title          Fireworks 4 for Windows and Macintosh Visual Quickstart Guide
Year Published 2001
ISBN           0201731339
PubID          1
AU_ID          2
imageURL       0201731339.jpg
Qty            100
Price          100
ID             13
```

In this example we can see that $WBRV will return exact content of the variable, as it is written in file. $WBV function will return an empty string, because WB_Query is not sent as form variable, but defined in file. $WBRRV will return processed value of the variable, **but the processing will take place in the moment of executing**

$WBRRV function, so returned value for $wbfn{rnd(28)} will not be the same as used to select the record ($WBQuery contains correct value).

21.52 $WBSETV - set value of WB variable

Important: $WBSETV function has effect on entire WBSP page and all it's sub elements (configuration section, included files, sub reports, etc.). All instances of $WBSETV function that use the same variable name will set same variable regardless of their location inside WBSP page (e.g. $WBSETV[somevar|0] in main WBSP page and $WBSETV[somevar|$wbp[RC]] in subreport will change the value of the same variable named **somevar**.

Availability
$WBSETV is available for use with all WBSP commands.

Syntax
$WBSETV{*varname*|*varvalue*|***showvar***}
$WBSETV[*varname*|*varvalue*|***showvar***]

Parameters
varname - the name of global WBSP variable to which the new value will be assigned
varvalue - the value that will be assigned
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the assigned value

Returns
Assigned value if showvar parameter is set to true, otherwise it returns nothing, just sets the value of a variable.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
$wbsetv[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=10|
The loopcounter value is:$wbgetv[loopcounter]<br>
$wbsetv[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
Loop ended, loopcounter value is $wbgetv[loopcounter]!
</body>
</html>
```

After running this example, the resulting page in browser may look like this:

```
The loopcounter value is:0
The loopcounter value is:1
The loopcounter value is:2
The loopcounter value is:3
The loopcounter value is:4
The loopcounter value is:5
The loopcounter value is:6
The loopcounter value is:7
The loopcounter value is:8
The loopcounter value is:9
The loopcounter value is:10
Loop ended, loopcounter value is 11!
```

21.53 $WBSPLIT - convert string to array

Important: $WBSPLIT function has effect on entire WBSP page and all it's sub elements (configuration section, included files, sub reports, etc.). All instances of $WBSPLIT function that use the same array name will set same array regardless of their location inside WBSP page.

Availability
$WBSPLIT is available for use with all WBSP commands.

Syntax
$WBSPLIT{*data*|*arrayname*|*separator*|***showvar***|***sort***}
$WBSPLIT[*data*|*arrayname*|*separator*|***showvar***|***sort***]

Parameters
data - the string value (text) to be split
arrayname - the name of the array where elements will be stored
separator - a string containing one or more characters to use in separating the string
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the last index (length-1) of resulting array
sort - optional parameter - if it exists WhizBase will sort the resulting array ascending (**sort=A**) or descending (**sort=D**). Any other sort value will be ignored.

Returns
Number of elements in resulting array if showvar parameter is set to true, otherwise it returns nothing, just creates the array.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBSplit</title>
</head>
<body>
```

```
$wbsetv[filecount|0]
$wbwhile[$wbgetv[filecount]<=$wbsplit[$wbdir[]|dirlist|,|t|A]
|
$wbgetv[dirlist($wbgetv[filecount])]<br>
$wbsetv[filecount|$wbcalc[$wbgetv[filecount]+1]]
]
</body>
</html>
```

After running this example, the resulting page in browser should contain the list of all files in the directory where current WBSP file is located.

21.54 $WBSUB - execute sub-routine

Availability
$WBSUB is available for use with all WBSP commands.

Syntax
$WBSUB{*subname*}
$WBSUB[*subname*]

Parameters
subname - name of the required subroutine

Returns
Processed code of the subroutine named *subname*.

Example

```
[FormFields]
wb_command=r
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBSUB test</title>
</head>
<body>

$wbwhile[$wbgetv[r]<100|
$wbsetv[r|$wbcalc[$wbgetv[r]+10]]
Disk radius: $wbgetv[r] - Disk area: $wbsub[diskarea]<br>
]
</body>
</html>
<!--wb_beginsub_diskarea-->
$wbformat[$wbcalc[$wbgetv[r]^2*3.14159265358979323846264338327950]|#.000
]
<!--WB_EndSub-->
```

After running this example, the resulting page in browser should look like this:

```
Disk radius: 10 - Disk area: 314,159
Disk radius: 20 - Disk area: 1256,637
Disk radius: 30 - Disk area: 2827,433
Disk radius: 40 - Disk area: 5026,548
Disk radius: 50 - Disk area: 7853,982
Disk radius: 60 - Disk area: 11309,734
Disk radius: 70 - Disk area: 15393,804
Disk radius: 80 - Disk area: 20106,193
Disk radius: 90 - Disk area: 25446,900
Disk radius: 100 - Disk area: 31415,927
```

21.55 $WBUNESC - decode URL-encoded string

Availability
$WBUNESC is available for use with all WBSP commands:

Syntax
$WBUNESC{*URLEncodedString*}
$WBUNESC[*URLEncodedString*]

Parameters
URLEncodedString - any string encoded using $WBESC, $WBVC, $WBFC

Returns
Decoded value of *URLEncodedString*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head><title>WBUNESC example</title></head>
<body>
Encoded text: %E4%DF%FC%E2<br>
WBUNESC:($WBUNESC[%E4%DF%FC%E2])<br>
</body>
</html>
```

After running this example the HTML code of resulting page in browser, may look like this:

```
<html>
<head><title>WBUNESC example</title></head>
<body>
Encoded text: %E4%DF%FC%E2<br>
WBUNESC:(äßüâ)<br>
</body>
</html>
```

21.56 $WBUNTIL - loop until a condition becomes True

Important: If condition is not set properly $WBUNTIL function can get in infinite loop and stop the page processing until servers CGITimeOut is reached.

Availability
$WBUNTIL is available for use with all WBSP commands.

Syntax
$WBUNTIL{*condition*|*content*}
$WBUNTIL[*condition*|*content*]

Parameters
condition - an expression that can be True or false when evaluated
content - block of code (including WhizBase elements) that will be repeated **until** condition becomes True

Returns
Code contained in *content* repeated until *condition* becomes True.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
$wbsetv[randomnum|0]
$WBUNTIL[$wbgetv[randomnum]=30|
The random number is:$wbgetv[randomnum]<br>
$wbsetv[randomnum|$wbfn[rnd(50)]]
]
Loop ended, random number value is $wbgetv[randomnum]!
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The random number is:0
The random number is:37
The random number is:33
The random number is:22
The random number is:41
The random number is:32
The random number is:23
The random number is:37
The random number is:46
Loop ended, random number value is 30!
```

Refresh the page few times and watch the changes of the result.

21.57 $WBURL - generate navigation url

Availability
QUERY command.

Syntax
$WBURL{*URLType*}
$WBURL[*URLType*]

Parameters
URLType - two-letter code describing required navigation link URL. Valid values are:
PP – previous page
NP – next page
FP – first page
LP – last page

Returns
URL of required navigation link.

Example

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
wb_maxrec=5
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>
<a >Previous page</a> <a >Next page</a>
</center>
</body>
</html>
```

After running this example, the resulting HTML code (part that includes generated navigation links), actually received by browser, may look like this:

```
<center>
<a>Previouspage</a><a>Next page</a>
</center>
```

21.58 $WBVDHR - virtual directory home reference

Availability
$WBVDHR is available for use with all WBSP commands.

Syntax
$WBVDHR
$WBVDHR[]
$WBVDHR{}

Returns
Path to the root directory of the current virtual host as it is defined in server-side variable VirtualDirHomeRef. Used to refer to wwwroot directory from files located in virtual directory .

Here's an example:

```
<!--
[FormFields]
wb_basename=$wbvdhr{}/database/biblio.mdb
-->
```

21.59 $WBWHILE - loop while a condition is True

Important: If condition is not set properly $WBWHILE function can get in infinite loop and stop the page processing until servers CGITimeOut is reached.

Availability
$WBWHILE is available for use with all WBSP commands.

Syntax
$WBWHILE{*condition*|*content*}
$WBWHILE[*condition*|*content*]

Parameters
condition - an expression that can be True or false when evaluated
content - block of code (including WhizBase elements) that will be repeated **while** condition is True

Returns
Code contained in *content* repeated while *condition* is True.

Example

```
<!--
[FormFields]
```

```
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
$wbsetv[loopcounter|0]
$WBWHILE[$wbgetv[loopcounter]<=10|
The loopcounter value is:$wbgetv[loopcounter]<br>
$wbsetv[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
Loop ended, loopcounter value is $wbgetv[loopcounter]!
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The loopcounter value is:0
The loopcounter value is:1
The loopcounter value is:2
The loopcounter value is:3
The loopcounter value is:4
The loopcounter value is:5
The loopcounter value is:6
The loopcounter value is:7
The loopcounter value is:8
The loopcounter value is:9
The loopcounter value is:10
Loop ended, loopcounter value is 11!
```

21.60 $WBXCHNAMES - XML node child node names

Availability
$WBXCHNAMES is available for use with all WBSP commands.

Syntax
$WBXCHNAMES{XMLSource|pattern|***delimiter***}
$WBXCHNAMES[XMLSource|pattern|***delimiter***]

Parameters
XMLSource - String value containing XML to be searched. Note that this is not an URL of the XML file but XML source itself.
pattern - Any valid XPath pattern (). When using square brackets in pattern use $WBFN[CHR(91)] for left and $WBFN[CHR(93)] for right square bracket, or use escape characters.
delimiter - optional paramater - contains the character that will be used to separate names of child nodes. Default value is comma (,).

Returns
Names of the child nodes of node(s) matching XPath pattern separated by delimiter.

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
    <title>XChNames</title>
</head>
<body>
$wbxchnames[$wbgeturl[xmlsample.xml]|//codes/condition]<br>
</body>
</html>
```

File XMLSample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<codes>
  <condition>
    <code>XXX</code>
    <description>XXX description</description>
    <day_icon>XXXDayIcon.png</day_icon>
    <night_icon>XXXNightIcon.png</night_icon>
  </condition>
  <condition>
    <code>YYY</code>
    <description>YYY description</description>
    <day_icon>DayYYYIcon.png</day_icon>
    <night_icon>NightYYYIcon.png</night_icon>
  </condition>
  <condition>
    <code>ZZZ</code>
    <description>ZZZ description</description>
    <day_icon>DayIconZZZ.png</day_icon>
    <night_icon>NightIconZZZ.png</night_icon>
  </condition>
</codes>
```

After running this example, the resulting page in browser should look like this:

```
code,description,day_icon,night_icon
```

21.61 $WBXMLHTTP - get data from URL

Availability
$WBXMLHTTP is available for use with all WBSP commands.

Syntax
$WBXMLHTTP{*URL*|*PostData*|***FullPage***|***ContentType***|***SoapAction***|***Options***|***Headers***|***Method***}
$WBXMLHTTP[*URL*|*PostData*|***FullPage***|***ContentType***|***SoapAction***|***Options***|***Headers***|***Method***]

Parameters
URL - The Internet address of the resource (page, file) you want to include in your

page. It can be either absolute (containing entire address including protocol, server, path and resource) or relative (to document root of virtual server if it starts with slash / character, or to the directory where the WBSP file is located).
PostData - data to be posted in format var1=value 1&var2= value 2&...&varn= value n for ordinary POST or as XML for SOAP.
FullPage - optional parameter. If set to true (T) function will return entire page code without truncating code outside <body> and </body> tags.
ContentType - optional parameter. Sets the Content-Type clause in HTTP request header. If omitted WhizBase will send the default value **application/x-www-form-urlencoded**
SoapAction - optional parameter. Sets the SOAPAction clause in HTTP request header for web services. If omitted no default value will be sent.
Options - optional parameter. Contains comma-separated list of ServerXMLHTTP options in form optionnumber=optionvalue.
Headers - optional parameter. Contains comma-separated list of ServerXMLHTTP request headers in form headername= headervalue
Method - optional parameter. Contains HTTP method. Valid values are GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE**.** Default value is POST.

Returns
Source code of the received page/file. If source code contains <body> and </body> tags and FullPage is not set to true, function will include only the source code between these tags.

Example
File PostCookiesA.wbsp

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
<LINK href="$wbif[
"$wbgc[css]"=""
|
$wbif["$wbrv[style]"=""|style1.css|$wbrv[style]]
|
$wbif["$wbv[css]"=""|style1.css|$WBXMLHTTP[changecss.wbsp|css=$wbv[css]
]]
]" type=text/css rel=stylesheet>
<title>Cookie test</title>
</head>
<body>
Cookie value:($wbgc[css])<br>
Variable value:($wbrv[style])<br><br>
<form action="setcookieA.wbsp" method="post">
<input type="hidden" name="css" value="style1.css">
<input type="hidden" name="sp" value="$wbe[script_name]">
<input type="submit" value="Style 1">
</form>
<form action="setcookieA.wbsp" method="post">
<input type="hidden" name="css" value="style2.css">
<input type="hidden" name="sp" value="$wbe[script_name]">
<input type="submit" value="Style 2">
```

```
</form>
</body>
</html>
```

File SetCookieA.wbsp

```
[FormFields]
wb_command=R
wb_redirect=http://$wbe[server_name]$wbv[sp]?css=$wbv[css]
wb_addcookie=CSS=$wbv{CSS}
```

File ChangeCss.wbsp

```
<!--
[FormFields]
wb_filename=/default.inc
wb_command=wf
wb_keyname=Style
wb_keyvalue=$wbv{css}
wb_section=Userdata
-->
<!--WB_BeginTemplate-->$wbv[css]
```

File style1.css

```
body{
font-family:Verdana;
font-size:12px;
color:#333333;
background-color:#efefef;
}
a{
text-decoration:none;
color:#0065b7;
font-weight:bold;
}
a:hover{
color:#cc0000;
}
```

File style2.css

```
body{
font-family:Verdana;
font-size:12px;
color:#0000cc;
background-color:#ffffff;
}
a{
text-decoration:none;
color:#6500b7;
font-weight:bold;
}
a:hover{
color:#00cc00;
}
```

After running this example (file postcookiesa.wbsp) the page will change its appearance depending on value of cookie named CSS (if it exists) or depending on a value of user defined variable Style in file /default.inc, or if neither of those two exists, it will use file style1.css. When you click the either of buttons ("Style 1" and "Style 2") it will set the cookie to selected value, call the file changecss.wbsp (with selected value as parameter named CSS) using $WBXMLHTTP function, and write new value to file /default.inc as user-defined variable "Style".

21.62 $WBXPATH – Xpath

Availability
$WBXPATH is available for use with all WBSP commands.

Syntax
$WBXPATH{*XMLSource*|*pattern*|***Namespaces***}
$WBXPATH[*XMLSource*|*pattern*|***Namespaces***]

Parameters
XMLSource - String value containing XML to be searched. Note that this is not an URL of the XML file but XML source itself.
pattern - Any valid XPath pattern (). When using square brackets in pattern use $WBFN[CHR(91)] for left and $WBFN[CHR(93)] for right square bracket, or use escape characters .
Namespaces - optional parameter containing XML namespace(s) in form **nsID=nsURI.**

Returns
Value(s) of node(s) matching XPath pattern .

Example

```
[FormFields]
wb_command=R
<!--WB_BeginTemplate-->
<html>
<head>
    <title>Xpath</title>
</head>
<body>
$wbsetv[xml|$wbgeturl[xmlsample.xml]]
$wbsetv[loopcounter|1]
$WBWHILE[$wbgetv[loopcounter]<=3|
$wbxpath[$wbgetv[xml]|//codes/condition$wbfn[chr(91)]position() =
$wbgetv[loopcounter]$wbfn[chr(93)]/description]<br>
$wbsetv[loopcounter|$wbcalc[$wbgetv[loopcounter]+1]]
]
</body>
</html>
```

File XMLSample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<codes>
  <condition>
```

```
    <code>XXX</code>
    <description>XXX description</description>
    <day_icon>XXXDayIcon.png</day_icon>
    <night_icon>XXXNightIcon.png</night_icon>
  </condition>
  <condition>
    <code>YYY</code>
    <description>YYY description</description>
    <day_icon>DayYYYIcon.png</day_icon>
    <night_icon>NightYYYIcon.png</night_icon>
  </condition>
  <condition>
    <code>ZZZ</code>
    <description>ZZZ description</description>
    <day_icon>DayIconZZZ.png</day_icon>
    <night_icon>NightIconZZZ.png</night_icon>
  </condition>
</codes>
```

After running this example, the resulting page in browser should look like this:

```
XXX description
YYY description
ZZZ description
```

21.63 DB related functions

Because these functions work with recordset and input functions are processed before opening the database it is not possible to use all these functions in input syntax *func{arg}*. Functions that can be used in input syntax ($WBSR and $WBSRQ) have both syntax forms listed.

$WBDetail
$WBF
$WBRF
$WBFF
$WBFU
$WBFC
$WBP
$WBSR
$WBSRQ

21.63.1 $WBDetail - show values of all fields

Availability
$WBDETAIL is available for use with following recordset related WBSP commands:
UPDATE
MULTI UPDATE
DELETE
QUERY

Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBDETAIL[*layout*]

Parameters
layout - single letter value defining the layout of generated detail section. Valid values are **T** for tabular layout and **F** for columnar layout

Returns
Complete detail section with all database fields contained in recordset, formatted using CSS classes as follows:
For tabular layout
wbspttbl - class for table (<table class="wbspttbl"...)
wbspthdr - class for cells in first (header) row (<td class="wbspthdr"...)
wbsptrow - class for cells in data rows (<td class="wbsptrow"...)
For columnar layout
wbspftbl - class for table (<table class="wbspftbl"...)
wbsptdlbl - class for cells in left (label) column (<td class="wbsptdlbl"...)
wbsptdfld - class for cells in right (data) column (<td class="wbsptdfld"...)
wbsptdhr - class for cell at the bottom of every record containing the horizontal line (<td class="wbsptdhr"...)

Example

```
[FormFields]
WB_Command=q
WB_Basename=biblio.mdb
WB_Rcdset=titles
wb_showlogo=F
[MsgAndLbl]
WB_Style=font-family:verdana;font-size:12px;color:#CC0000;
<!--WB_BeginTemplate-->
<html>
<head>
<style>
.wbspttbl{
border:1px solid #000000;
font-family:verdana;
font-size:12px;
border-collapse:collapse;
border-spacing:0px;
}
.wbspthdr{
background-color:#CC0000;
border:1px solid #000000;
color:#C0C0C0;
}
.wbsptrow{
background-color:#FFCC00;
border:1px solid #000000;
color:#0000CC;
}
</style>
<title>Simple database example</title>
```

```
</head>
<body>
$wbdetail[t]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Title	Year Published	ISBN	PubID	AU_ID
McGraw-Hill's Encyclopedia of Networking & Telecommunications	2001	0072120053	10	10
Microsoft SMS Installer	2000	0072124474	10	9
Windows 2000 Iis 5.0 : A Beginner's Guide	2001	0072133724	9	9
Windows Nt Security Handbook	1996	0078822408	10	11
Microsoft Internet Information Server 4: the Complete Reference	1998	0078824575	10	10
Non-Designer's Scan and Print Book, The	1999	0201353946	1	2
Real World Adobe InDesign 1.5	2000	0201354780	1	1
HTML 4 for the World Wide Web: Visual Quickstart Guide	2000	0201354934	1	6
Real World Freehand 7	1997	0201688875	1	1
Netscape 3 for Macintosh Visual Quickstart Guide	1996	0201694085	1	6
Kai's Power Tools 3 for Windows Visual Quickstart Guide	1997	0201696681	1	2
InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide	2000	0201710366	1	2
Fireworks 4 for Windows and Macintosh Visual Quickstart Guide	2001	0201731339	1	2
Macromedia FreeHand 10 for Windows and Macintosh: Visual QuickStart Guide	2001	0201749653	1	2
Real World FreeHand 5.0/5.5	1996	0201883600	4	1
Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours	2000	0672318830	12	13
Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours	2000	0672320428	12	13
Photoshop 6 Photo-Retouching Secrets	2001	0735711461	3	3
www.color	2000	0823058573	8	7
Www.Layout : Effective Design and Layout for the World Wide Web	2001	0823058581	8	8

1 2
First page Next page Last page

21.63.2 $WBF - show field value

Availability
$WBF is available for use with following recordset related WBSP commands:
UPDATE
MULTI UPDATE

DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBF[*fieldname*]

Parameters
fieldname - exactly the same field name as defined in recordset.

Returns

1. The value stored in recordset field *fieldname* for current record.
2. Nothing (empty string) if recordset does not exist or there is no current record, or if field value is NULL or empty string and WB_ShowEmpty is not set to True.
3. &nbsp; if field value is NULL or empty string and WB_ShowEmpty is set to True.

Example

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Year published	Title	ISBN
2001	McGraw-Hill's Encyclopedia of Networking & Telecommunications	0072120053
2000	Microsoft SMS Installer	0072124474
2001	Windows 2000 Iis 5.0 : A Beginner's Guide	0072133724
1996	Windows Nt Security Handbook	0078822408
1998	Microsoft Internet Information Server 4: the Complete Reference	0078824575
1999	Non-Designer's Scan and Print Book, The	0201353946
2000	Real World Adobe InDesign 1.5	0201354780
2000	HTML 4 for the World Wide Web: Visual Quickstart Guide	0201354934
1997	Real World Freehand 7	0201688875
1996	Netscape 3 for Macintosh Visual Quickstart Guide	0201694085
1997	Kai's Power Tools 3 for Windows Visual Quickstart Guide	0201696681
2000	InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide	0201710366
2001	Fireworks 4 for Windows and Macintosh Visual Quickstart Guide	0201731339
2001	Macromedia FreeHand 10 for Windows and Macintosh: Visual QuickStart Guide	0201749653
1996	Real World FreeHand 5.0/5.5	0201883600
2000	Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours	0672318830
2000	Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours	0672320428
2001	Photoshop 6 Photo-Retouching Secrets	0735711461
2000	www.color	0823058573
2001	Www.Layout : Effective Design and Layout for the World Wide Web	0823058581

1 2
First page Next page Last page

21.63.3 $WBFC - show URL-encoded field value

Availability
$WBFC is available for use with following recordset related WBSP commands:
UPDATE
MULTI UPDATE
DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBFC[*fieldname*]

Parameters
fieldname - exactly the same field name as defined in recordset.

Returns

1. The url-encoded value stored in recordset field *fieldname* for current record.
2. Nothing (empty string) if recordset does not exist or there is no current record, or if field value is NULL.

Example

```
<!--
[FormFields]
WB_basename=test.mdb
wb_rcdset=demo
wb_command=Q
wb_query=ID=4
-->
<!--WB_BeginTemplate-->
<html>
<body>
<!--WB_BeginDetail-->
This is original value (using WBF): $wbf[demotext]<br>
This is a converted value (using WBFC): $WBFC[demotext]!
<!--WB_EndDetail-->
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

This is original value (using WBF):
This is some text that includes a lot of special characters like {} braces, [] brackets, ? question mark and this nice string !"#$%&/()=*
This is a converted value (using WBFC):
This%20is%20some%20text%20that%20includes%20a%20lot%20of%20special%20characters%20like%20%7B%7D%20braces%2C%20%5B%5D%20brackets%2C%20%3F%20question%20mark%20and%20this%20nice%20string%20%21%22%23%24%25%26%2F%28%29%3D*

21.63.4 $WBFF - show formated field value

Availability
$WBFF is available for use with following recordset related WBSP commands:
UPDATE
MULTI UPDATE
DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBFF[*fieldname*|*formatstring*]

Parameters
fieldname - exactly the same field name as defined in recordset.
formatstring - any valid named or user-defined format string.

Returns

1. The value stored in recordset field *fieldname* for current record formatted using formatstring
2. Nothing (empty string) if recordset does not exist or there is no current record, or if field value is NULL.

Example

```
<!--
[FormFields]
WB_basename=test.mdb
wb_rcdset=demo
wb_command=Q
wb_query=ID=2
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Demo</title>
</head>
<body>
<!--WB_BeginDetail-->
<b>Date/time</b><br>
Original content (using WBF): $wbf[demodate]<br>
Formated content (using WBFF): $wbff[demodate|dd.mm.yyyy]<br>
Formated content (using WBFF): $wbff[demodate|dddd, dd mmmm, yyyy]<br>
Formated content (using WBFF): $wbff[demodate|hh:mm:ss]<br>
Formated content (using WBFF): $wbff[demodate|hh:mm:ss AM/PM]<br>
<b>Number</b><br>
Original content (using WBF): $wbf[demonum]<br>
Formated content (using WBFF): $wbff[demonum|#,###.00]<br>
Formated content (using WBFF): $wbff[demonum|0.000]<br>
<!--WB_EndDetail-->
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Date/time
Original content (using WBF): 1.7.2008 15:35:48
Formated content (using WBFF): 01.07.2008
Formated content (using WBFF): Tuesday, 01 July, 2008
Formated content (using WBFF): 15:35:48
Formated content (using WBFF): 03:35:48 PM
Number
Original content (using WBF): 365724,65625
Formated content (using WBFF): 365.724,66
Formated content (using WBFF): 365724,656

21.63.5 $WBFU - show field value as UTF-8

Availability
$WBFU is available for use with following recordset related WBSP commands:
UPDATE
MULTI UPDATE
DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBFU[*fieldname*]

Parameters
fieldname - exactly the same field name as defined in recordset.

Returns

1. The value stored in recordset field *fieldname* for current record converted to UTF-8 charset.
2. Nothing (empty string) if recordset does not exist or there is no current record, or if field value is NULL.

Example

```
<!--
[FormFields]
WB_basename=test.mdb
wb_rcdset=demo
wb_command=Q
wb_query=ID=3
-->
<!--WB_BeginTemplate-->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<!--WB_BeginDetail-->
This is original value (using WBF): $wbf[demotext]<br>
This is a converted value (using WBFU): $wbfu[demotext]!
<!--WB_EndDetail-->
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

This is original value (using WBF): ????
This is a converted value (using WBFU): ©®™ß!

21.63.6 $WBP - recordset properties

Availability
$WBP is available for use with following recordset related WBSP commands:
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBP[*propertyname*]

Parameters
propertyname - two letter name of the required property of current recordset.
Valid values for propertyname are:
RC - record count
RN - record number
AP - absolute position
PP - percent position

Returns
The recordset property specified in *propertyname*.
RC - record count returns total number of records in a recordset
RN - record number returns actual record number for every record in a recordset (starting with 1)
AP - absolute position returns the position for every record in a recordset (starting with 0)
PP - percent position returns the percent position for every record in a recordset.
This *propertyname* value is not supported if ADO object is used!

Example

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_dbobject=D35
wb_rcdset=titles
wb_command=Q
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Title</td>
<td>Record number</td>
<td>Absolute position</td>
<td>Percent position</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[title]</td>
<td>$wbp[RN]</td>
<td>$wbp[AP]</td>
```

```
<td>$wbp[PP]</td>
</tr>
<!--WB_EndDetail-->
</table>
Total number of records:$wbp[RC]
<center>$wbnavigator</center>
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Title	Record number	Absolute position	Percent position
McGraw-Hill's Encyclopedia of Networking & Telecommunications	1	0	0
Microsoft SMS Installer	2	1	3,571429
Windows 2000 Iis 5.0 : A Beginner's Guide	3	2	7,142857
Windows Nt Security Handbook	4	3	10,71429
Microsoft Internet Information Server 4: the Complete Reference	5	4	14,28571
Non-Designer's Scan and Print Book, The	6	5	17,85714
Real World Adobe InDesign 1.5	7	6	21,42857
HTML 4 for the World Wide Web: Visual Quickstart Guide	8	7	25
Real World Freehand 7	9	8	28,57143
Netscape 3 for Macintosh Visual Quickstart Guide	10	9	32,14286
Kai's Power Tools 3 for Windows Visual Quickstart Guide	11	10	35,71429
InDesign 1.0/1.5 for Macintosh and Windows: Visual QuickStart Guide	12	11	39,28571
Fireworks 4 for Windows and Macintosh Visual Quickstart Guide	13	12	42,85714
Macromedia FreeHand 10 for Windows and Macintosh: Visual QuickStart Guide	14	13	46,42857
Real World FreeHand 5.0/5.5	15	14	50
Sams Teach Yourself Macromedia Dreamweaver 3 in 24 Hours	16	15	53,57143
Sams Teach Yourself Macromedia Dreamweaver 4 in 24 Hours	17	16	57,14286
Photoshop 6 Photo-Retouching Secrets	18	17	60,71429
www.color	19	18	64,28571
Www.Layout : Effective Design and Layout for the World Wide Web	20	19	67,85714

Total number of records:28

1 2

First page Next page Last page

21.63.7 $WBRF - show field value with processing WhizBase code

Availability
$WBRF is available for use with following recordset related WBSP commands:
UPDATE
MULTI UPDATE
DELETE
QUERY
Send Personal Mail (if recordset is used)
Send Bulk Mail.

Syntax
$WBRF[*fieldname*]

Parameters
fieldname - exactly the same field name as defined in recordset.

Returns

1. The rendered value stored in recordset field *fieldname* for current record. This means that WBSP will process any WBSP tag and/or function contained in the field value before it sends it to the client
2. Nothing (empty string) if recordset does not exist or there is no current record, or if field value is NULL.

Example

```
<!--
[FormFields]
WB_basename=test.mdb
wb_rcdset=demo
wb_command=Q
wb_query=ID=1
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Demo</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Original content (using WBF)</td>
<td>Rendered content (using WBRF)</td>
</tr>
<!--WB_BeginDetail-->
<tr>
<td>$wbf[demotext]</td>
<td>$wbrf[demotext]</td>
</tr>
<!--WB_EndDetail-->
</table>
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Original content (using WBF)	Rendered content (using WBRF)
$wbfn[RND(10)]	5

Refresh the page few times and watch the changes of the result.

21.63.8 $WBSR - sub report

Availability
$WBSR is available for use with all WBSP commands:

Syntax
$WBSR{*reportfilename*}
$WBSR[*reportfilename*]

Parameters
reportfilename - full path and file name of the sub report file.

Returns
Source code of the processed report. If source code contains <body> and </body> tags, function will include only the source code between these tags.

To learn more about sub reports, please read "WhizBase sub reports" page.

21.63.9 $WBSRQ - sub report with SQL where clause

Availability
$WBSRQ is available for use with all WBSP commands:

Syntax
$WBSRQ{*reportfilename*|*whereclause*}
$WBSRQ[*reportfilename*|*whereclause*]

Parameters
reportfilename - full path and file name of the sub report file
whereclause - the condition(s) that records must satisfy to be included in the report.

Returns
Source code of the processed report. If source code contains <body> and </body> tags, function will include only the source code between these tags.

To learn more about sub reports, please read "WhizBase sub reports" page.

21.64 INI file functions

INI files are simple text files, usually associated with Microsoft Windows. The filename extension usually used in Microsoft Windows is ".INI", but in WBSP, files using the INI file format can use any extension, such as ".CFG", ".conf", or ".TXT".

The INI file structure is very simple. It contains parameters grouped in sections and every parameter has a name and a value, delimited by an equals sign (=). The sections are defined in a line by itself, in square brackets ([and]). All parameters below the section definition are members of that section.

```
[Section]
parameter = value
```

WhizBase uses two functions for reading INI files:
$WBGV
$WBGS

21.64.1 $WBGS - get INI section

Availability
$WBGS is available for use with all WBSP commands:

Syntax
$WBGS{*filename*|*sectionname*|*separator*|*size*|*render*}
$WBGS[*filename*|*sectionname*|*separator*|*size*|*render*]

Parameters
filename - full path and file name of the configuration file
sectionname - the name of the requested section
separator - optional parameter that can be specified to separate individual variables in form var1=value1*separator*var2=value2*separator*...varN=valueN
size - optional parameter that defines the amount of memory allocated for returned values. The default value is 16384 bytes (16K). If more space is needed specify greater number
render - optional boolean parameter (T/F) that defines whether the values will be processed by WhizBase or returned as plain text

Returns
Values of all parameters parameter in specified section.

Example

File WBGS.wbsp

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBGS example</title>
</head>
<body>
$wbsetv[lng|eng]
$WBGS[resource.cfg|Messages|<br>||T]
</body>
</html>
```

File resource.cfg

```
[Messages]
HelloEng=Hello visitor from IP $wbe{remote_host}
HelloGer=Hallo Besucher von IP $wbe{remote_host}
HelloEsp=Hola visitante de IP $wbe{remote_host}
HelloIta=Ciao ospite da IP $wbe{remote_host}
HelloFra=Bonjour visiteur de IP $wbe{remote_host}
```

After running this example, the resulting page in browser, may look like this:

```
HelloEng=Hello visitor from IP 127.0.0.1
HelloGer=Hallo Besucher von IP 127.0.0.1
HelloEsp=Hola visitante de IP 127.0.0.1
HelloIta=Ciao ospite da IP 127.0.0.1
HelloFra=Bonjour visiteur de IP 127.0.0.1
```

21.64.2 $WBGV - get INI variable

Availability
$WBGV is available for use with all WBSP commands:

Syntax
$WBGV{*filename*|*sectionname*|*varname*|*default*|*render*}
$WBGV[*filename*|*sectionname*|*varname*|*default*|*render*]

Parameters
filename - full path and file name of the configuration file
sectionname - the name of the section that contains required parameter
varname - the name of the required parameter
default - optional default value that will be returned if required parameter does not have a value
render - optional boolean parameter (T/F) that defines whether the value will be processed by WhizBase or returned as plain text

Returns
Value of required parameter.

Example

File wbgv.wbsp

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBGV example</title>
</head>
<body>
```

```
$wbsetv[lng|eng]
$wbgv[resource.cfg|Messages|Hello$wbgetv[lng]||T]
</body>
</html>
```

File resource.cfg

```
[Messages]
HelloEng=Hello visitor from IP $wbe{remote_host}
HelloGer=Hallo Besucher von IP $wbe{remote_host}
HelloEsp=Hola visitante de IP $wbe{remote_host}
HelloIta=Ciao ospite da IP $wbe{remote_host}
HelloFra=Bonjour visiteur de IP $wbe{remote_host}
```

After running this example, the resulting page in browser, may look like this:

```
Hello visitor from IP 127.0.0.1
```

Then simply change the line $wbsetv[lng|eng] to $wbsetv[lng|esp] and the resulting page will change to:

```
Hola visitante de IP 127.0.0.1
```

Try changing the value to ger, ita and fra and see the result.

21.65 Request related functions

$WBV
$WBVA
$WBVC
$WBVR
$WBVS
$WBVSC

21.65.1 $WBV - request variable

Availability
$WBV is available for use with all WBSP commands:

Syntax
$WBV{*varname*|*render*}
$WBV[*varname*|*render*]

Parameters
varname - the name of requested variable sent by client as a part of request using either POST or GET method. It can be either WhizBase variable or any other variable sent by client
render - optional boolean parameter (T/F) that defines whether the value will be processed by WhizBase or returned as plain text

Returns
The value of the variable *varname*

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBV example</title>
</head>
<body>
$wbif["$wbv[thx]"<>""
|
$wbv[thx|t]
|
<form action="$wbe[script_name]" method="post">
<input type="hidden" name="thx" value="Thank you for submitting the
form, $wbv{name}!">
Please enter your name:<br><input type="text" name="name" size="20">
<input type="submit" name="Sbutt" value="submit">
</form>
]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Please enter your name:

After submitting the form the resulting page will change to something like this:

Thank you for submitting the form, Faik Djikic!

21.65.2 $WBVA - separated list of request variables

Availability
$WBVA is available for use with all WBSP commands:

Syntax
$WBVA{*typeofvars*|*separator*}
$WBVA[*typeofvars*|*separator*]

Parameters
typeofvars - single letter flag that defines which POST or GET variables will be returned. Valid values are:
A - all variables received through HTTP request (POST or GET)
W - WhizBase system variables (variable name starting with WB_) received through HTTP request (POST or GET)
O - non WhizBase variables received through HTTP request (POST or GET)
separator - optional text parameter that will be used to separate returned varaibles.

If omitted, the default value **<br>** will be used

Returns
Variables received through HTTP request (POST or GET), in form variablename=variablevalue*separator*variablename=variablevalue*separator*variablename=variablevalue

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBVA example</title>
</head>
<body>
$wbif["$WBV[thx]"<>""
|
$WBVA[A]
|
<form action="$wbe[script_name]" method="post">
<input type="hidden" name="thx" value="Thank you for submitting the
form, $WBV{name}!">
Please enter your name:<br><input type="text" name="name" size="20">
<input type="submit" name="Sbutt" value="submit">
</form>
]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Please enter your name:
```

After submitting the form the resulting page will change to something like this:

```
thx=Thank you for submitting the form, $WBV{name}!
name=Faik Djikic
Sbutt=submit
```

21.65.3 $WBVC - URL-encoded request variable

Availability
$WBVC is available for use with all WBSP commands:

Syntax
$WBVC{*varname*|*render*}
$WBVC[*varname*|*render*]

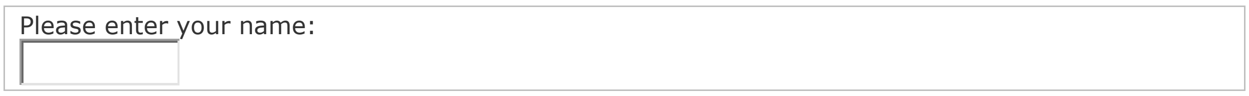
Parameters
varname - the name of requested variable sent by client as a part of request using either POST or GET method. It can be either WhizBase variable or any other variable sent by client
render - optional boolean parameter (T/F) that defines whether the value will be processed by WhizBase or returned as plain text

Returns
The url-encoded value of the variable *varname*

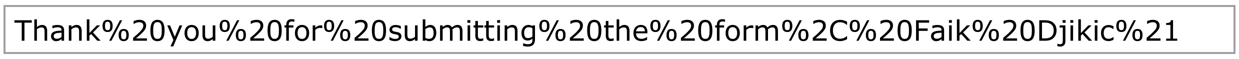
Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBVC example</title>
</head>
<body>
$wbif["$WBV[thx]"<>""
|
$WBVC[thx|t]
|
<form action="$wbe[script_name]" method="post">
<input type="hidden" name="thx" value="Thank you for submitting the
form, $WBV{name}!">
Please enter your name:<br><input type="text" name="name" size="20">
<input type="submit" name="Sbutt" value="submit">
</form>
]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:


Please enter your name:

After submitting the form the resulting page will change to something like this:


Thank%20you%20for%20submitting%20the%20form%2C%20Faik%20Djikic%21

21.65.4 $WBVR - unprocessed request variable

Important: The difference between $WBV and $WBVR functions appears when they are used in wbsp file that receives data sent by form with **ENCTYPE="multipart/form-data"**, in which case function $WBV will always return empty string if used in input syntax $wbv{varname}, and $WBVR will always return empty string when used in report syntax $WBVR[varname]. In all other cases there is no difference between these two functions.

Availability
$WBVR is available for use with all WBSP commands:

Syntax
$WBVR{*varname*}
$WBVR[*varname*]

Parameters
varname - the name of requested variable sent by client as a part of request using either POST or GET method. It can be either WhizBase variable or any other variable sent by client

Returns
The value of the variable *varname*.

Example

```
<!--
[FormFields]
wb_command=r
wb_allowmultipart=T
[Upload]
WB_Disallow=![jpg,gif,txt]
WB_Overwrite=F
WB_MaxFSize=24576
WB_UploadLog=$wbvr{name}up.log
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBVR example</title>
</head>
<body>
$wbif["$wbv[name]"<>""
|
<div style="position:relative;float:left;width:49%;border:1px solid
#000000;"><b>WBV</b><br>$wbv[name]</div>
<div style="position:relative;float:left;width:49%;border:1px solid
#000000;"><b>WBVR</b><br>$wbvr[name]</div>
|
<form action="$wbe[script_name]" method="post" ENCTYPE="multipart/form-
data">
Enter your name, please: <input type="text" name="name" size="20"><br>
Please select the file (*.jpg;*.gif;*.txt - maximum size 24 KB): <input
type="file" name="File" size="20">
<input type="submit" name="Sbutt" value="submit">
</form>
]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

Enter your name, please:
Please select the file (*.jpg;*.gif;*.txt - maximum size 24 KB):

After submitting the form the resulting page will change to something like this:

WBV Faik
WBVR

And in directory where wbsp file was located you will find file Faikup.log with upload details. If you change the line `WB_UploadLog=$wbvr{name}up.log` to `WB_UploadLog=$wbv{name}up.log` and submit the same form data again, the log file name will be up.log (without value of form variable **Name**). This will happen because $WBV returns an empty string when tries to read value of a variable submitted using multipart form.

21.65.5 $WBVS - multi-value variable separated as QUERY_STRING

Availability
$WBVS is available for use with all WBSP commands:

Syntax
$WBVS{*varname*}
$WBVS[*varname*]

Parameters
varname - the name of requested variable sent by client as a part of request using either POST or GET method. It can be either WhizBase variable or any other variable sent by client

Returns
The values of the multi-value variable *varname* separated as required for GET request

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBVS example</title>
</head>
<body>
WBVS:<br>
id=$wbvs[id]<br>
WBV:<br>
id=$wbv[id]
</body>
</html>
```

After running this example (e.g.usingGETmethod:http://localhost/wbvs.wbsp?id=3&id=5&id=7), the resulting page in browser, may look like this:

```
WBVS:
id=3&id=5&id=7
WBV:
id=3;5;7
```

21.65.6 $WBVSC - multi-value variable separated as QUERY_STRING and URL-encoded

Availability
$WBVSC is available for use with all WBSP commands:

Syntax
$WBVSC{*varname*}
$WBVSC[*varname*]

Parameters
varname - the name of requested variable sent by client as a part of request using either POST or GET method. It can be either WhizBase variable or any other variable sent by client

Returns
The url-encoded values of the multi-value variable *varname* separated as required for GET request

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBVSC example</title>
</head>
<body>
WBVSC:<br>
id=$WBVSC[id]<br>
WBV:<br>
id=$wbv[id]
</body>
</html>
```

After running this example (e.g. using GET method:
http://localhost/WBVSC.wbsp?id=WhizBase Server Pages&id=Whizbase 3.000&id=WhizBase CGI Engine), the resulting page in browser, may look like this:

```
WBVSC:
id=WhizBase%20Server%20Pages&id=Whizbase%203.000&id=WhizBase%20CGI%2
0Engine
WBV:
id=WhizBase Server Pages;Whizbase 3.000;WhizBase CGI Engine
```

21.66 Session related functions

$WBGETS
$WBSETS

21.66.1 $WBGETS - get value of session variable

Availability
$WBGETS is available for use with all WBSP commands when WB_UseSession is set to TRUE.

Syntax
$WBGETS{*varname*}
$WBGETS[*varname*]

Parameters
varname - the name of the variable

Returns
Value of requested session variable

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Cart</title>
</head>
<body>
Item $WBSETS[cartitems|$WBGETS[cartitems];$WBV[ItemID]|T] added
successfully!
</body>
</html>
```

21.66.2 $WBSETS - set value of session variable

Availability
$WBSETS is available for use with all WBSP commands when WB_UseSession is set to TRUE.

Syntax
$WBSETS{*varname*|*varvalue*|***showvar***}
$WBSETS[*varname*|*varvalue*|***showvar***]

Parameters
varname - the name of the variable
varvalue - the value that will be saved
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the saved value

Returns
Saved value if showvar parameter is set to TRUE, otherwise it returns nothing, just saves the value of a variable in session

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Cart</title>
</head>
<body>
Item $WBSETS[cartitems|$WBGETS[cartitems];$WBV[ItemID]|T] added
successfully!
</body>
</html>
```

21.67 String manipulation functions

$WBCNL
$WBCSTR
$WBFORMAT
$WBHE
$WBINDOF
$WBLEFT
$WBLEN
$WBLINDOF
$WBMID
$WBMREPL
$WBREPL
$WBRIGHT
$WBRXE
$WBRXR
$WBTRIM

21.67.1 $WBCNL - clear new line

Availability
$WBCNL is available for use with all WBSP commands:

Syntax
$WBCNL{*anystring*}
$WBCNL[*anystring*]

Parameters
anystring - any textual value that can be either WhizBase tag, function, plain text or any combination of those

Returns
value of *anystring* without new line characters (ASCII 13 and ASCII 10).

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBCNL example</title>
</head>
<body>
WBV:($wbv[test])<br>
WBCNL:($WBCNL[$wbv[test]])<br>
</body>
</html>
```

After running this example (e.g. using GET method: http://localhost/WBCNL.wbsp?test=first line%0D%0Asecond line), the HTML code of resulting page in browser, may look like this:

```
<html>
<head>
<title>WBCNL example</title>
</head>
<body>
WBV:(first line
second line)<br>
WBCNL:(first linesecond line)<br>
</body>
</html>
```

21.67.2 $WBCSTR - count string appearances

Availability
$WBCSTR is available for use with all WBSP commands.

Syntax
$WBCSTR{*str1*|*str2*|***casesensitive***}
$WBCSTR[*str1*|*str2*|***casesensitive***]

Parameters
str1 - string to be searched
str2 - string to search for
casesensitive - optional parameter - if set to true (T,ON,1) WhizBase will perform case sensitive search (so "A" will not match "a")

Returns
A number (count) of appearances (instances) of str2 inside str1.

Example

```
<html>
    <body>
    Word whizbase in whizbase@whizbase.com repeats
$WBCSTR[whizbase@whizbase.com|whizbase] times.<br>
    Word WhizBase in whizbase@whizbase.com repeats
$WBCSTR[whizbase@whizbase.com|WhizBase|T] times.
    </body>
</html>
```

Result:

```
Word whizbase in whizbase@whizbase.com repeats 2 times.
Word WhizBase in whizbase@whizbase.com repeats 0 times.
```

21.67.3 $WBFORMAT - format text

Availability
$WBFORMAT is available for use with all WBSP commands:

Syntax
$WBFORMAT{*sourcetext*|*formatstring*}
$WBFORMAT[*sourcetext*|*formatstring*]

Parameters
sourcetext - text to be formatted
formatstring - any valid named or user-defined format string.

Returns
The *sourcetext* formatted using *formatstring*

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBFORMAT example</title>
</head>
<body>
Original number: 123456789<br>
Formatted number: $WBFORMAT[123456789|#,###.00]<br>
> Original date: 12/7/08<br>
Formatted date: $WBFORMAT[12/7/08|dd-mmmm-yyyy]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Original number: 123456789
Formatted number: 123,456,789.00
Original date: 12/7/08
Formatted date: 12-July-2008
```

21.67.4 $WBHE - HTML entity

Availability
$WBHE is available for use with all WBSP commands.

Syntax
$WBHE{*sourcetext*|*validchars*}
$WBHE[*sourcetext*|*validchars*]

Parameters
sourcetext - the text containing characters that should be converted to HTML entities (&#*number*;)
validchars - the string containing characters that **will not be replaced**. If not defined, default string is used. The default string contains upper and lower case English alphabet characters, number characters, space, new line (ASCII 10) and carriage return (ASCII 13)

Returns
The *sourcetext* with HTML entities instead of any character not in *validchars* list.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBHE</title>
</head>
<body>
These characters $wbhe[ßäëöü] will be replaced by HTML entities!
</body>
</html>
```

After running this example, the source code of a resulting page should look like this:

```
<html>
<head>
<title>WBHE</title>
</head>
<body>
These characters &#223;&#228;&#235;&#246;&#252; will be replaced by HTML
entities!
</body>
</html>
```

21.67.5 $WBINDOF - index of

Availability
$WBINDOF is available for use with all WBSP commands:

Syntax
$WBINDOF{*texttobesearched*|*texttosearchfor*|***start***}
$WBINDOF[*texttobesearched*|*texttosearchfor*|***start***]

Parameters
texttobesearched - text being searched
texttosearchfor - text sought
start - optional numeric parameter specifying the position inside *texttobesearched* to begin search. If it does not exist, search begins at the beginning of the *texttobesearched*.

Returns
Number specifying position of first occurrence of *texttosearchfor* inside *texttobesearched* after *start*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBINDOF example</title>
</head>
<body>
The word Spain starts at $WBINDOF[The rain in Spain stays mainly in
plains.|spain] position in sentence The rain in Spain stays mainly in
plains.
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The word Spain starts at 13 position in sentence The rain in Spain stays mainly in
plains.
```

21.67.6 $WBLEFT - left substring

Availability
$WBLEFT is available for use with all WBSP commands:

Syntax
$WBLEFT{*sourcetext*|*length*}
$WBLEFT[*sourcetext*|*length*]

Parameters
sourcetext - text from which the leftmost characters are returned
length - numeric parameter indicating how many characters to return

Returns
Text containing a specified number (*length*) of characters from the left side of *sourcetext*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBLEFT example</title>
</head>
<body>
First 17 characters of sentence The rain in Spain stays mainly in
plains are:<br>
$WBLEFT[The rain in Spain stays mainly in plains.|17]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
First 17 characters of sentence The rain in Spain stays mainly in plains are:
The rain in Spain
```

21.67.7 $WBLEN - string length

Availability
$WBLEN is available for use with all WBSP commands:

Syntax
$WBLEN{*sourcetext*}
$WBLEN[*sourcetext*]

Parameters
sourcetext - text for which to count length

Returns
The number of characters in a *sourcetext* (length of the *sourcetext*)

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBLEN example</title>
</head>
<body>
The sentence The rain in Spain stays mainly in plains<br>
has $WBLEN[The rain in Spain stays mainly in plains] characters.
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The sentence The rain in Spain stays mainly in plains
has 40 characters
```

21.67.8 $WBLINDOF - last index of

Availability
$WBLINDOF is available for use with all WBSP commands:

Syntax
$WBLINDOF{*texttobesearched*|*texttosearchfor*|***start***}
$WBLINDOF[*texttobesearched*|*texttosearchfor*|***start***]

Parameters
texttobesearched - text being searched
texttosearchfor - text sought
start - optional numeric parameter specifying the position inside *texttobesearched* to begin search. If it does not exist, search begins at the end of the *texttobesearched*.

Returns
Number specifying position (from the beginning of the *texttobesearched*) of first occurrence of *texttosearchfor* inside *texttobesearched*, **starting from the end of *texttobesearched*** or **from *start* position, backwards**.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBLINDOF example</title>
</head>
<body>
The last occurrence of word in is at $WBLINDOF[The rain in Spain stays
mainly in plains.| in ] position in sentence The rain in Spain stays
mainly in plains.
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The last occurrence of word in is at 31 position in sentence The rain in Spain stays
mainly in plains.
```

Please note that there are two spaces around word **in** used as the second parameter of function $WBLINDOF. If you clear these spaces, function will return position 38, because last occurrence of string **in** (without spaces around it) is in word **plains**.

21.67.9 $WBMID - substring at the specified location

Availability
$WBMID is available for use with all WBSP commands:

Syntax
$WBMID{*sourcetext*|*start*|*length*}
$WBMID[*sourcetext*|*start*|*length*]

Parameters
sourcetext - text from which the rightmost characters are returned
start - position inside *sourcetext* at which the part to be taken begins
length - numeric parameter indicating how many characters to return

Returns
Text containing a specified number (*length*) of characters starting from the *start* position of *sourcetext*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBMID example</title>
</head>
<body>
The 8 character substring of sentence The rain in Spain stays mainly in
plains starting at position 10 is:<br>
$WBMID[The rain in Spain stays mainly in plains|10|8]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
The 8 character substring of sentence The rain in Spain stays mainly in plains
starting at position 10 is:
in Spain
```

21.67.10 $WBMREPL - multi replace string

Availability
$WBMREPL is available for use with all WBSP commands:

Syntax
$WBMREPL{*sourcetext*|*arrayofstringstobereplaced*|*arrayofstringstoreplacewith*|*separator*|***casesensitive***}
$WBMREPL[*sourcetext*|*arrayofstringstobereplaced*|*arrayofstringstoreplacewith*|*separator*|***casesensitive***]

Parameters
sourcetext - text on which to perform the replacement
arrayofstringstobereplaced - an array of text elements to be searched for inside *sourcetext*, separated by *separator*. **It mush have same number of elements as** *arrayofstringstoreplacewith*
arrayofstringstoreplacewith - an array of text elements to replace for every successful match of equivalent element of *arrayofstringstobereplaced* in *sourcetext*. **It mush have same number of elements as** *arrayofstringstobereplaced*
separator - optional parameter that defines the separator character used to separate elements of two array parameters. If omitted, a comma character (,) is used
casesensitive - optional parameter - if set to true (T,ON,1) WhizBase will perform case sensitive search (so "A" will not match "a").

Returns
A copy of *sourcetext* with every occurrence of every element of *arrayofstringstobereplaced* replaced by equivalent element of *arrayofstringstoreplacewith*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBMREPL example</title>
</head>
<body>
Original text: The rain in Spain stays mainly in plains.<br>
Changed text: $WBMREPL[The rain in Spain stays mainly in
plains.|Spain,in plains,rain|France,on mountains,snow]<br>
Example of changing decimal symbol and digit grouping character:<br>
Original number: 1,451,345.67<br>
Changed number: $wbmrepl[1,451,345.67|,-.-;|;-,-.|-]
```

```
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Original text: The rain in Spain stays mainly in plains.
Changed text: The snow in France stays mainly on mountains.
Example of changing decimal symbol and digit grouping character:
Original number: 1,451,345.67
Changed number: 1.451.345,67
```

Please note that in second example (changing decimal symbol and digit grouping character) we had to change the separator character to dash (-). You can also see that elements are replaced by their order in the array. So, WhizBase first replaced all commas (,) with semicolon (;). Then it replaced the full stop character (.) with comma (,), and finally replaced semicolons (;) with full stop characters (.)

21.67.11 $WBREPL - replace string

Availability
$WBREPL is available for use with all WBSP commands:

Syntax
$WBREPL{*sourcetext*|*stringtobereplaced*|*stringtoreplacewith*|***casesensitive***}
$WBREPL[*sourcetext*|*stringtobereplaced*|*stringtoreplacewith*|***casesensitive***]

Parameters
sourcetext - text on which to perform the replacement
stringtobereplaced - text to be searched for inside *sourcetext*
stringtoreplacewith - text to replace for every successful match of *stringtobereplaced* in *sourcetext*
casesensitive - optional parameter - if set to true (T,ON,1) WhizBase will perform case sensitive search (so "A" will not match "a")

Returns
A copy of *sourcetext* with every occurrence of *stringtobereplaced* replaced by *stringtoreplacewith*.

Example

```
[FormFields]
wb_command=r
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBREPL example</title>
</head>
<body>
Original text: The rain in Spain stays mainly in plains.<br>
Changed text: $WBREPL[The rain in Spain stays mainly in
plains.|Spain|France]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Original text: The rain in Spain stays mainly in plains.
Changed text: The rain in France stays mainly in plains.
```

21.67.12 $WBRIGHT - right substring

Availability
$WBRIGHT is available for use with all WBSP commands:

Syntax
$WBRIGHT{*sourcetext*|*length*}
$WBRIGHT[*sourcetext*|*length*]

Parameters
sourcetext - text from which the rightmost characters are returned
length - numeric parameter indicating how many characters to return

Returns
Text containing a specified number (*length*) of characters from the right side of *sourcetext*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBRIGHT example</title>
</head>
<body>
Last 22 characters of sentence The rain in Spain stays mainly in plains
are:<br>
$WBRIGHT[The rain in Spain stays mainly in plains|22]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Last 22 characters of sentence The rain in Spain stays mainly in plains are:
stays mainly in plains
```

21.67.13 $WBRXE - execute a regular expression

Availability
$WBRXE is available for use with all WBSP commands.

Syntax
$WBRXE{*source*|*pattern*|*arrayname*|***global***|***ignorecase***|***showvar***}
$WBRXE[*source*|*pattern*|*arrayname*|***global***|***ignorecase***|***showvar***]

Parameters
sourcetext - text on which to perform the search
pattern - regular expressions pattern
arrayname - the name of the array where elements will be stored and prefix for array where positions will be stored (arrayname_pos).
global - optional parameter - a True/False value that indicates if a pattern should match all occurrences in an entire *sourcetext* or just the first one
ignorecase - optional parameter - a True/False value that indicates if a pattern search is case-sensitive or not
showvar - optional parameter - if set to true (T,ON,1) WhizBase will show the last index (length-1) of resulting array

Returns
Number of elements in resulting array if showvar parameter is set to true, otherwise it returns nothing, just creates the array.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
String: $wbsetv[rxstr|IS1 is2 IS3 is4|t]<br>
Pattern: $wbsetv[rxp|is.|t]<br>
Number of occurrences:
$wbsetv[MCount|$WBRXE[$wbgetv[rxstr]|$wbgetv[rxp]|match|t|t|t]|t]<br>
$wbsetv[LCount|$wbcalc[$wbgetv[Mcount]-1]]
$wbwhile[$wbgetv[Lcount]>=0|
$wbgetv[match($wbgetv[LCount])] at position
$wbgetv[match_pos($wbgetv[LCount])]<br>
$wbsetv[LCount|$wbcalc[$wbgetv[LCount]-1]]
]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
String: IS1 is2 IS3 is4
Pattern: is.
Number of occurrences: 4
is4 at position 12
IS3 at position 8
is2 at position 4
IS1 at position 0
```

21.67.14 $WBRXR - regular expression replace

Availability
$WBRXR is available for use with all WBSP commands.

Syntax
$WBRXR{*source*|*pattern*|*strtoreplacewith*|***global***|***ignorecase***}
$WBRXR[*source*|*pattern*|*strtoreplacewith*|***global***|***ignorecase***]

Parameters
sourcetext - text on which to perform the replacement
pattern - regular expressions pattern
strtoreplacewith - text to replace for successful match
global - optional parameter - a True/False value that indicates if a pattern should match all occurrences in an entire *sourcetext* or just the first one
ignorecase - optional parameter - a True/False value that indicates if a pattern search is case-sensitive or not

Returns
A copy of *sourcetext* with every occurrence of *pattern* replaced by *stringtoreplacewith*.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
Source text: Quick brown fox jumps over the lazy dog.<br>
New text: $WBRXR[Quick brown fox jumps over the lazy dog.|fox|cat|t|t]
</body>
</html>
```

After running this example, the resulting page in browser, may look like this:

```
Source text: Quick brown fox jumps over the lazy dog.
New text: Quick brown cat jumps over the lazy dog.
```

21.67.15 $WBTRIM - removes both leading and trailing spaces

Availability
$WBTRIM is available for use with all WBSP commands:

Syntax
$WBTRIM{*anystring*}
$WBTRIM[*anystring*]

Parameters
anystring - any textual value that can be either WhizBase tag, function, plain text or any combination of those

Returns
value of *anystring* without both leading and trailing spaces.

Example

```
<!--
[FormFields]
wb_command=r
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBTRIM example</title>
</head>
<body>
WBV:($wbv[test])<br>
WBTRIM:($wbtrim[$wbv[test]])<br>
</body>
</html>
```

After running this example (e.g. using GET method:
http://localhost/WBTRIM.wbsp?test=%20%20%20%20%20%20%20%20some text), the HTML code of resulting page in browser, may look like this:

```
<html>
<head>
<title>WBTRIM example</title>
</head>
<body>
WBV:(        some text)<br>
WBTRIM:(some text)<br>
</body>
</html>
```

21.68 Encryption functions

$WBDECRYPT
$WBENCRYPT
$WBHASH
$WBSXOR

21.68.1 $WBDECRYPT - decrypt encrypted string

Availability
$WBDECRYPT is available for use with all WBSP commands.

Syntax
$WBDECRYPT{*AlgType*|*Key*|*data*|***InputType***}
$WBDECRYPT[*AlgType*|*Key*|*data*|***InputType***]

Parameters
AlgType - algorithm type that will be used **(must be the one used for original encryption).** Valid types are AES, AES192, AES256, BF (Blowfish), CAST, DES, RC2, RC4, RC5, 3DES (TripleDES), UC (UNIXcrypt).
data - encrypted data to be decrypted
key - the string containing encryption key **(must be exactly the same as one used for original encryption)**
InputType - optional parameter - defines the format of input data (the string to be decrypted). Valid types are H for hexadecimal (default), T for text and B for Base64 encoded string.

Returns
Decrypted value of data. When decrypting data, you must use exactly the same algorithm and key used for encryption! Different algorithms and/or different keys will produce unusable results.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBDECRYPT</title>
</head>
<body>
Encrypted text:
xHBuQv8ae1vXCVA6/3/YCd5IUjL3lbnbzHDjlWt74fGfcNuS3osQtID1BMhepJI3<br>
Encryption key: my secret word<br>
Decrypted text: $WBDECRYPT[AES|my secret
word|xHBuQv8ae1vXCVA6/3/YCd5IUjL3lbnbzHDjlWt74fGfcNuS3osQtID1BMhepJI3|B
]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
Encrypted text:
xHBuQv8ae1vXCVA6/3/YCd5IUjL3lbnbzHDjlWt74fGfcNuS3osQtID1BMhepJI3
Encryption key: my secret word
Decrypted text: this is text to be encrypted
```

21.68.2 $WBENCRYPT - encrypt a string

Availability
$WBENCRYPT is available for use with all WBSP commands.

Syntax
$WBENCRYPT{*AlgType*|*Key*|*data*|***OutputType***}
$WBENCRYPT[*AlgType*|*Key*|*data*|***OutputType***]

Parameters
AlgType - algorithm type that will be used. Valid types are AES, AES192, AES256, BF (Blowfish), CAST, DES, RC2, RC4, RC5, 3DES (TripleDES), UC (UNIXcrypt).
data - the string to be encrypted
key - the string containing encryption key
OutputType - optional parameter - defines the output format. Valid types are H for hexadecimal (default), T for text and B for Base64 encoded string.

Returns
The encrypted value of data formated as hexadecimal value, text or Base64 encoded string. When you want to decrypt data, you must use exactly the same algorithm and key! Different algorithms and/or different keys will produce completely different encryption results.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBENCRYPT</title>
</head>
<body>
Original text: this is text to be encrypted<br>
Encryption key: my secret word<br>
Encrypted text: $WBENCRYPT[AES|my secret word|this is text to be
encrypted|B]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
Original text: this is text to be encrypted
Encryption key: my secret word
Encrypted text:
xHBuQv8ae1vXCVA6/3/YCd5IUjL3lbnbzHDjlWt74fGfcNuS3osQtID1BMhepJI3
```

21.68.3 $WBHASH - calculate hash/digest

Availability
$WBHASH is available for use with all WBSP commands.

Syntax
$WBHASH{*AlgType*|*data*|***OutputType***}
$WBHASH[*AlgType*|*data*|***OutputType***]

Parameters
AlgType - algorithm type that will be used. Valid types are MD5, SHA1, SHA256, SHA384, SHA512.
data - the string to be digested

OutputType - optional parameter - defines the output format. Valid types are H for hexadecimal (default), T for text and B for Base64 encoded string.

Returns
The hash/digest value of string formated as hexadecimal value, text or Base64 encoded string.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBHASH</title>
</head>
<body>
Original text: this is text to be digested<br>
MD5 Hash: $WBHASH[MD5|this is text to be digested|B]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
Original text: this is text to be digested
MD5 Hash: GlFJb6uRzUC7MeLUBLdVzg==
```

21.68.4 $WBSXOR - simple XOR encryption/decryption

Availability
$WBSXOR is available for use with all WBSP commands.

Syntax
$WBSXOR{*key*|*data*}
$WBSXOR[*key*|*data*]

Parameters
key - the string containing encryption key
data - the string to be encrypted or decrypted

Returns
The XOR encrypted or decrypted string. XOR Encryption is the simplest stream cipher also called the Vernam cipher. The function XORs every data character with appropriate character of the key. The process is exactly same for encryption as well as decryption. With short keys Vernam or XOR Ciphers turn out to be easy to break. Still it is often used by some service providers to encrypt form data sent from web page.

Example

```
<!--
[FormFields]
wb_command=R
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>WBSXOR</title>
</head>
<body>
Encrypted text: $WBSETV[enctext|$WBSXOR[myshortkey|This is plain, clear
text]|T]<br>
Decrypted text: $WBSXOR[myshortkey|$WBGETV[enctext]]
</body>
</html>
```

After running this example, the resulting page in browser should look like this:

```
Encrypted text: 9 O K  DO M
Decrypted text: This is plain, clear text
```

22. Variables

WhizBase variables are divided in five sections - FormFields, MsgAndLbl, Upload, Referrer Check and UserData. There is also a special section for user-defined error messages - ErrorMessages section.
Variables from UserData section can be read only by using WhizBase functions $WBRV, $WBRRV, $WBGV and $WBGS .
Not every WBSP document has to have all of these sections defined, but only those that are really needed for specific task (e.g. you do not put Upload section in WBSP file that will not accept any uploaded files).
Although these sections can be located anywhere in WBSP document and they do not have to be placed together, **we strongly recommend to place these sections at the top of the file** and to separate them from rest of the report template using <!--WB_BeginTemplate--> comment, or, at least, to enclose these sections in a comment tags (use appropriate tags for every content type - <!-- --> for HTML, /* */ for JavaScript, etc.).
Commands starting with **WB_** are called WhizBase variables, which are processed at start, BEFORE opening the database and/or processing the WBSP tags and functions.
All variables can contain WhizBase tags and functions in their input syntax (syntax with braces - {}), and **some variables** can contain WhizBase tags and functions in report syntax (syntax with square brackets - []).
Functions in input syntax are processed when WB reads the variable, and those in report syntax (where applicable) are processed when WB builds the report.
Variables that can use WhizBase tags and functions in report syntax contain proper syntax example.

Important: To set the value of a single variable for entire site (e.g. WB_ShowLogo=F), put the variable definition in proper section of the file default.inc located in web sites document root directory. In the same way you can define value of a variable for entire directory (and all its subdirectories) by putting the variable definition in proper section of the file default.inc located in that directory.

22.1 FormFields variables - Subsection [FormFields]

This subsection contains the variables that are essential for processing WBSP file. Here you put information about the database, recordset, template, error template, log file, redirection, etc.
These are the variables that can be stored in FormFields subsection (in alphabetic order):

WB_AddCookie	WB_Debug	WB_Order
WB_AddJoker	WB_Defaults	WB_Pass
WB_AllowMultipart	WB_Destination	WB_PID
WB_AndOr	WB_ErrFile	WB_Predicate
WB_AppendMode	WB_ErrMail	WB_Query
WB_Attach	WB_ExactCount	WB_RcdSet
WB_AttachField	WB_Exclusive	WB_ReadOnly
WB_BaseName	WB_Execute	WB_Redirect
WB_BCC	WB_FileName	WB_Required
WB_BCCField	WB_Forced	WB_Section
WB_CC	WB_From	WB_Separator
WB_CDate	WB_FULID	WB_SetADOCompatible
WB_ChangeHFOn	WB_Group	WB_ShowEmpty
WB_Command	WB_Having	WB_ShowLogo
WB_Connect	WB_HideLogin	WB_StartRec
WB_ContentType	WB_HTTPHeader	WB_Subject
WB_DBAddData	WB_InsBR	WB_System
WB_DBAdmin	WB_KeyName	WB_SysVarByForm
WB_DBDelData	WB_KeyValue	WB_TempName
WB_DBEditData	WB_LCID	WB_TimeOut
WB_DBFlds	WB_Log	WB_To
WB_DBGroup	WB_LogData	WB_ToField
WB_DBLock	WB_LogTemp	WB_UID
WB_DBModDes	WB_MailPort	WB_Unicode
WB_DBNewPass	WB_MailServer	WB_UniFTS
WB_DBNPassCh	WB_MatchCase	WB_UserData
WB_DBObject	WB_MaxPages	WB_Usr
WB_DBOldPass	WB_MaxRec	WB_ValDelimiter
WB_DBReadData	WB_MQ	WB_WC
WB_DBReadDes	WB_Null	WB_WholeWord
WB_DBUser		

Here's an example of FormFields section (marked blue):

```
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=publishers
WB_Command=Q
wb_showlogo=F
wb_order=name
<!--WB_BeginTemplate-->
<html>
<head>
<title>Publishers</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!--WB_BeginDetail-->
<span style="font-family:Verdana;font-size:14px;font-
```

```
weight:bold;color:#0066cc;">$wbf[Name]</span><br>
$wbsr[titles.sr]<br>
<!--WB_EndDetail-->
</body>
</html>
```

22.1.1 WB_AllowMultipart - accept uploaded files

Syntax
WB_AllowMultipart=*boolean value*

Syntax example
WB_AllowMultipart= T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable controls if the current WBSP page will accept uploaded files (sent by client using multipart form). If this variable is set to TRUE WhizBase will accept and process uploaded files. If it is set to FALSE, and WBSP page receives multipart form data, the error will be generated.

22.1.2 WB_AppendMode - append report to existing file

Syntax
WB_AppendMode=*boolean value*

Syntax example
WB_AppendMode=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable controls if WBSP engine will overwrite (default - WB_AppendMode=False) the destination file defined in variable WB_Destination or new content will be **appended** to the existing file (WB_AppendMode=True). The variable is ignored by WhizBase if WB_Destination is not defined (empty).

22.1.3 WB_Command - the action to be performed by WhizBase

Syntax
WB_Command=*command*

Syntax example
WB_Command= Q

Valid inputs

R	Render
Q	Query
A	Add
D	Delete
U & MU	Update & Multi update
T	Test
SF	Send file
P	Personalized email
L	Mail to list of recipients
DF	Delete file
WF	Write to file

Database administration (available with MS Access databases only)	
AU	Add DB user or group
DU	Delete DB user or group
AG	Add DB user to group
DG	Delete DB user from group
SP	Set DB permissions
RP	Read DB permissions
CD	Compact database
CP	Change DB password

Default value
R

Description
This variable defines which WhizBase command (operation) will be executed by current WBSP page. This variable is one of the most important configuration elements in WhizBase so we will explain every value separately)please follow the links in the tables above.

22.1.3.1 Add - A

Description
This command is used for adding the new records to the database. When executed, it opens the recordset (variables WB_BaseName and WB_RcdSet must be defined when you use this command), adds the record(s) and displays processed WBSP page or redirects client to URL defined in WB_Redirect variable.

Note: If WBSP page with command A receives form data for more than one record, it will add all the records received. In this case WhizBase requires arrays of WBF_ fields with equal number of members (e.g. if you are adding 3 fields for 4 records you MUST send four sets of WBF_field1, WBF_fied2 and WBF_field3 form fields).

22.1.3.2 Add DB user or group - AU

Description
This command is used for adding specified user (WB_DBUser) or specified group (WB_DBGroup) to workgroup file (WB_System). **The result can be displayed using $WBAdmin[] tag.**

22.1.3.3 Add DB user to group - AG

Description
This command is used for adding specified user (WB_DBUser) to specified group (WB_DBGroup) in workgroup file (WB_System). **The result can be displayed using $WBAdmin[] tag.**

22.1.3.4 Change DB user password - CP

Description
This command is used for setting a new password for specified user (WB_DBUser) in workgroup file (WB_System), using old password value (WB_DBOldPass) and new password value (WB_DBNewPass) that can be verified using control password value (WB_DBNPassCh). **The result can be displayed using $WBAdmin[] tag.**

22.1.3.5 Compact database - CD

Description
This command is used for compacting MS Access database *.MDB file.
Variable WB_BaseName is required.

22.1.3.6 Delete - D

Description
This command is used for deleting the records from the database. When executed, it opens the recordset (variables WB_BaseName and WB_RcdSet must be defined when you use this command), deletes all records in the selected recordset and displays processed WBSP page or redirects client to URL defined in WB_Redirect variable.

Important: Delete command can and will permanently delete all the records that match the condition. We strongly recommend you to password-protect either your database (using MS Access system.md? file) or the WBSP page containing this command (using WB_HTAccess). To disable deletion of more than one record at the time define the WB_UID variable in WBSP page containing the DELETE command.

22.1.3.7 Delete DB user from group - DG

Description
This command is used for deleting specified user (WB_DBUser) from specified group (WB_DBGroup) in workgroup file (WB_System). **The user will not be deleted from workgroup file, but simply will not be a member of the group. The result can be displayed using $WBAdmin[] tag.**

22.1.3.8 Delete DB user or group - DU

Description
This command is used for deleting specified user (WB_DBUser) or specified group (WB_DBGroup) from workgroup file (WB_System). **The result can be displayed using $WBAdmin[] tag.**

22.1.3.9 Delete file - DF

Description
This command is used for irreversibly deleting file specified in variable WB_FileName. The file will **not** be sent to recycle bin, but deleted permanently.

22.1.3.10 Mail to list of recipients - L

Description
This command is used for sending WBSP page to email recipient(s) using SMTP mail server defined in variable WB_MailServer. Depending on variable used for defining email address the report template will be processed differently.
If email address is defined in WB_BCCField report template will be processed once and the (same) result will be sent to all recipients. The email addresses will be set as BCC so no recipient will see the email addresses of other recipients. The **mail server will be contacted once** and a single mail with many BCC addresses will be sent. Please note that some mail servers limit the maximum number of recipients for a single mail.
If email address is defined in WB_ToField report template will be processed as many times as there is records in recordset (once per record) and the result will be sent to single recipient every time, which means that mail server will be contacted **once for every recipient**.
If there are any attachments defined in WB_Attach or WB_AttachField variables, they will also be sent.
The defined recordset (using WB_BaseName, WB_RcdSet and WB_Query) will be opened same as with Query command, and all database related functions will be processed too.

22.1.3.11 Multi update - MU

Description
This command is used for updating the multiple records with same field value(s). Unlike Update command that requires arrays of WBF_ fields with equal number of members (e.g. if you update 1 field for 4 records you MUST send four sets of WBF_field1 (WB_UID member) and WBF_fied2 form fields), multi update command accept an array of WBF_field ONLY for fields defined in WB_UID. Upon receiving the WB_UID members as an array of WBF_field it will create the recordset and modify the content of all selected records with SAME value(s).

e.g.
Original table values

ID (WB_UID field)	Title	Show
1	McGraw-Hill's Encyclopedia of Networking & Telecommunications	True

2	Microsoft SMS Installer	True
3	Windows 2000 Iis 5.0 : A Beginner's Guide	True
4	Windows Nt Security Handbook	True

Table values after Update command (wbf_id=1&wbf_Show=False&wbf_id=2&wbf_Show=True&wbf_id=3&wbf_Show=False&wbf_id=4&wbf_Show=True)

ID (WB_UID field)	Title	Show
1	McGraw-Hill's Encyclopedia of Networking & Telecommunications	False
2	Microsoft SMS Installer	True
3	Windows 2000 Iis 5.0 : A Beginner's Guide	False
4	Windows Nt Security Handbook	True

Table values after Multi update command (wbf_Show=False&wbf_id=1&wbf_id=2&wbf_id=3&wbf_id=4)

ID (WB_UID field)	Title	Show
1	McGraw-Hill's Encyclopedia of Networking & Telecommunications	False
2	Microsoft SMS Installer	False
3	Windows 2000 Iis 5.0 : A Beginner's Guide	False
4	Windows Nt Security Handbook	False

22.1.3.12 Personalized email - P

Description
This command is used for sending WBSP page to email recipient(s) using SMTP mail server defined in variable WB_MailServer. The report template will be processed and the (same) result will be sent to all recipients defined in WB_To, WB_CC and WB_BCC variables together with attachments defined in WB_Attach variable (if it exists). If WBSP defines recordset (using WB_BaseName, WB_RcdSet and WB_Query) it will be opened same as with Query command, and all database related functions will be processed too.

22.1.3.13 Query - Q

Description
This command is used when you need to access the database recordset for reading the records. When executed, it opens the recordset (variables WB_BaseName and WB_RcdSet must be defined when you use this command), processes all WhizBase tags and functions that exist in the page, repeating the detail section as many times as there is records in the page (this is defined by the value of the WB_MaxRec variable). When it builds the recordset, this command will use WB_Query variable together with *WBF_recordsetfieldname* form fields to generate SQL WHERE clause. To learn more about variables used for generating recordset, please read the "Recordset" page.

Important: If your WBSP page displays only one record, and you are positive that there is more records in recordset, and that you have placed <!--WB_BeginDetail--> and <!--WB_EndDetail--> properly, then it very possible that you have forgot to set WB_Command to Q and that your WBSP page executes RENDER command.

22.1.3.14 Read DB permissions - RP

Description
This command is used for reading access permissions on database (WB_BaseName) table (WB_RcdSet) for specified user (WB_DBUser) or group (WB_DBGroup) in workgroup file (WB_System). **The result can be displayed using $WBAdmin[] tag.**

22.1.3.15 Render - R (default)

Description
This is a default command (e.g. WBSP page with no configuration section or with no WB_Command value will execute RENDER command). When executed, it processes all WhizBase tags and functions that exist in the page. If database related variables are defined, this command will open the recordset, **but it will display only first record of the recordset**.

Important: If your WBSP page displays only one record, and you are positive that there is more records in recordset, and that you have placed <!--WB_BeginDetail--> and <!--WB_EndDetail--> properly, then it is very possible that you have forgot to set WB_Command properly (to Q, P or L) and that your WBSP page executes RENDER command.

22.1.3.16 Send file - SF

Description
This command is used for sending **unchanged** file defined in variable WB_TempName to client. The file will **not** be processed or changed in any way, regardless of its content or type.

22.1.3.17 Send SMS - SMS

Description
This command is used for sending WBSP page to SMS recipient(s) using SMS modem or mobile phone connected to computer's COM port (defined in WB_SMSPort) using

cable, Infrared device or Bluetooth connection. The report template will be processed and the result will be sent to recipient defined in WB_SMSNumber . If WB_SMSField is defined, the WB_SMSNumber will be ignored, and WhizBase will use content of the defined database field as recipient's SMS number. It will process the report template for every record separately and send it to the corresponding recipient's SMS number.

22.1.3.18 Set DB permissions - SP

Description
This command is used for setting access permissions on database (WB_BaseName) table (WB_RcdSet) for specified user (WB_DBUser) or group (WB_DBGroup) in workgroup file (WB_System). The permissions are set using True/False flags in variables WB_DBAddData, WB_DBAdmin, WB_DBDelData, WB_DBEditData, WB_DBModDes, WB_DBReadData, WB_DBReadDes. **The result can be displayed using $WBAdmin[] tag.**

22.1.3.19 Test - T

Description
This command is used for generating test report. To learn more, please read the "Test mode" page.

22.1.3.20 Update - U

Description
This command is used for updating the record(s). When executed, it opens the recordset (variables WB_UID, WB_BaseName and WB_RcdSet must be defined when you use this command), changes the record(s) and displays processed WBSP page or redirects client to URL defined in WB_Redirect variable.

Important: If WBSP page with command U receives form data for more than one record, it will update all the records received. In this case WhizBase requires arrays of WBF_ fields with equal number of members (e.g. if you update 1 field for 4 records you MUST send four sets of WBF_field1 (WB_UID member) and WBF_fied2 form fields). For more info on updating more records at once please also read Multi update command. We strongly recommend you to password-protect either your database (using MS Access system.md? file) or the WBSP page containing this command (using WB_HTAccess). WB_UID variable in WBSP page containing the UPDATE command defines the name of recordset field that has a unique value for every single record. If recordset for single WBF_UIDfieldname form field value returns more than one record the error will be generated.

22.1.3.21 Write to file - WF

Description
This command is used for adding, changing and deleting key values in configuration file specified in variable WB_FileName, belonging to section specified in WB_Section variable.

22.1.4 WB_Config

This variable does not exist in WBSP engine (WhizBase version 4 and above), because the WBSP page itself is the configuration file.

22.1.5 WB_Defaults - set the default values for request variables

Syntax
WB_Defaults=*field pairs*

Syntax example
WB_Defaults=wbf_expires=$wbfn{date},wbf_user=$wbfn{htuser}

Valid inputs
Any list of pairs in formatfieldname=value,fieldname1=value1,...

Default value
None

Description
This variable contains field pairs (in format fieldname=fieldvalue) separated by character(s) defined in variable WB_ValDelimiter, that will be used to provide the value(s) for defined field(s) if user does not enter any data for that fields in HTML form. **This variable cannot be set by HTML form.**

22.1.6 WB_Destination - set the file name for saving the output

Syntax
WB_Destination=*file name of the destination file*

Syntax example
WB_Destination=current.txt

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
This variable contains the file name of the file to be used as destination file for saving the output of the WBSP file instead of sending the result to the HTTP client. This variable can be used with all WB_Command values and in all cases it **requires** that WB_Redirect variable is also defined. If this variable contains proper file name, WhizBase will process WBSP file and save (or append depending upon value of WB_AppendMode variable) the resulting code (everything below <!--WB_BeginTemplate-->) to file defined in WB_Destination. After this HTTP client will be redirected to the URL defined in WB_Redirect variable.

22.1.7 WB_Forced - force values for request variables

Syntax
WB_Forced=*field pairs*

Syntax example
WB_Forced=wbf_expires=$wbfn{date},wbf_user=$wbfn{htuser}

Valid inputs
Any list of pairs in formatfieldname=value,fieldname1=value1,...

Default value
None

Description
This variable contains field pairs (in format fieldname=fieldvalue) separated by character(s) defined in variable WB_ValDelimiter, that will be used to replace the value(s) for defined field(s) **even if user provides the data for that fields** using any valid or invalid (even illegal) method. **This variable cannot be set by HTML form.**

22.1.8 WB_FULID - generate unique form upload ID

Syntax
WB_FULID=*boolean value*

Syntax example
WB_FULID= T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
If this variable is set to TRUE, the WhizBase will generate server-side **unique form upload ID**, that can be retrieved on **the same WBSP page** using $WBFULID tag. This variable together with tag $WBFULID and functions $WBFUP and $WBFUT is used for tracking the form upload progress using AJAX or IFRAME. Generated value will be deleted when upload finishes. This variable should be set **in a WBSP page containing upload form**.

22.1.9 WB_HideLogin - scramble login data in navigation URL

Syntax
WB_HideLogin=*boolean value*

Syntax example
WB_HideLogin= F

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
TRUE

Description
This variable controls if WBSP engine will encode values for database access login variables WB_Usr and WB_Pass in navigation links (Next page, Previous page, etc.). If this variable is set to FALSE, WBSP will pass values for WB_Usr and WB_Pass without any changes so they will be visible in browser's address bar. **This variable cannot be set by HTML form.**

22.1.10 WB_Required - list of required request variables

Syntax
WB_Required=*comma separated list of required form field names*

Syntax example
WB_Required= WBF_name,WBF_phone,WBF_email

Valid inputs
comma separated names of required form fields that exist in form that sends the data to the current WBSP page

Default value
none (no form fields are required)

Description
This variable controls if the current WBSP page has received all required form fields with GET or POST request generated by HTML form that called the WBSP page. If any of the form fields defined in WB_Required list is empty (does not contain any data), **or does not exist**, the error will be generated, **even if that variable exists in WBSP page itself**. The list can contain name of any existing form filed (not only WB_ and WBF_ fields) that is required by current WBSP page for any reason.

22.1.11 WB_ShowLogo - display powered by WhizBase logo

Syntax
WB_ShowLogo=*boolean value*

Syntax example
WB_ShowLogo= F

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
TRUE

Description
This variable controls if the current WBSP page will display the "Powered by WhizBase" logo at the bottom of every WBSP page. To hide logo set this variable to FALSE.

22.1.12 WB_SysVarByForm - allow system variables (wb_) set as request variables (by form)

Syntax
WB_SysVarByForm=*boolean value*

Syntax example
WB_SysVarByForm= F

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
TRUE

Description
This variable controls if the current WBSP page will accept WhizBase variables (with name starting with WB_) sent by client using HTML form. If this variable is set to FALSE WhizBase will **not** accept any form field with name starting with WB_. If you want to disable WB_ form fileds for most of the system variables, but not all, add variables for which you want to enable form fields to proper section of default.inc file in document root directory and set their value to **$form$**.

22.1.13 WB_TempName - report template file name

Syntax
WB_TempName=*file name of the report template*

Syntax example
WB_TempName= bibliotable.htm

Valid inputs
Any valid file name respecting WhizBase path rules and special values $default$, $self$ and $form$

Default value
none (WBSP page itself, all code below <!--WB_BeginTemplate--> comment will be used as report template)

Description
This variable contains the file name of the file to be used by WhizBase as template for generating output (report). This variable is mostly used with SF command because in most other cases WBSP file (page) itself contains the template located below <!--WB_BeginTemplate--> comment. However, it is very useful when Content-type of the output is not text/html or text/plain, or when single configuration section is used for various outputs. If special values are used their functions are:

$default$	WhizBase will generate the default report page with all fields from recordset included in it, just as if you used $wbdetail function. For recordsets having more than 5 fields it will return columnar report, and for others the returned report will be in tabular format.
$form$	Use this value if you, for any reason, want to allow your visitor to choose report template file (using POST or GET method and form fields) when accepting system variables from HTML form is disabled (WB_SysVarsByForm=False).
$self$	Use this value to force WBSP page to use itself as report template, and to avoid accepting the value for this variable from HTML form field, even when accepting system variables from HTML form is not disabled (WB_SysVarsByForm=True).

22.1.14 WB_TimeOut - set script time-out interval

Syntax
WB_TimeOut=*number of seconds*

Syntax example
WB_TimeOut= 60

Valid inputs
Any number in range 0-86400 (0 - no time out - not recomended)

Default value
Value of ScriptTimeOutSec server configuration variable

Description
This variable defines the maximum number of seconds that WBSP engine will wait for WBSP file to execute before it terminates the execution and generates the error. Counting starts **after** WhizBase receives all form data from client (to prevent timeout error due to slow client connection and file upload).

22.1.15 WB_UseEscapes - use escape sequences for special WB characters

Syntax
WB_UseEscapes=*boolean value*

Syntax example
WB_UseEscapes= T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable controls if the current WBSP page is using escape characters. If this

variable is set to TRUE WhizBase will search the page for escape characters and replace them with proper content.

22.1.16 WB_UserData - user defined content

Important: WB_UserData will not be processed if placed in configuration section of **sub report** file.

Syntax
WB_UserData=*any user defined value*

Syntax example
WB_UserData=$wbtimer

Valid inputs
Any text and/or numeric value including WhizBase tags and functions

Default value
none

Description
This variable has no effect on the execution of WBSP page by itself. However since it will be processed as any other WBSP system variable at the beginning of the session and its value can be read using $wbfn[userdata] it can be very useful. For example it can be used to get the system time on the beginning of the session (using $wbtimer tag) and to be compared to the system time on the end of the WBSP report to measure time needed for report to complete.

22.1.17 WB_ValDelimiter - delimiter for wb_defaults and wb_forced

Syntax
WB_ValDelimiter=*delimiter string*

Syntax example
WB_ValDelimiter= ;

Valid inputs
Any valid string

Default value
, (comma)

Description
This variable controls which character(s) will be used to separate field pairs in WB_Defaults and WB_Forced variables. **This variable can not be set by HTML form.**

22.1.18 Access control

WhizBase controls page access using both IP filtering (WB_IPListFile) and user's credentials (WB_HTAccess, WB_HTUsr, WB_HTPass).

22.1.18.1 HTAccess File - configuration file for authentication

HTAccess file is ordinary ASCII text file that contains WhizBase variables WB_AuthType, WB_Realm, WB_Scramble and WB_LoginPage and username/password pairs, organized by following structure:

```
[Authentication]
WB_AuthType=type (valid settings B-Basic, C-Cookie)
WB_LoginFile=filename (ignored if WB_AuthType=B)
WB_Realm=Realm

[AuthUsers]
username=password
username1=password1
...
usernameN=passwordN
```

For security reasons we strongly recommend to use wbsp extension for HTAccess file and to add WB_Command=R and <!--WB_BeginTemplate--> in it.
e.g.

```
[FormFields]
WB_Command=R
[Authentication]
WB_HTAccess=C
WB_LoginFile=/login.htm
[AuthUsers]
username=password
username1=password1
<!--WB_BeginTemplate-->
<html><body>
You can not read content of this file!
</body></html>
```

You can also use **aut** extension and set the server to handle files with that extension by WBSP and do not change HiddeDocuments SSC variable, or you can place HTAccess file above wwwroot directory and use absolute path. For this you have to change AbsolutePath SSC variable.

22.1.18.1.1 WB_AuthType - authentication method

Syntax
WB_AuthType=*any valid WBSP constant*

Syntax example
WB_AuthType=C

Valid inputs
This variable accepts following WBSP constants:

- C - cookie authentication
- B - basic authentication
- D - Digest authentication

Default value
B

Description
This variable defines authentication method that will be used. Its value can be set **only** in HTAccess file in section [Authentication] and valid values are B for basic authentication, D for digest authentication and C for server-side cookie authentication. Basic authentication is default, but some servers do not support it or they require additional configuration. Cookie authentication method is supported by all web servers.

This variable cannot be set by HTML form.

22.1.18.1.2 WB_LoginPage - file name of login page for cookie authentication

Syntax
WB_LoginPage=*file name of the WBSP file*

Syntax example
WB_LoginPage=/login/login.WBSP

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
defines file that will be used for login with cookie authentication method. If this value is not specified WBSP will generate default login page. If specified file does not exist WBSP will generate an error.
Login page is ordinary HTML file containing form where user can enter two variables WB_HTUsr and WB_HTPass.
Form action should be set to $wbe[script_name], so user will be transferred to requested page once the login data are processed.

Here is an example:

```
<form method='post' action='$wbe[script_name]'>
User name:<input type='text' name='WB_HTUsr'>
Password:<input type='password' name='wb_htpass'>
<input type='submit' name='sButt' value='Login'>
</form>
```

This variable cannot be set by HTML form.

22.1.18.1.3 WB_Realm - realm for basic and digest authentication

Syntax
WB_Realm=*realm*

Syntax example
WB_Realm=Members area

Valid inputs
Any alphanumeric string

Default value
WBSP Login

Description
This variable contains realm value for WBSP basic and digest authentication. Its value can be set only in htaccess file in section [Authentication].

This variable cannot be set by HTML form.

22.1.18.1.4 WB_Scramble - scramble (hide) password(s) stored in htaccess file

Syntax
WB_Scramble=*boolean value*

Syntax example
WB_Scramble=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable controls if the WhizBase will scramble the passwords in AuthUsers section of the htaccess file upon user's first login. To make passwords unreadable set this variable to TRUE. Its value can be set only in htaccess file in section [Authentication].

This variable cannot be set by HTML form. The scrambled passwords will remain valid even if you set this variable to FALSE later, but no new (or changed) password will be scrambled until you set it back to TRUE.

22.1.18.2 WB_HTAccess - location of configuration file for authentication

Syntax
WB_HTAccess=*file name of the HTAccess file*

Syntax example
WB_HTAccess=/login/users.aut

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
This variable is used to engage Authentication procedure using usernames/passwords contained in specified file. Authentication can be either WWW-Authentication Basic or based on server-side cookies. This variable must be set in all WBSP files that need to be protected, either in a file itself or in include file. To protect entire directory set WB_HTAccess value in default.inc file in that directory. To protect entire site set WB_HTAccess in default.inc located in wwwroot directory. Some web servers will not support WBSP Basic authentication because they process WWW_Authenticate field sent by browser before it reaches WBSP engine. In that cases cookie authentication type should be used.

This variable cannot be set by HTML form.

22.1.18.3 WB_HTPass - authentication password

Syntax
WB_HTPass=*user password*

Syntax example
```
<input type="password" name="WB_HTPass" size="20">
```

Valid inputs
Any alphanumeric string

Default value
none

Description
This variable contains password for WBSP authentication using HTAccess file. **Its value is case sensitive**.

This value cannot be set in any WBSP file. WBSP will accept it only from HTML form.

22.1.18.4 WB_HTUsr - authentication user name

Syntax
WB_HTUsr=*user name*

Syntax example
```
<input type="text" name="WB_HTUsr" size="20">
```

Valid inputs
Any alphanumeric string

Default value
none

Description
This variable contains user name for WBSP authentication using HTAccess file. **Its value is not case sensitive**.

This value cannot be set in any WBSP file. WBSP will accept it only from HTML form.

22.1.18.5 WB_IPListFile - location of file containing list of (dis)allowed IP address ranges

Syntax
WB_IPListFile=*file name of the IPList file*

Syntax example
WB_IPListFile=/AcceptIP.aut

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
This variable is used to engage Authentication procedure using IP address of remote computer and list of IP ranges contained in specified file. IPList file is ordinary ASCII text file with single IP range per line.

Here is an example:

```
192.168.*.*
216.191.90.144
10.0.0.0-10.0.127.255
192.168.0.0/16
```

As you can see from example above, IPList file contains list of valid IP address ranges what means that WBSP will serve WBSP files only to users with IP address in specified range. If you want **to block access** for IP addresses in IP ranges specified in IPList file, **then add exclamation mark preceding file name in WB_IPListFile** like in following example:

```
WB_IPListFile=IPrange.aut (this will allow access only to users with IP
address in range(s) specified in file IPrange.aut)
WB_IPListFile= !IPrange.aut (this will allow access to all users except
those with IP address in range(s) specified in file IPrange.aut)
```
Please note that in both cases actual file name is the same IPrange.aut
This variable must be set in all WBSP files that need to be protected. It can be set either in a file itself or in include file. To protect entire directory set WB_IPListFile value in default.inc file in that directory. To protect entire site set WB_IPListFile in default.inc located in wwwroot directory.

This variable cannot be set by HTML form.

22.1.19 Database

These variables are used for configuring database-related parameters.

WB_AddJoker	WB_InsBr	WB_ReadOnly
WB_AndOr	WB_LCID	WB_SetADOCompatible
WB_BaseName	WB_MatchCase	WB_ShowEmpty
WB_CDate	WB_MaxPages	WB_StartRec
WB_ChangeHFOn	WB_MaxRec	WB_System
WB_Connect	WB_MQ	WB_UID
WB_DBFlds	WB_Null	WB_Unicode
WB_DBLock	WB_Order	WB_UniFTS
WB_DBObject	WB_Pass	WB_UniQS
WB_ExactCount	WB_Predicate	WB_Usr
WB_Exclusive	WB_Query	WB_WC
WB_Execute	WB_RcdSet	WB_WholeWord
WB_Group		
WB_Having		

Recordset field value container: WBF_field

22.1.19.1 WB_AddJoker - position of automatically added wildcards

Syntax
WB_AddJoker=*any valid WBSP constant*

Syntax example
WB_AddJoker=N

Valid inputs
This variable accepts following WBSP constants:

- S - add wildcard at the beginning of the word
- E - add wildcard at the end of the word
- N - wildcard will not be added
- B - add wildcard to both ends of the word

Default value
B

Description
This variable defines how WhizBase engine will perform the search for values provided by WBF_ form fields. WhizBase engine uses SQL to retrieve data from the database, and SQL uses joker (wildcard) characters like * (% in ANSI SQL) or ? (_ in ANSI SQL) for "LIKE" comparison (WhizBase can do LIKE comparison for fields of TEXT and MEMO data type). By default WhizBase adds wildcards on both sides of data provided using WBF_ form fields. By using WB_AddJoker variable, this behavior can be changed.

e.g.

WBF_Name value	WB_AddJoker value	WHERE clause
John	B	Name LIKE "*John*"
John	E	Name LIKE "John*"
John	S	Name LIKE "*John"
John	N	Name LIKE "John"

22.1.19.2 WB_AndOr - condition concatenation type

Syntax
WB_AndOr=*any valid WBSP constant*

Syntax example
WB_AndOr=AND

Valid inputs
AND - for returning the records that meet **all** conditions
OR - for returning the records that meet **any** of the conditions

Default value
OR

Description
This variable defines how WhizBase engine will perform the search for values provided by WBF_ form fields. When WhizBase generates a query for retrieving the records based upon WBF_ values for more than one field (receives wbf_ form fields for more than one database field) it can be set to combine the conditions (AND) or to return records that meet any single condition (OR).

e.g.

WBF_Title	WBF_Year published	WB_AndOr	WHERE clause	Number of records
www	2000	OR	Title LIKE "*www*" OR Year published=2000	11
www	2000	AND	Title LIKE "*www*" AND Year published=2000	2

22.1.19.3 WB_BaseName - name of the database used

Syntax
WB_BaseName=*name of the MDB file or directory where ISAM databases are located*

Syntax example
WB_BaseName=biblio.mdb

Valid inputs
Any valid file or directory name (depending on what database type is used)

Default value
none

Description
This variable contains the file name of the file to be used (for MS Access databases and MS Excel files) or in case of using ISAM database the name of the directory containing the data.
In some cases (depending upon WB_Command, WB_Connect and WB_DBObject) this variable is required.

e.g.

WB_Connect	WB_BaseName	WB_RcdSet
	/database/biblio.mdb	titles
dBASE III;	/database/	titles.dbf
Text;	/database/	titles.txt
Excel 8.0;	/database/biblio.xls	titles

22.1.19.4 WB_CDate - set conversion of date/time and boolean fields

Syntax
WB_CDate=*boolean value*

Syntax example
WB_CDate=F

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
TRUE

Description
This is a True/False flag that determines how WhizBase will treat boolean and date/time fields value (received by WBF_ form field). If set to true (default), WhizBase will convert received value representing date/time to numeric date value, and true/false string to actual boolean value. Actually this variable should be changed **only** in case of connecting to database type that uses ANSI SQL (e.g. MySQL, Oracle, etc.). If this variable is set to False WhizBase will compare the exact value for date and boolean field as it was received from HTML form.

22.1.19.5 WB_ChangeHFOn - report header/footer grouping field(s)

Syntax
WB_ChangeHFOn=*database field name*

Syntax example
WB_ChangeHFOn=[Year published]

Valid inputs
Any database field name or list of names

Default value
none

Description
This variable defines which field(s) will be monitored for value change that will initiate showing header and footer report sections. If there is more than one field, change in any of those fields will initiate showing header and/or footer section. Field names that have a space in name must be enclosed in square brackets.
If WB_Order variable does not exist it will get exactly the same value as WB_ChangeHFOn. If it exists, then it must start with same fields in same order as they appear in WB_ChangeHFOn, or WBSP will generate the error.

e.g.

WB_ChangeHFOn	valid WB_Order	invalid WB_Order
[Year published]	[Year published]	Title
[Year published]	[Year published], Author	Author, [Year published]
[Year published], Author		[Year published], Publisher

22.1.19.6 WB_Connect - ISAM driver or ODBC DSN

Syntax
WB_Connect=*any valid connection string*

Syntax example
WB_Connect=dBASE III;

Valid inputs

Database type	WB_Connect value (do not forget semicolons)
Microsoft Jet Database	none
dBASE III	dBASE III;
dBASE IV	dBASE IV;
dBASE 5	dBASE 5.0;
Paradox 3.x	Paradox 3.x;
Paradox 4.x	Paradox 4.x;
Paradox 5.x	Paradox 5.x;
FoxPro 2.0	FoxPro 2.0;
FoxPro 2.5	FoxPro 2.5;
FoxPro 2.6	FoxPro 2.6;
Excel 3.0	Excel 3.0;
Excel 4.0	Excel 4.0;
Excel 5.0 or Excel 95	Excel 5.0;
Excel 97-2003	Excel 8.0;
Excel 2007+	Excel 12.0 Xml;HDR=YES;
Text	Text;
ODBC	ODBC;DATABASE=database;UID= user;PWD=password;DSN=datasourcename;[LOGIN TIMEOUT= seconds;]

Default value
Microsoft Jet Database - MS Access (empty value of WB_Connect)

Description
This variable contains a database type specifier defining which ISAM driver or ODBC DSN will be used for accessing the database. Do not set any value to this variable if you are using MS Access database (*.mdb or *.accdb).

22.1.19.7 WB_DBFlds - field(s) included in recordset

Syntax
WB_DBFlds=*database field list*

Syntax example
WB_DBFlds=Title, Year published, ISBN, Qty, Price, Qty*Price as Amount

Valid inputs
Any database field name, list of names or valid SQL expression

Default value
* (all fields from recodrset)

Description
This variable contains SQL expression that defines what database fields will be included in the recordset.

22.1.19.8 WB_DBLock - record locking type

Syntax
WB_DBLock=*any valid WBSP constant for record locking*

Syntax example
WB_DBLock=P

Valid inputs
This variable accepts following WBSP constants:

- A - Automatic
- P - Pessimistic locking
- O - Optimistic locking
- U - Unspecified (for DAO connections same as A)

e.g.
```
WB_DBLock=P
or
WB_DBLock=O
```

Default value
A

Description
This variable defines what type of record locking will be used when opening the

recordset. If you are not sure what type of record locking you need do not change the default value.

22.1.19.9 WB_DBObject - object used to access the database

Syntax
WB_DBObject=*any valid WBSP constant or ADO connection string*

Syntax example
WB_DBObject=D35

Valid inputs
This variable accepts following WBSP constants:

- D35 - use DAO object version 3.5
- D36 - use DAO object version 3.6
- A35 – use ADO object with Microsoft.Jet.OLEDB.3.5 provider
- A40 – use ADO object with Microsoft.Jet.OLEDB.4.0 provider
- A07 - use ADO object with MS Access 2007 databases - *.accdb files (requires 2007 Office System Driver: Data Connectivity Components installed on the server)
- A10 - use ADO object with MS Access 2010 databases - *.accdb files (requires Microsoft Access Database Engine 2010 Redistributable installed on the server)

If ADOConnectionString server configuration variable is set On then this variable can accept any valid ADO connection string.

e.g.
```
WB_DBObject=Driver={OracleODBCDriver};Dbq=YourDataBaseName;Uid=YourUser
Name;Pwd=YourPassword
or
WB_DBObject=Provider=sqloledb;DataSource=(local);Initial Catalog=
YourDataBaseName;UserID=YourUserName;Password=YourPassword
```

Default value
A40

Description
This variable defines what type of object will be used to access the database. If ADO connection string is provided, WBSP will use ADO object.

22.1.19.10 WB_ExactCount - count all records in recordset

Syntax
WB_ExactCount=*boolean value*

Syntax example
WB_ExactCount=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WBSP engine will perform full page count. If the result of the query is a recordset with huge number of records, WBSP engine will count and display links for first N pages, and then as user moves forward it will provide links for more pages. This way WBSP engine works much faster. If you for some reason want WBSP engine to show exact number of pages for a recordset with a huge number of records, set this variable to TRUE. Default value is FALSE. This field has no effects if used with cross-table query (in that case WBSP engine will process all records regardless of WB_ExactCount value) and it also does not affect $WBP[RC] function.

22.1.19.11 WB_Exclusive - open the database in exclusive mode

Syntax
WB_Exclusive=*boolean value*

Syntax example
WB_Exclusive=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WhizBase engine will open the database in exclusive mode (single user mode). This is required when WBSP engine is used on read-only media (such as web on CD-ROM). Placing WBSP engine and wbsp pages on read-only media also requires setting field WB_ ReadOnly to TRUE.

22.1.19.12 WB_Execute - execute SQL statement(s)

Syntax
WB_Execute=*SQL action query*

Syntax example
WB_Execute=update titles set [Year Published]=2000 where ([Year published]=2010)

Valid inputs
Any valid SQL statement supported by current database. To use more than one SQL statement separate them with semicolon followed by the new line characters (e.g. ;$wbfn{chr(13)}$wbfn{chr(10)}) or place them in include file one statement par line ending with semicolon and call it using $wbrinc function.

Default value
none

Description
This variable contains SQL statement that will be executed **before** main action defined in WB_Command variable. When it is set in subreport WBSP file, this variable **can contain** the $WBF function in report format. In this case $WBF function will refer to superior recordset (the one defined in file containing the sub report).

22.1.19.13 WB_Group - SQL clause "GROUP BY"

Syntax
WB_Group=*database field name*

Syntax example
WB_Group=[Year published]

Valid inputs
Any database field name or list of names

Default value
none

Description
This variable contains the field name (or comma-separated list of field names) upon which WBSP engine will group the records (SQL clause GROUP BY) when calculating aggregate SQL functions. Field names that have a space or other special character in name must be enclosed in square brackets.

22.1.19.14 WB_Having - SQL clause "HAVING"

Syntax
WB_Having=*condition*

Syntax example
WB_Having=(((Sum(IncPayment.AmountPaid))<>[AmountDue]))

Valid inputs
Any valid SQL condition

Default value
none

Description
This variable contains the SQL Clause HAVING that specifies which grouped records are displayed in a SELECT statement with a GROUP BY clause (WB_Having). After GROUP BY combines records, HAVING displays any records that satisfy the conditions of the HAVING clause.

22.1.19.15 WB_InsBr - replace new line characters with <br>

Syntax
WB_InsBr=*boolean value*

Syntax example
WB_InsBr=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WhizBase engine will send break tag (**<BR>**) instead of line break character (ASCII 10) contained in any database field.

22.1.19.16 WB_LCID - locale identifier ID

Syntax
WB_LCID=*numeric value*

Syntax example
WB_LCID=3079

Valid inputs
Any numeric value of Locale ID (LCID)

Default value
none

Description
This variable defines Locale identifier different from the the one defined by system. Complete list of locales can be found here.

22.1.19.17 WB_MatchCase - case sensitive search

Syntax
WB_MatchCase=*boolean value*

Syntax example
WB_MatchCase=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable defines if the WhizBase engine will perform case sensitive search on ISAM databases. It has no effect on MS Access databases. However, search will be faster with this field set to TRUE with any database type.

22.1.19.18 WB_MaxPages - maximum number of page links in report navigation

Syntax
WB_MaxPages=*numeric value*

Syntax example
WB_MaxPages=30

Valid inputs
Any numeric value

Default value
20

Description
This variable contains the maximum number of direct links to the other pages of the report. If it is not set, WhizBase will generate and display links to 20 pages (if $WBNavigator or $WBPageNums tag exists). WhizBase will generate links to the other pages by placing current page number in the middle of the list if possible.

22.1.19.19 WB_MaxRec - maximum number of records per page

Syntax
WB_MaxRec=*numeric value*

Syntax example
WB_MaxRec=30

Valid inputs
Any numeric value and WBSP constant all

Default value
20 for main report
all for subreport

Description
In order to prevent overloading of user's browser WBSP engine breaks the report in pages. Use WB_MaxRec variable to define the number of the records to be displayed on each report page. Unlike previous versions (where maximum value for this field was limited to 100 records) since version 3.000 there is no limit to this value. There is, however, one special value - **all** (WB_MaxRec=all). The value all defines that WBSP engine will generate report that contains ALL records from selected recordset.

22.1.19.20 WB_MQ - make query (yes/no)

Syntax
WB_MQ=*boolean value*

Syntax example
WB_MQ=F

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
TRUE

Description
It is possible to prevent WhizBase from creating the WHERE clause based on values of WBF_ fields by setting this variable to FALSE. This can speed up WBSP pages that use WB_Query variable or reports that show all the records from recordset. In all other cases it is wise not to change the value of WB_MQ.

22.1.19.21 WB_Null - update string for clearing field value

Syntax
WB_Null=*any string value*

Syntax example
WB_Null=$CLR

Valid inputs
Any string value

Default value
$WBNULL$

Description
This variable sets the value that will be used to identify field that should be set to NULL value during the Update command. WBSP uses string defined in WB_Null variable as a value for WBF_ form field(s) to define which database table fields in current (updated) record should be set to empty string. It does not apply to the numeric fields (to clear them set their value to 0).

22.1.19.22 WB_Order - SQL clause "ORDER BY"

Syntax
WB_Order=*database field name*

Syntax example
WB_Order=[Year published] DESC, Publishers.Name

Valid inputs
Any database field name or list of names

Default value
none

Description
This variable contains the the name of database field (or comma separated list of fields) by which the records will be sorted. Field names that have a space in name must be enclosed in square brackets. To sort the records is descending order add word **DESC** after the name of field that should be sorted in descending order.

If WB_Order variable is not set and WB_ChangeHFOn variable is defined, WB_Order variable will get exactly the same value as WB_ChangeHFOn. If both variables are defined, then WB_Order must start with same fields as defined in WB_ChangeHFOn **but it can have** more fields in the list.

22.1.19.23 WB_Pass - database password

Syntax
WB_Pass=*user password*

Syntax example
WB_Pass=mypassword

Valid inputs
Any valid password depending upon database type. Password can not start with numeric character.

Default value
none

Description
This variable contains user password for confirming identification in database system.

22.1.19.24 WB_Predicate - SQL predicate

Syntax
WB_Predicate=*SQL predicate*

Syntax example
WB_Predicate=TOP 10%

Valid inputs
One of the following predicates: ALL, DISTINCT, DISTINCTROW, or TOP.

Default value
none

Description
This variable defines the SQL predicate to restrict the number of records returned. If none is specified, the default is ALL.

22.1.19.25 WB_Query - SQL clause "WHERE"

Syntax
WB_Query=*condition*

Syntax example
WB_Query=date=$wbfn{date}

Valid inputs
Any valid condition that can be used as SQL WHERE clause

Default value
none

Description
This variable contains condition (or more conditions) that will be used as WHERE clause of SQL SELECT statement. It can contain any type of condition respecting SQL syntax and current recordset structure. When it is set in subreport WBSP file, this variable **can contain** the $WBF function in report format. In this case $WBF function will refer to superior recordset (the one defined in file containing the sub report - e.g.WB_Query=PubID=$WBF[PubID]).

22.1.19.26 WB_RcdSet - SQL clause "FROM"

Syntax
WB_RcdSet=*table name or recordset definition (SQL FROM clause)*

Syntax example
WB_RcdSet=titles inner join publishers on titles.PubID=Publishers.PubID

Valid inputs
Any valid SQL FROM clause.

Default value
none

Description
This variable contains recordset definition (cross-table query or table name). If you use ISAM database then table name is the same as the file name without extension, except for WB_Connect=Text; (for example to open table authors.dbf you don't need the dbf extension). This variable is required if WB_Command has any of these values: Q, U, D, A and L. It is also required when WB_Command has value P and WB_BaseName is not empty.

22.1.19.27 WB_ReadOnly - open the database in read-only mode

Syntax
WB_ReadOnly=*boolean value*

Syntax example
WB_ReadOnly=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WBSP engine will open the database in read-only mode. This is required when WBSP engine is used on read-only media (such as web on CD-ROM). Placing WBSP engine and wbsp pages on read-only media also requires setting WB_Exclusive variable to TRUE.

22.1.19.28 WB_SetADOCompatible - ANSI wildcard compatibility

Syntax
WB_SetADOCompatible=*boolean value*

Syntax example
WB_SetADOCompatible=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WBSP engine will change asterisk wildcard (*) in WB_Query variables with percent sign (%) - ANSI wildcard. **This variable should be used only in projects built with previous versions of WBSP that extensively use WB_Query variable and asterisk wildcard.**

22.1.19.29 WB_ShowEmpty - display empty database fields as space

Syntax
WB_ShowEmpty=*boolean value*

Syntax example
WB_ShowEmpty=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WBSP engine will return a non breaking space character (&nbsp;) for empty database fields. In some cases it can be useful to set this variable to true (when using $wbdetail function or WB_TempName=$default$) because HTML tables do not render empty cells same way as they do ones with content (this default behavior of <table ...> element can be changed using CSS).

22.1.19.30 WB_StartRec - internal page counter

Syntax
WB_StartRec=*numeric value*

Syntax example
WB_StartRec=30

Valid inputs
Any numeric value

Default value
1

Description
This variable is used as WhizBase internal page counter. It is generated automatically in navigation links, but it can also be very useful in some special cases - e.g. when WBSP engine updates the record it is much better to redirect user to same page of the report then to redirect him to first page, or when for some reason you do not want your report to start with first record, etc.

22.1.19.31 WB_System - system database (MDA/MDW) file name

Syntax
WB_System=*file name of the system database*

Syntax example
WB_System=/database/system.mdw

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
Sets the path for the current location of the workgroup information file (MS Access system database).

22.1.19.32 WB_UID - unique record identifier field(s)

Syntax
WB_UID=*database field name*

Syntax example
WB_UID=PubID

Valid inputs
Any field name (or comma separated field list) from current recordset

Default value
none

Description
This variable is used to define what database fields uniquely identify every single record. It is required for updating the records, but it can also be very useful for adding and deleting the records. It can have a single database field name or comma-separated list of fields.
When WB_UID is defined the value in fields that are used in WB_UID cannot be changed. WB_UID field have effect with commands (actions) (A)dd, (D)elete and (U)pdate.

When WB_UID is defined WhizBase will disallow adding records with duplicate values for WB_UID fields and will also disallow deleting more than one record at a time.

22.1.19.33 WB_Unicode - send field value as unicode

Syntax
WB_Unicode=*boolean value*

Syntax example
WB_Unicode=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WhizBase engine will send the report in Unicode format. This variable has effect only if WB_Command value is set to Q. This variable should be used only with clients that require content in Unicode format. For converting strings to UTF-8 charset use $WBFN[UTF(anystring)] and $WBFU[fieldname].

22.1.19.34 WB_UniFTS - field(s) to be included in Universal Query Search

Syntax
WB_UniFTS=*database field name*

Syntax example
WB_UniFTS=Title

Valid inputs
Any field name (or comma separated field list) from current recordset

Default value
none

Description
Purpose of this variable is to narrow Universal Query Search (started when WhizBase receives value for WB_UniQS variable) to listed fields only. **Fields must NOT be enclosed in square brackets.**

22.1.19.35 WB_UniQS - string to be searched for in Universal Query Search

Syntax
WB_UniQS=*search string*

Syntax example
<input type="text" name="WB_UniQS" size="20">

Valid inputs
Any search string including SQL pattern rules

Default value
none

Description
This variable contains Universal Query String form field. When WhizBase receives WB_UniQS it will return all records that contain the word(s) or phrase (enclosed in quotation marks) provided by WB_UniQS variable, in any part of any non-numeric field (or fields defined in WB_UniFTS variable) in recordset (whole field, beginning, end or in the middle of the field).

This field can not be set in WBSP file, but it must be a part of a request (GET or POST form field).

22.1.19.36 WB_WC - database wildcard character for LIKE comparison

Syntax
WB_WC=*wildcard character*

Syntax example
WB_WC=%

Valid inputs
Any valid wildcard supported by database provider

Default value
* for DAO and % for ADO connections

Description
This variable contains the wildcard character to be used with WB_AddJoker. WhizBase automatically selects the value for this field depending on connection type (* for DAO and ISAM and % for ODBC and ADO), but this field can be used to set this value manually if for any case some other character has to be used. **In most cases this field has to have default value and we strongly recommend not to change it.**

22.1.19.37 WB_WholeWord - whole word search

Syntax
WB_WholeWord=*boolean value*

Syntax example
WB_WholeWord=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WBSP engine will add pattern string [!A-Z0-9a-z] at the beginning and at the end of the values provided by WBF_ and WB_UniQS form fields. In most cases this variable should not be changed from it's default value. It is not removed for compatibility reasons.

22.1.19.38 WB_Usr - database user name

Syntax
WB_Usr=*user name*

Syntax example
WB_Usr=someuser

Valid inputs
Any valid user name depending upon database type. User name can not start with numeric character.

Default value
none

Description
This variable contains user name for confirming identification in database system.

22.1.19.39 WBF_field - sending field values as request variables

Syntax
WBF_*DBFieldName=value for DBFieldName*

Syntax example (GET method)
WBF_ISBN=0201694085

Syntax example (POST method)
< input type="hidden" name="WBF_ISBN" value="0201694085">

Valid inputs
Any text and/or numeric value, including WhizBase tags and functions, that will be accepted by DBField (e.g. do not send text to a numeric or date/time field).

Default value
none

Description
Form fields with name starting with WBF_ are used to carry a value for specific database field. The name of the field is specified **after** WBF_ prefix and it has to be **exactly the same as in the database**. The action that will be performed with received value depends upon value of the WB_Command and (for some WB_Command values) WB_UID. Basically the most common case is that this value will be used to filter records (WB_Command values Q, D, P, L). In other cases the value will be stored in database field (WB_Command values A and U). **The exception to this rule** is if field name following the WBF_ prefix is the same as field name defined in WB_UID variable and WB_Command value is U. In this case

received value for that field will be used to identify the record to be updated and all other WBF_ values will be stored to proper database fields.

The best way to illustrate usage of these fields is example.

```
[FormFields]
WB_Command=q
WB_Basename=biblio.mdb
WB_Rcdset=titles
wb_showlogo=F
[MsgAndLbl]
WB_Style=font-family:verdana;font-size:12px;color:#CC0000;
<!--WB_BeginTemplate-->
<html>
<head>
<style>
.wbspttbl{
border:1px solid #000000;
font-family:verdana;
font-size:12px;
border-collapse:collapse;
border-spacing:0px;
}
.wbspthdr{
background-color:#CC0000;
border:1px solid #000000;
color:#C0C0C0;
}
.wbsptrow{
background-color:#FFCC00;
border:1px solid #000000;
color:#0000CC;
}
</style>
<title>Simple database search example</title>
</head>
<body>
<form method="POST" action="$wbe[script_name]">
Title: <input type="text" name="WBF_Title" size="20"><br>
Year published: <input type="text" name="WBF_Year published"
size="20"><br>
<input type="radio" value="And" checked name="WB_AndOr">AND <input
type="radio" name="WB_AndOr" value="Or">OR<br>
<input type="submit" value="Go" name="B1">
</form>
$wbdetail[t]
</body>
</html>
```

Run the file with this example code and experiment with various values for "Title" and "Year published" fields, and try changing AND/OR form values.

There are few rules:

- To let user search through specific field just send form field named exactly as database field with prefix WBF_ (don't forget the underscore) to a WBSp file with WB_Command=Q

- Depending on WB_AndOr field WBSP engine will return records that meet all the conditions (AND) or any of the conditions (OR) (WB_Command values Q, D, P or L)
- If user leaves any of the fields empty, those fields will be ignored (all WB_Command values)
- If WB_Command value is Q, D, P or L:
    - All SQL pattern rules are supported
    - In Numeric and Date fields user can enter characters for comparison at the beginning of entered value (> greater than, < less then, >= greater or equal,<= less or equal and <> not equal)
      e.g.
      <input type="hidden" name="WBF_Year published" value=">1999">
      <input type="hidden" name= "WBF_Yearpublished"value="=<2000">
      Try typing comparison characters as a part of form field in example above
- If WB_Command value is U
    - To update database field to NULL value send exactly the same value in WBF_field and WB_Null variables (text and memo field types only)
    - To store new value that is related to old one use update prefixes

These variables **can not** be part of WBSP file - they can only be sent by client as a part of http (or https) request using **either POST or GET method**.

22.1.20 DB administering (DAO only)

These variables are used to modify database and table permissions, users, groups and passwords for MS Access *.MDB databases. They can not be used with other database types including MS Access 2007 files (*.ACCDB).

WB_DBAddData
WB_DBAdmin
WB_DBDelData
WB_DBEditData
WB_DBGroup
WB_DBModDes
WB_DBNewPass
WB_DBNPassCh
WB_DBOldPass
WB_DBReadData
WB_DBReadDes
WB_DBUser
WB_PID

22.1.20.1 WB_DBAddData - permission for adding records to the database

Syntax
WB_DBAddData=*boolean value*

Syntax example
WB_DBAddData=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for adding records to the database table (defined in** WB_RcdSet**)** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.2 WB_DBAdmin - permission for administering the database

Syntax
WB_DBAdmin=*boolean value*

Syntax example
WB_DBAdmin=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for administering the database** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.3 WB_DBDelData - permission for deletinging records from the database

Syntax
WB_DBDelData=*boolean value*

Syntax example
WB_DBDelData=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for deleting records from the database table (defined in** WB_RcdSet**)** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE

WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.4 WB_DBEditData - permission for updating records in the database

Syntax
WB_DBEditData=*boolean value*

Syntax example
WB_DBEditData=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for updating records in the database table (defined in** WB_RcdSet**)** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.5 WB_DBGroup - name of the database users' group

Syntax
WB_DBGroup=*string value*

Syntax example
WB_DBGroup=EmlUsers

Valid inputs
Any valid group name

Default value
none

Description

WB_Command value	WB_DBGroup contains
AU and DU	the name of group to be added to or deleted from workgroup information file (system database)
AG and DG	the name of the group to which new user (defined in WB_DBUser) will be added or from which it will be deleted
SP	the name of the group to which new permissions will be assigned

22.1.20.6 WB_DBModDes - permission for modifying the database structure

Syntax
WB_DBModDes=*boolean value*

Syntax example
WB_DBModDes=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for modifying the design of the database table (defined in** WB_RcdSet**)** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.7 WB_DBNewPass - new database password

Syntax
WB_DBNewPass=*string value*

Syntax example
<input type="password" name="WB_DBNewPass" size="20">

Valid inputs
Any string value not starting with numeric character

Default value
none

Description
This variable contains a new password for user defined in WB_DBUser when WB_Command value is CP.

This value cannot be set in any WBSP file. WBSP will accept it only from HTML form.

22.1.20.8 WB_DBNPassCh - control value of the new password

Syntax
WB_DBNPassCh=*string value*

Syntax example
<input type="password" name="WB_DBNPassCh" size="20">

Valid inputs
Same value as entered for WB_NewPass variable

Default value
none

Description
This variable contains control value for new password for user defined in WB_DBUser when WB_Command value is CP. If this variable is not empty it **MUST** match WB_NewPass or the error will be generated.

This value cannot be set in any WBSP file. WBSP will accept it only from HTML form.

22.1.20.9 WB_DBOldPass - old database password

Syntax
WB_DBOldPass=*string value*

Syntax example
```
<input type="password" name="WB_DBOldPass" size="20">
```

Valid inputs
Exact value of user's existing password

Default value
none

Description
This variable contains control value for **old** password for user defined in WB_DBUser when WB_Command value is CP. If this variable does not match existing password, the error will be generated.

This value cannot be set in any WBSP file. WBSP will accept it only from HTML form.

22.1.20.10 WB_DBReadData - permission for reading records from the database

Syntax
WB_DBReadData=*boolean value*

Syntax example
```
WB_DBReadData=T
```

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for reading records to the database table (defined in WB_RcdSet)** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.11 WB_DBReadDes - permission for reading records the database design (structure)

Syntax
WB_DBReadDes=*boolean value*

Syntax example
WB_DBReadDes=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
none

Description
This variable sets a value that establishes the permissions **for reading the design of the database table** for the user or group identified by the WB_DBUser or WB_DBGroup of a table (DAO only). If it has value TRUE WhizBase will assign this permission to the user. If this field is omitted or it has value FALSE WhizBase will deny this permission to the user. This variable has effect only if the WB_Command value is SP.

22.1.20.12 WB_DBUser - database user name for administering

Syntax
WB_DBUser=*string value*

Syntax example
WB_DBUser=JohnDoe

Valid inputs
Any valid user name

Default value
none

Description

WB_Command value	WB_DBUser contains
AU and DU	the user name to be added to or deleted from workgroup information file (system database)

AG and DG	the user name that will be added to or deleted from group (defined in WB_DBGroup)
SP	the user name to which new permissions will be assigned

22.1.20.13 WB_PID - personal identification

Syntax
WB_PID=*alphanumeric value*

Syntax example
WB_PID=JD234

Valid inputs
Any combination of alphanumeric characters

Default value
Same value as WB_DBUser or WB_DBGroup if WB_DBUser has no value

Description
This field contains user's personal identification to be used with user's name defined in WB_DBUser when WhizBase executes Add User command (WB_Command=AU).

22.1.21 Error reporting

These variables are used for formatting error reports.

WB_ErrFile
WB_ErrMail

22.1.21.1 WB_ErrFile - template for error reports

Syntax
WB_ErrFile=*file name*

Syntax example
WB_ErrFile=/noread/fileerror.htm

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
error.htm

Description
This variable contains the file name of the file that should be used as a template for error messages. If the specified file can not be found, WhizBase will generate standard error message page.

22.1.21.2 WB_ErrMail - email address at the end of error message

Syntax
WB_ErrMail=*email address*

Syntax example
WB_ErrMail=errorreport@wbsp.com

Valid inputs
Any valid e-mail address

Default value
webmaster@$wbe[server_name]

Description
This variable contains e-mail address that will appear at the end of error message, right after the sentence **Then contact the administrator of this service:**
This e-mail address is also used for sending the test e-mail message (WB_Command value T).

22.1.22 File related (WF and DF commands)

These variables are used with two WB_Command values - DF and WF. In all other cases their values are ignored by WhizBase.

WB_FileName
WB_KeyName
WB_KeyValue
WB_Section
WB_Separator

22.1.22.1 WB_FileName - file name for DF and WF commands

Syntax
WB_FileName=*file name*

Syntax example
WB_FileName=/noread/users.cfg

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
This variable contains the file name of the configuration file that should be used with WB_Command values DF (file will be deleted) and WF (key value vill be written in file). This variable is required if WB_Command value is DF or WF, and it is ignored in all other cases.

22.1.22.2 WB_KeyName - variable name(s) for WF command

Syntax
WB_KeyName=*key name*

Syntax example
WB_KeyName=BgndColor

Valid inputs
Any alphanumeric string or list of strings separated by WB_Separator

Default value
none

Description
This variable contains the name(s) of the key(s) that will be changed/added/deleted using WB_Command value WF. This variable is required if WB_Command value is WF, and it is ignored in all other cases. If it contains list of names separated by WB_Separator, then the number of names must match the number of values in WB_KeyValue variable.

22.1.22.3 WB_KeyValue - variable value(s) for WF command

Syntax
WB_KeyValue=*key value*

Syntax example
WB_KeyValue=#C0C0C0

Valid inputs
Any alphanumeric string or list of strings separated by WB_Separator

Default value
none

Description
This variable contains the value(s) for the key(s) that will be changed or added using WB_Command value WF. **If this variable is empty or it contains an empty string in list, then the matching key name will be deleted**. This variable is required if WB_Command value is WF, and it is ignored in all other cases. If it contains list of values separated by WB_Separator, then the number of values must match the number of names in WB_KeyName variable.

22.1.22.4 WB_Section - file section for WF command

Syntax
WB_Section=*section name*

Syntax example
WB_Section=Colors

Valid inputs
Any alphanumeric string

Default value
none

Description
This variable contains the name of the section containing the key that will be changed/added/deleted using WB_Command value WF. This variable is required if WB_Command value is WF, and it is ignored in all other cases.

22.1.22.5 WB_Separator - character used to separate different keys and different values for WF command

Syntax
WB_Separator=*separator character*

Syntax example
WB_Separator=;

Valid inputs
Any character

Default value
none

Description
This variable contains the character which will be used to separate different entries in the WB_Keyname and WB_KeyValue variables. This variable is required if WB_Command value is WF, and it is ignored in all other cases. Following example shows how it should be used:

```
[FormFields]
WB_KeyName=Airpressure|Temperature|Humidity
WB_KeyValue=$wbv{fAirP};|$wbv{fTemp};|$wbv{fHum};
WB_Separator=|
WB_Section=$wbfn{fdt(dd-mmm-yyyy hh:mm)}
WB_FileName=Weather.cfg
WB_Command=WF
<!--WB_BeginTemplate-->
<html>
<body>
Values sucessfully modified!
</body>
</html>
```

And this is what might be the result written in file Weather.cfg upon calling this script using following form field values < B>-fAirP=1034&fTemp=20&fHum=85:

```
[16-sep-2008 15:05]
Airpressure=1034;
Temperature=20;
Humidity=85;
```

22.1.23 HTTP

These variables are used to modify the default content of a HTTP header sent to the client. They can be very powerful tools in building complex web solutions.

WB_HTTPHeader
WB_AddCookie
WB_ContentType
WB_Redirect

22.1.23.1 WB_AddCookie - name and value of the cookie to be added/modified

Syntax
WB_AddCookie=*cookiename=cookie value;*

Syntax example
WB_AddCookie=LoginTime=$wbfn{fdt(dd.mmm.yyyy hh:mm:ss)};

Valid inputs
Any text and/or numeric value including WhizBase tags and functions.

Default value
none

Description
Content of this variable will be sent to client as **Set-Cookie:** clause in HTTP header and added to existing HTTP_COOKIE value. HTTP_COOKIE can be retrieved using WBSP function $wbgc[cookiename]. All existing HTTP cookies can be retrieved using $wbe[HTTP_COOKIE] . Different cookies must be separated by double semicolon **(;;)**, and entire value of WB_SetCookie must end with semicolon **(;)** .

22.1.23.2 WB_ContentType - value for HTTP Content-Type: clause

Syntax
WB_ContentType=*any valid MIME type*

Syntax example
WB_ContentType=application/rss+xml

Valid inputs
Any text and/or numeric value including WhizBase tags and functions.

Default value
text/html

Description
Content of this variable will be sent to client as **Content-Type:** clause in HTTP header.

22.1.23.3 WB_HTTPHeader - additional clauses for HTTP header

Syntax
WB_HTTPHeader=*any valid HTTP header value*

Syntax example
WB_HTTPHeader=Content-disposition: attachment; filename=report.rtf

Valid inputs
Any text and/or numeric value including WhizBase tags and functions. If you want to send HTTP header that contains more than one line, save the header to file and use $wbinc{} or $wbrinc{} (e.g. WB_HTTPHeader = $wbinc{header.ic})

Default value
none

Description
Content of this variable will be sent to client as part of standard HTTP header together with other values (Status, Content-type, etc.).

22.1.23.4 WB_Redirect - the URL for 301 and 302 redirect (HTTP header Location: clause)

Syntax
WB_Redirect=*any valid URL*

Syntax example (HTTP temporary redirect - 302)
WB_Redirect=http://$wbe{server_name}/thanks.wbsp?user=$wbf[username]

Syntax example (HTTP permanent redirect 301)
WB_Redirect=PERM:http://$wbe{server_name}/new_page_url

Syntax example (server-side redirect)
WB_Redirect=WBSP:/thanks.wbsp?user=$wbf[username]

Valid inputs
Any text and/or numeric value including WhizBase tags and functions. This variable **can** contain WhizBase functions both in input and report syntax. Functions in input syntax will be processed **when WhizBase reads the variable**. Functions in report syntax will be processed **after processing all variables and completing the current command**.

Default value
none

Description
This variable contains URL of the web content where the WBSP engine will redirect visitor after completing the requested operation. **It can be used in two ways** – it can be set to perform normal redirect (send to client HTTP header **Location:** clause) or it can be set to perform server-side redirect (redirecting to other WBSP file without contacting client – visitor's browser, for example). To perform server-side redirect this value must have the name of the WBSP file with prefix **WBSP:**, including colon character **(:)**.

Whizbase performs temporary (**302**) redirect by default. For permanent redirect (**301**) please add prefix **PERM:** to the URL including colon character **(:)**.

22.1.24 Logging

These variables are used for enabling and formatting logging.

WB_Log
WB_Logdata
WB_LogTemp
WB_Debug

22.1.24.1 WB_Debug - file name for storing the debug information

Syntax
WB_Debug=*file name*

Syntax example
WB_Debug=/noread/debug.log

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
This variable contains the file name of the file that should be used for storing the debug information in following format:
The variable WB_Debug is used to pass to WBSP the name of the file where WBSP will save the debug information in following format:

```
**DATE TIME**
Form data:
Data posted using POST method
Data posted using POST method
Data posted using POST method
...
Data posted using POST method
------------EOFormDataSection------------
Environ:
Environment variable
Environment variable
Environment variable
...
Environment variable
------------EOEnvironSection------------
```

If this variable is not specified no debug information will be saved.

22.1.24.2 WB_Log - file name for storing the log data

Syntax
WB_Log=*file name*

Syntax example
WB_Log=/noread/useraccess.log

Valid inputs
Any valid file name respecting WhizBase path rules

Default value
none

Description
This variable contains the file name of the file that should be used for storing log data. If this variable is not specified no logging will be performed.

22.1.24.3 WB_LogData - data to be stored in log

Syntax
WB_LogData=*comma-separated list of variables*

Syntax example
WB_LogData=HTTP_USER_AGENT, REMOTE_ADDR, WB_QUERY, WB_HTUSR

Valid inputs
Any valid environment or WhizBase variable

Default value
HTTP_USER_AGENT, REMOTE_ADDR, REMOTE_HOST, REMOTE_IDENT, REMOTE_USER, WB_TEMPNAME, WB_QUERY, WB_ADDJOKER, WB_ORDER, WB_USR

Description
This variable contains comma-separated list of WhizBase and environment variables that will be added to the log file specified in WB_Log variable. Date and time are added automatically.

22.1.24.4 WB_LogTemp - template for log record

Syntax
WB_LogTemp=*log line template*

Syntax example
WB_LogTemp=$wbfn[date] $wbfn[time] - $wbe[HTTP_USER_AGENT], $wbe[REMOTE_ADDR], $WBQuery

Valid inputs
Any valid WhizBase tag or function in input or report syntax

Default value
none

Description
This variable contains a template for a single line in log file. It can contain both input functions and report tags and functions. WhizBase will first process the input functions, then report tags and functions and then add processed line to the log file. If values for both variables WB_LogData and WB_LogTemp are defined, the WB_LogData will be ignored.
WhizBase will not add to log line any value that is not defined in WB_LogTemp.

22.1.25 Mail related

These variables are used to configure mail related settings for use with WB_Command values P, L and T.

WB_Attach
WB_AttachField
WB_BCC
WB_BCCField
WB_CC
WB_From
WB_Embed
WB_MailAuth
WB_MailPass
WB_MailPort
WB_MailServer
WB_MailSSL
WB_MailUser
WB_PlainText
WB_Subject
WB_To
WB_ToField

22.1.25.1 WB_Attach - mail attachment

Syntax
WB_Attach=*attachment file name*

Syntax example
WB_Attach=$wbv{attachfile}

Valid inputs
Any valid file name (or list of files separated by comma or semicolon) respecting WhizBase path rules

Default value
none

Description
This variable contains full path (absolute or relative to the location of WBSP file) and file name of the file(s) that will be attached to the email sent by WBSP engine.

22.1.25.2 WB_AttachField - name of field containing attachment file name(s)

Syntax
WB_AttachField=*database field name*

Syntax example
WB_AttachField=AttachVersion

Valid inputs
Any valid field name from current recordset

Default value
none

Description
This variable contains the name of the field from current recordset that contains the full path and file name of file(s) that will be attached to the email message sent by WhizBase. Unlike WB_Attach variable that is used for sending same file(s) to all recipients, WB_AttachField is used when there is a need to send different attachment to every recipient. The field defined as WB_AttachField should contain valid file name(s) respecting WhizBase path rules.

22.1.25.3 WB_BCC - email BCC address(es)

Syntax
WB_BCC=*email address(es)*

Syntax example
WB_BCC=support@whizbase.com

Valid inputs
Any valid email address or list of addresses separated with list separator character (as defined in regional settings).

Default value
none

Description
This variable contains email address(es) for blind carbon copy recipient(s).

22.1.25.4 WB_BCCField - database field name containing email BCC address(es)

Syntax
WB_BCCField=*database field name*

Syntax example
WB_BCCField=Email

Valid inputs
Any valid field name from current recordset

Default value
none

Description
This variable contains **the name of the field from current recordset** that contains the email address to be added as blind carbon copy recipient.

22.1.25.5 WB_CC - email CC address(es)

Syntax
WB_CC=*email address(es)*

Syntax example
WB_CC=support@whizbase.com

Valid inputs
Any valid email address or list of addresses separated with list separator character (as defined in regional settings).

Default value
none

Description
This variable contains email address(es) of carbon copy recipient(s).

22.1.25.6 WB_Embed - the name(s) of the file(s) to be embedded in the email

Syntax
WB_Embed=*attachment file name*

Syntax example
WB_Embed=$wbv{embedfile}

Valid inputs
Any valid file name (or list of files separated by comma or semicolon) respecting WhizBase path rules

Default value
none

Description
This variable contains full path (absolute or relative to the location of WBSP file) and file name of the file(s) that will be embedded to the email sent by WBSP engine. The difference from WB_Attach variable is that embedded images can be referenced in message body by setting their URL to $WBCID[file name] (see the example for $WBCID function).

22.1.25.7 WB_From - email from address

Syntax
WB_From=*email address*

Syntax example
WB_From=support@whizbase.com

Valid inputs
Any valid email address

Default value
none

Description
This variable contains email address that will appear as sender's email address to originate from.

22.1.25.8 WB_MailAuth - mail server authentication type

Syntax
WB_MailAuth=*authorization type*

Syntax example
WB_MailAuth=L

Valid inputs
Any valid type from following table:
0 - None - No authentication required.
L - Login - Authentication using LOGIN type.
P - Plain - Authentication using PLAIN type.
C - CramMD5 - Authentication using CRAM-MD5 type.
N - NTLM - Authentication using NTLM (SPA) type.
A - Auto - WhizBase tries to determine authentication automatically.

Default value
0 (zero) - No authentication required.

Description
This variable determines authentication type with the server.

22.1.25.9 WB_MailPass - mail server authentication password

Syntax
WB_MailPass=*mailserver password*

Syntax example
WB_MailPass=MjZ6hZTgDsE

Valid inputs
Valid password for account defined in WB_MailUser .

Default value
none

Description
This variable contains password for SMTP authentication.

22.1.25.10 WB_MailPort - mail server SMTP port

Syntax
WB_MailPort=*numeric value*

Syntax example
WB_MailPort=527

Valid inputs
Any valid SMTP port number on defined mail server

Default value
25

Description
This variable contains the Internet port number of the mail server that will be used by WBSP engine for sending mail.

22.1.25.11 WB_MailServer - mail server name or IP address

Syntax
WB_MailServer=*mail server name*

Syntax example
WB_MailServer= mail.whizbase.com

Valid inputs
Any valid SMTP mail server

Default value
none

Description
This variable contains the name of the mail server that will be used by WBSP engine for sending mail. It does not have to be same server that hosts WBSP files. It also does not have to be Windows SMTP server – it can also be on UNIX, LINUX, etc. This variable is required if WB_Command value is P or L.

22.1.25.12 WB_MailSSL - mail server authentication type

Syntax
WB_MailSSL=*SSL type*

Syntax example
WB_MailSSL=I

Valid inputs
Any valid type from following table:
0 - No security/SSL required
A - Security/SLL is allowed, but not required
R - Security is required
I - Implicit security through SSL wrapper

Default value
0 (zero) - No security/SSL required.

Description
This variable determines if SSL is used.

22.1.25.13 WB_MailUser - mail server authentication user name

Syntax
WB_MailUser=*mailserver user name*

Syntax example
WB_MailUser=someuser@somedomain.com

Valid inputs
Any valid username that exist on mail server defined in WB_MailServer .

Default value
none

Description
This variable contains login (user name) for SMTP authentication.

22.1.25.14 WB_PlainText - plain text email messsage part

Syntax
WB_PlainText=*text/plain part of the message*

Syntax example
WB_PlainText=Thank you for your submission! You can find more details at http://www.whizbase.com/formsubmissions/?id=$wbf[id]

Valid inputs
Any text without HTML code. It can contain WhizBase report and input functions and tags.

Default value
none

Description
This variable contains text/plain part of the e-mail message for non-HTML email clients.

22.1.25.15 WB_Subject - email subject

Syntax
WB_Subject=*any string value*

Syntax example
WB_Subject=Report for $wbfn{fdt(dd-mmm-yyyy)}

Valid inputs
Any string value

Default value
none

Description
This variable contains the text that will be used as subject of the email(s) sent by WhizBase using WB_Command P or L.

22.1.25.16 WB_To - email TO address(es)

Syntax
WB_To=*email address(es)*

Syntax example
WB_To=support@whizbase.com

Valid inputs
Any valid email address or list of addresses separated with list separator character (as defined in regional settings).

Default value
none

Description
This variable contains email recipient(s) email address(es).

22.1.25.17 WB_ToField - database field name containing email TO address(es)

Syntax
WB_ToField=*database field name*

Syntax example
WB_ToField=Email

Valid inputs
Any valid field name from current recordset

Default value
none

Description
This variable contains **the name of the field from current recordset** that contains valid email address to be used as **TO** address.

22.1.26 Sessions

These variables are used for enabling and maintaining sessions.

WB_ClearSessions
WB_LogOffSession
WB_UseSessions

22.1.26.1 WB_ClearSessions - clear incative (expired) sessions

Syntax
WB_ClearSessions=*boolean value*

Syntax example
WB_ClearSessions= T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable controls if the current WBSP page will start session cleaning routine for removing inactive sessions and their data. If this variable is set to TRUE WhizBase will start the cleaning routine **after it finish processing the WBSP page**. The WBSP page containing this variable **does not have to use sessions,** meaning that WB_UseSessions can, but does not have to, be set to TRUE. The same result is achieved with usage of report tag $WBACTSES.

22.1.26.2 WB_LogOffSession - clear current session

Syntax
WB_LogOffSession=*boolean value*

Syntax example
WB_LogOffSession= T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
If this variable is set to TRUE WhizBase will stop a session by removing session ID

from HTTP cookie. After this, if same client (browser) opens a WBSP page with WB_UseSessions set to TRUE, WhizBase will start a completely new session.

22.1.26.3 WB_UseSessions - use server sessions

Syntax
WB_UseSessions=*boolean value*

Syntax example
WB_UseSessions= T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable controls if the current WBSP page will use sessions and have access to the session variables. If this variable is set to TRUE WhizBase will be able to read/save session variables. If session does not exist WhizBase will create one automatically.

22.1.27 SMS

These variables are used to configure SMS related settings for use with WB_Command value SMS .

WB_SMSBR
WB_SMSC
WB_SMSCharacter
WB_SMSDB
WB_SMSField
WB_SMSIgnoreErrors
WB_SMSNumber
WB_SMSParity
WB_SMSPIN
WB_SMSPort
WB_SMSSB
WB_SMSSD
WB_SMSSR
WB_SMSTO

22.1.27.1 WB_SMSBR - SMS baud rate

Syntax
WB_SMSBR=*phone/modem baud rate*

Syntax example
WB_SMSBR=57600

Valid inputs
110, 300, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 230400, 460800, 921600

Default value
19200

Description
This variable contains the baud rate at which the connected GSM Modem or Phone communicates with the computer.

22.1.27.2 WB_SMSC - SMS center number

Syntax
WB_SMSC=*service center number*

Syntax example
WB_SMSC=+38761123456

Valid inputs
Any valid phone number containing digit characters and a plus (+) sign (for international number format)

Default value
none

Description
This variable contains the service centre number which will be used for sending text messages. If this variable is not specified the system will use the service center number defined in the modem/phone.

22.1.27.3 WB_SMSCharacter - SMS character type

Syntax
WB_SMSCharacter=*character encoding type*

Syntax example
WB_SMSCharacter=ANS

Valid inputs
7BIT - 7-bit default SMS character encoding (max. 160 characters/message)
ANS - 8-bit ANSI character encoding (max. 140 characters/message)
UNI - 16-bit 8-bit Unicode character encoding (max. 70 characters/message)

Default value
7BIT

Description
This variable contains the type of character encoding to be used for sending text messages.

22.1.27.4 WB_SMSDB - SMS data bits

Syntax
WB_SMSDB=*data bits size*

Syntax example
WB_SMSDB=8

Valid inputs
4, 5, 6, 7, 8

Default value
8

Description
This variable contains the communication Data Bits size between GSM Modem or Phone and the computer.

22.1.27.5 WB_SMSField - database field name containing phone number of text message recipient

Syntax
WB_SMSField=*database field name*

Syntax example
WB_SMSField=PhoneNum

Valid inputs
Any valid field name from current recordset

Default value
none

Description
This variable contains **the name of the field from current recordset** that contains valid phone number to be used as SMS recipient. If this variable is defined, the WB_SMSNumber variable will be ignored.

22.1.27.6 WB_SMSIgnoreErrors - ignore SMS error messages

Syntax
WB_SMSIgnoreErrors=*boolean value*

Syntax example
WB_SMSIgnoreErrors=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This is a True/False flag that determines if the WhizBase engine will ignore the SMS error messages when using SMS command with WB_SMSNumber. **This flag has no effects when WB_SMSField is defined**, so send report returned by SMS command will contain list of all errors in form of warnings.

22.1.27.7 WB_SMSNumber - SMS recipient phone number

Syntax
WB_SMSNumber=*phone number*

Syntax example
WB_SMSNumber=+387339414238

Valid inputs
Any valid phone number in national or international format.

Default value
none

Description
This variable contains SMS recipient phone number. If WB_SMSField is defined, the WB_SMSNumber will be ignored.

22.1.27.8 WB_SMSParity - SMS parity type

Syntax
WB_SMSParity=*parity type*

Syntax example
WB_SMSParity=O

Valid inputs
N - None
O - Odd
E - Even
M - Mark
S - Space

Default value
N

Description
This variable contains the type of parity check for communication between GSM Modem or Phone and the computer.

22.1.27.9 WB_SMSPIN - SMS Personal Identity Number (PIN)

Syntax
WB_SMSPIN=*phone/modem PIN*

Syntax example
WB_SMSPIN=0000

Valid inputs
Any four digit number

Default value
none

Description
This variable contains the PIN (Personal Identity Number) for activating the services in the GSM Modem or Phone, if it is PIN protected.

22.1.27.10 WB_SMSPort - SMS modem COM port

Syntax
WB_SMSPort=*com port name*

Syntax example
WB_SMSPort=COM5

Valid inputs
Any valid COM port name

Default value
none

Description
This variable contains the name of the COM port where the GSM modem (or mobile phone) is connected.

22.1.27.11 WB_SMSSB - SMS stop bits size

Syntax
WB_SMSSB=*stop bits size*

Syntax example
WB_SMSSB=1.5

Valid inputs
1, 1.5, 2

Default value
1

Description
This variable contains the communication Stop Bits size between GSM Modem or Phone and the computer.

22.1.27.12 WB_SMSSD - SMS send delay

Syntax
WB_SMSSD=*send delay in milliseconds*

Syntax example
WB_SMSSD=500

Valid inputs
1 to 30000

Default value
1000

Description
This variable contains the minimum time delay interval between two consecutive messages in milliseconds.

22.1.27.13 WB_SMSSR - SMS send retry

Syntax
WB_SMSSR=*number of retries*

Syntax example
WB_SMSSR=1

Valid inputs
0 to 10

Default value
2

Description
This variable contains the maximum number of retries after there is a failure in first attempt.

22.1.27.14 WB_SMSTO - SMS timeout

Syntax
WB_SMSTO=*timeout in milliseconds*

Syntax example
WB_SMSTO=10000

Valid inputs
1 to 60000

Default value
30000

Description
This variable contains the timeout value in milliseconds which is used whenever the GSM Modem or Phone stops responding.

22.2 MsgAndLbl variables - Subsection [MsgAndLbl]

This subsection contains the variables that define labels and style for navigation links (for previous, next, first and last page) and messages (empty recordset and deleted records). They can contain any valid HTML code including WhizBase report tags and functions both in input and output syntax.
It can also contain variables for formatting navigation links WB_DigitDir and WB_Style .

These are variables that can be stored in this subsection (in alphabetic order):
WB_DigitDir
WB_Style
WBL_FirstPage
WBL_LastPage
WBL_NextPage
WBL_PrevPage
WBM_Deleted
WBM_NoMatch

Here is the example of MsgAndLbl section (marked blue):

```
<!--
[FormFields]
WB_basename=biblio.mdb
wb_rcdset=titles
wb_command=Q
[MsgAndLbl]
WB_DigitDir=digits/
WB_Style=color:#CC0000; text-decoration: none
wbl_nextpage=<img src="images/rightarrow.gif" border="0">
wbl_prevpage=<img src="images/leftarrow.gif" border="0">
wbl_lastpage=<img src="images/last.gif" border="0">
wbl_firstpage=<img src="images/first.gif" border="0">
wbm_nomatch=Sorry!<br>Your search returned no records!<br><a >Please
try again with other parameters</a>
wbm_deleted=Operation completed!<br><b>$WBDeleted</b> record(s)
successfully removed from your database!
-->
<!--WB_BeginTemplate-->
<html>
<head>
<title>Titles</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<td>Year published</td>
<td>Title</td>
<td>ISBN</td>
</tr>
<!--WB_BeginDetail-->
<tr>
```

```
<td>$wbf[Year published]</td>
<td>$wbf[title]</td>
<td>$wbf[ISBN]</td>
</tr>
<!--WB_EndDetail-->
</table>
<center>$wbnavigator</center>
</body>
</html>
```

To test this example, create images rightarrow.gif, leftarrow.gif, first.gif and last.gif and place them in directory images located in same directory with this wbsp file. Also create images of the digits 0 to 9 (name them 0.gif, 1.gif, 2.gif, ..., 9.gif) and place them in directory digits located in same directory with this wbsp file.

22.2.1 WB_AddToURL - additional request variables for navigation URLs formated as QUERY_STRING

Syntax
WB_AddToURL=*a string that WhizBase should add to the end of all navigation links created using navigation tags.*

Syntax example
WB_AddToURL= selectedlanguage=eng&rowsbypage=4

Valid inputs
Any valid text including HTML tags. **It must not start with ampersand (&)**.

Default value
none (WhizBase will not add anything to navigation links)

Description
When WhizBase creates navigation links it adds standard WhizBase variables to the URL. However, it does not add non-WhizBase variables (form fields) received from visitor using POST or GET. This variable contains the string that will be added at the end of the link and will pass the variables to next page, without validating or changing them. **It will process the WhizBase functions and tags if they exist.**

22.2.2 WB_DigitDir - directory containing image files for graphic navigation links

Syntax
WB_DigitDir=*directory URL*

Syntax example
WB_DigitDir=/digits/

Valid inputs
URL of the directory that contains digit images named 0.gif, 1.gif, 2.gif, 3.gif,...,9.gif.

Default value
none (direct page links will use text, not images)

Description
This variable defines the name of the directory that contains digit images. If it is defined, direct links to the report pages will be created in following form:
<a href=*"linktoreportpage"*><*img src="directory URL/digitimage.gif"*
border="0"></a>
e.g.

```
<a><img src="/pgnumdigits/2.gif" border="0"></a>
```

If WB_DigitDir is not defined direct links to the report pages will be created using text:

<a href=*"linktoreportpage"*>*pagenumber*</a>
e.g.

```
<a>2</a>
```

22.2.3 WB_Style - navigation links CSS style

Syntax
WB_Style=*CSS style definition*

Syntax example
WB_Style=font-family:verdana;font-size:11px;font-weight:bold;

Valid inputs
Any valid CSS property

Default value
none (navigation links will use default style(s) used by current WBSP page)

Description
This variable defines the css style properties for navigation links that will be used on page in this form:
<a href=*"linktoreportpage" style="CSS style definition"*>*Next page*</a>

22.2.4 WBL_FirstPage - link text for First page link

Syntax
WBL_FirstPage=*text to be used as label for link to first page*

Syntax example
WBL_FirstPage=<img src="/images/fpg.gif" border="0">

Valid inputs
Any valid text that can be placed between <a > and </a> including HTML tags (IMG for example)

Default value
First page

Description
This variable defines the text or HTML code that will be used for link to first page of the report

22.2.5 WBL_LastPage - link text for Last page link

Syntax
WBL_LastPage=*text to be used as label for link to last page*

Syntax example
WBL_LastPage=<img src="/images/lpg.gif" border="0">

Valid inputs
Any valid text that can be placed between <a > and </a> including HTML tags (IMG for example)

Default value
Last page

Description
This variable defines the text or HTML code that will be used for link to last page of the report

22.2.6 WBL_NextPage - link text for Next page link

Syntax
WBL_NextPage=*text to be used as label for link to next page*

Syntax example
WBL_NextPage=<img src="/images/npg.gif" border="0">

Valid inputs
Any valid text that can be placed between <a > and </a> including HTML tags (IMG for example)

Default value
Next page

Description
This variable defines the text or HTML code that will be used for link to next page of the report

22.2.7 WB_PassVars - comma delimited list of additional request variables for navigation URLs

Syntax
WB_PassVars=*comma separated form field names that WhizBase should include in all navigation links created using navigation tags.*

Syntax example
WB_PassVars= name,zip,rowsbypage

Valid inputs
Any valid text including HTML tags

Default value
none (WhizBase will add only standard variables to navigation links)

Description
When WhizBase creates navigation links it adds standard WhizBase variables to the URL. However, it does not add non-WhizBase variables (form fields) received from visitor using POST or GET. This variable defines which form variables will be included in navigation link beside standard vars. The variables will be included inaform:varname1=varvalue1&varname2=varvalue2

22.2.8 WBL_PrevPage - link text for Previous page link

Syntax
WBL_PrevPage=*text to be used as label for link to previous page*

Syntax example
WBL_PrevPage=<img src="/images/ppg.gif" border="0">

Valid inputs
Any valid text that can be placed between <a > and </a> including HTML tags (IMG for example)

Default value
Previous page

Description
This variable defines the text or HTML code that will be used for link to previous page of the report

22.2.9 WBM_Deleted - message template for reporting deleted records

Syntax
WBM_Deleted=*text to be used as report after deleting records*

Syntax example
WBM_Deleted= Operation completed! <b>$WBDeleted</b> record(s) removed from your database!

Valid inputs
Any valid text including HTML tags

Default value
Command successful! $WBDeleted record(s) deleted!

Description
This variable defines the text or HTML code that will be displayed on the report when WBSP engine completes delete command.

22.2.10 WBM_NoMatch - text for reporting that search returned no records

Syntax
WBM_NoMatch=*text to be used as report when recordset contains no records*

Syntax example
WBM_NoMatch=<img src="/images/nomatch.gif"><script>alert("Your search returned no records!");history.back();</script>

Valid inputs
Any valid text including HTML tags

Default value
No matching records!

Description
This variable defines the text or HTML code that will be displayed on the report when WBSP engine completes query command and recordset contains no records.

22.3 Upload section variables - Subsection [Upload]

This subsection contains variables needed to control file upload using WBSP page. Using these variables, the developer can control where uploaded files will be saved, what is maximum size of a single file, what file types can not be uploaded, weather existing file will be overwritten or not, where to store log data and what prefix to add to URL of uploaded file. The variables that can be stored in this subsection are (in alphabetic order):
WB_BaseURL
WB_Disallow
WB_MaxFSize
WB_Overwrite
WB_UploadDir
WB_UploadLog

Here's an example of upload subsection (marked with blue):

```
[FormFields]
WB_BaseName=biblioA.mdb
WB_AllowMultipart=T
WB_Command=A
WB_UID=ISBN
WB_Redirect=titlesQ.wbsp
WB_RcdSet=Titles
[Upload]
WB_Disallow=![jpg,gif]
WB_UploadDir=images/
WB_Overwrite=T
WB_MaxFSize=24576
WB_UploadLog=upload.log
```

To learn more about uploading files using WBSP read the "Uploading files using WBSP" page.

22.3.1 WB_BaseUrl - URL prefix to be added to uploaded file name

Syntax
WB_BaseURL=*URL*

Syntax example
WB_BaseURL=/images/

Valid inputs
URL can be any valid relative address or absolute address of directory on same or other server. Absolute address can include protocol

Default value
The URL of the directory where current WBSP file is located

Description
This variable defines an URL prefix added to the value of the field used to upload file.

If WB_BaseURL value is	**Value of the form field used to upload file will be changed from mypic.gif to**
ftp://ftp.myserver.com/pub/	ftp://ftp.myserver.com/pub/mypic.gif
/images/	/images/mypic.gif
undefined *(no WB_BaseURL variable in WBSP file)*	*/pathtowbspfile/*mypic.gif

22.3.2 WB_Disallow - list of file types (extensions) that can not be uploaded

Syntax
WB_Disallow=*comma-separated list containing file extensions that are not allowed for upload*

Syntax example
WB_Disallow=![jpg,jpeg,gif,png]

Valid inputs
1. comma-separated list of disallowed file extensions in form
ext1,ext2,ext3,...,extN
2. comma-separated list of allowed file extensions in form
![ext1,ext2,ext3,...,extN]

Default value
wbsp

Description
This variable defines file types (extensions) that can not be uploaded using current WBSP file. It can contain either list of the **disallowed** file types, or a list of allowed file types enclosed in square brackets with preceded with exclamation mark

If WB_Disallow value is	Current WBSP file will
wbsp,asp,exe,bat,com	**allow upload** of all files except: *.wbsp *.asp *.exe *.bat *.com
![jpg,jpeg,gif,png]	**disallow upload** of all files except: *.jpg *.jpeg *.gif *.png

Important: This variable cannot be sent using HTML form unless you specify value $form$ (**WB_Disallow=$form$**) in WBSP file. Setting this variable value to $form$ could be unwise and very risky decision.

22.3.3 WB_MaxFSize - maximum size for a single uploaded file

Syntax
WB_MaxFSize=*maximum number of bytes*

Syntax example
WB_MaxFSize=65536

Valid inputs
Any valid integer.

Default value
none (the maximum file size for upload is not limited by default)

Description
This variable defines maximum size (**in bytes**) of the single file that can be uploaded using current WBSP file.

22.3.4 WB_Overwrite - overwrite existing file with uploaded one

Syntax
WB_Overwrite=*boolean value*

Syntax example
WB_Overwrite=T

Valid inputs
T,TRUE,1,ON for True
F,FALSE,0,OFF for False

Default value
FALSE

Description
This variable defines if the file with same name that already exists on the server be overwritten by newly uploaded file.

If WB_Overwrite value is	Existing file with same name will be
TRUE	overwritten
FALSE	unchanged, and new file will be saved using unique name generated by WhizBase

Important: This variable cannot be sent using HTML form unless you specify value $form$ (**WB_Overwrite=$form$**) in WBSP file. Setting this variable value to $form$ is not recommended.

22.3.5 WB_UploadDir - destination directory for uploaded files

Syntax
WB_UploadDir=*name of the directory*

Syntax example
WB_UploadDir=/upldimg/

Valid inputs
Any valid directory name respecting the WhizBase path rules.

Default value
Directory of the current WBSP file

Description
This variable defines the name of the directory on the server where Whizbase will save files uploaded using current WBSP file.

22.3.6 WB_UploadLog - file name for logging upload activities

Syntax
WB_UploadLog=*name of log file*

Syntax example
WB_UploadLog=/logs/upload.log

Valid inputs
Any valid file name respecting the WhizBase path rules.

Default value
wbupload.log

Description
This variable defines the name of the file where Whizbase will write log of uploaded files in format:
Date *date when file is uploaded*
Filename *original (local) file name*
Saved as *name of the uploaded file*
Content-Type *MIME type of the uploaded file*

Size *the size of uploaded file*
From *IP address and remote host name*

23. Error messages

23.1 Common system errors

These errors are not specific to WhizBase but the are related to system configuration, disk/file access permissions and regional settings.

23.1.1 Error 5 - Invalid procedure call or argument

This error can be generated for various reasons but most usual case is when WBSP engine receives invalid registration key value.
It can also be caused by invalid argument of $WBFN function.

23.1.2 Error 13 - Type mismatch

WhizBase is not very sensible to variable's data type. However, working with database fields require proper data type for every field type. This error occures when Numeric, Boolean and TimeDate field receives value that can not be converted (e.g. string instead of number or date in invalid format, etc.).

23.1.3 Error 75 - Path/File access error

This error is generated when WhizBase tries to write file in a directory where it has no write permissions.
To solve this assign the user that starts WhizBase (usually Internet Guest Account) full access permissions (read/write) to directories that require writing (e.g. upload directories, directories where log and database files are located, etc.).

23.1.4 Error 429 - ActiveX component can't create object.

This error is generated when DAO, ADO, Scripting server or HTMLMailer are not installed properly on the web server. If this error is generated by DAO or ADO try changing the WB_DBObject variable. If all values reports the same error then it is required to install Jet engineering on the web server.
If error was generated by HTMLMailer then it is required to install WhizBase on the web server.

23.2 Database errors

These errors are not specific to WhizBase but they are related to database usage. Their error numbers are between 3000 and 3999 and they can also have a negative error number. If you can not find the explanation in this manual, please search the Internet using following search term:
error *errornumber*

23.2.1 Error 3027 - Driver: Text; produced following error: Can't update. Database or object is read-only.

Text ISAM (driver) does not support deleting and updating records.
However, it supports adding and reading records so it can be used for all WhizBase

operations that do not require deleting or updating records.
If these features are required use other database type.

23.2.2 Error 3051 - The Microsoft Jet database engine cannot open the file '<file name>'. It is already opened exclusively by another user, or you need permission to view its data.

This error is generated when WBSP engine tries to open MS Access database and Jet engine is not able to complete operation.
Possible causes:
- Your permissions in system.md? file are not valid for required operation. You must have permission to read data in the specified file in order to view its data. To change your permission assignments, see your system administrator or the table or query's creator.
- WhizBase does not have write access on the directory where database is located and was unable to create <file name>.LDB file. Assign WBSP engine write permissions over the directory where the database file is located.

23.2.3 Error 3061 - Too few parameters. Expected <number>

In some cases, this message is generated when unknown field names or expressions are interpreted as query parameters. Be sure to enclose field names containing spaces or punctuation with square brackets [ ].

23.2.4 Error 3146 - ODBC--call failed.

This error may occur when the ODBC data source is on a network drive and user is not connected to the network or network is unavailable to user for any other reason. Make sure the network is available, and then try the operation again.
If this does not help make sure that user has the necessary permissions to access the ODBC source and the network drive. Remember that WBSP scripts are executed by user "internet_user", "nobody", "guest", etc.

23.2.5 Error 3170 - Couldn't find installable ISAM.

This error is generated when WBSP engine is unable to locate required ISAM libraries.
If WBSP engine is installed properly (no errors reported during the test mode) then the error is probably caused by mistyped WB_Connect value (check the semicolon at the end of WB_Connect value).

23.2.6 Error 3265 - Item not found in this collection.

This error is generated when WhizBase tries to access database table or field that does not exist.
Possible causes:

Table:
- table does not exist - the WB_RcdSet contains mistyped table name. To solve this check the name(s) of the table(s) defined in WB_RcdSet.
- table name contains dash, underscore, space or other special character. To solve this enclose table name in square brackets.

Field:
- field does not exist in the recordset. Make sure you spelled the field name correctly.
- there is more than one field with this name in the recordset (e.g. using complex recordset generated from two tables having few fields with same name); to solve this problem use $WBF[tablename.fieldname] and WBF_tablename.fieldname instead of $WBF[fieldname] and WBF_fieldname.

23.2.7 Error 3633 - Driver: MS Access; produced following error: Can't load DLL: 'MSJET35.DLL'

This error is generated when web server does not have MS Jet 3.5 database support. To solve this please install all necessary files on the web server. For more info please contact us at setup@whizbase.com.

23.3 WhizBase specific errors

23.3.1 Error 5010 - Duplicate value in UID!

This error is generated when user tries to add a record with defined WB_UID variable, and record with same value in fields defined in WB_UID already exists. Change the value in field(s) included in WB_UID.

23.3.2 Error 5011 - Empty mailing list recordset!

This error is generated when user tries to execute Mail List command (WB_Command=L) and resulting recordset is empty.

23.3.3 Error 5012 - Error reading system file <file name>!

This error does not exist since WhizBase v. 4.000

23.3.4 Error 5013 - Illegal referring page!

This error is generated when someone try to execute WhizBase command that require referrer check either from server that is not included in referrer list (Referrer check section), or by typing the URL in "Address" field of browser or using window.open() JavaScript function!
If you want to enable executing command only from authorized web servers (domains) include server name in referrer line in file Referrer check section.

23.3.5 Error 5014 - Illegal unique identifier! Query returned more than one record!

This error is generated when user tries to update or delete a record and query created upon WB_UID field returns more than one record. The solution is to check again the WB_UID value and to create new combination of database fields for WB_UID that will uniquely identify every record.

23.3.6 Error 5015 - Illegal unique identifier! Query returned no records!

This error is generated when user tries to update or delete a record and query created upon WB_UID field returns no records. The record does not exist, try typing other value for WBF_Fields that identify the record.

23.3.7 Error 5016 - Illegal use of unregistered trial version of WhizBase engine!

This error does not exist since WhizBase v. 4.000

23.3.8 Error 5017 - Incorrect password!

This error is generated when user submits invalid value for WB_DBOldPass when executing Change Password command (WB_Command=CP).

23.3.9 Error 5018 - Invalid command string!

This error is generated when WBSP engine does not recognize WB_Command value.

23.3.10 Error 5019 - Invalid data passed to UrlDecode() function.

This error is generated when a browser parse form data containing the percentage (%) character to WBSP engine. E.g. if the browser mistakenly called WBSP file with the following
http://www.domain.com/somefile.wbsp?WB_BaseName=base.mdb&WB_Query=100% free&...
If you use form for sending data to WBSP file your browser and WBSP engine will take care of transferring the data.

23.3.11 Error 5020 - Invalid data passed to UrlEncode() function.

Reserved for future use. If you get this error message, please send us description of operation that caused this error!

23.3.12 Error 5021 - Invalid value for True/False field!

This error is generated when database field of type Boolean (True/False) gets invalid value in WBF_field. Valid settings are:
for True value:
-1
1
T
TRUE
ON
Y
YES

for False value:
0 (zero)
NO
N
OFF
F
FALSE
Any other value will generate this error message!!

23.3.13 Error 5022 - New password check failed!

This error is generated when user submits different values for WB_DBNewPass and WB_DBNPassCh when executing Change Password command (WB_Command=CP).

23.3.14 Error 5023 - No data received! Nothing to add to database!

This error is generated when user tries to add an empty record (by sending an Add command with empty value in all WBF_fields). At least one database field (WBF_fields) must have value set.

23.3.15 Error 5024 - Query string empty! Unable to identify record!

This error is generated user tries to update (edit) record without specifying the vale for every database field defined in WB_UID variable.
Executing update command is possible only if all fields defined in WB_UID have value in WBF_Fields.
E.g.
If
wb_uid=Fname,Lname
then WBF_Fname and WBF_Lname must not be empty.

23.3.16 Error 5025 - Record(s) can't be added; no insert permission on <recordset>!

This error does not exist since WhizBase v. 4.000

23.3.17 Error 5026 - Record(s) can't be deleted; no delete permission on <recordset>!

This error does not exist since WhizBase v. 4.000

23.3.18 Error 5027 - Record(s) can't be edited; no update permission on <recordset>!

This error does not exist since WhizBase v. 4.000

23.3.19 Error 5028 - Record(s) can't be read; no read permission on <recordset>!

This error does not exist since WhizBase v. 4.000

23.3.20 Error 5029 - Required form field 'WB_BaseName' missing!

Neither your form nor configuration file have specified value for variable WB_BaseName and you are trying to execute command that requires this value to be set.
Check your spelling, and make sure that you did not forget to put the field in form/config file.

23.3.21 Error 5030 - Required form field 'WB_Pass' missing!

Neither your form nor configuration file have specified value for variable WB_Pass. This value is required when WB_System variable is not empty.
Check your spelling, and make sure that you did not forget to put the field in form/config file.

23.3.22 Error 5031 - Required form field 'WB_RcdSet' missing!

Neither your form nor configuration file have specified value for variable WB_RcdSet and you are trying to execute command that requires this value to be set.
Check your spelling, and make sure that you did not forget to put the field in form/config file.

23.3.23 Error 5032 - Required form field 'WB_UID' missing! Unable to identify record!

This error is generated when user tries to update (edit) record with WB_UID field empty. WB_UID must have database field name (or comma-separated names of the database fields) that uniquely identify every single record. For more information please read the explanation for WB_UID variable (section FormFields).

23.3.24 Error 5033 - Required form field 'WB_Usr' missing!

Neither your form nor configuration file have specified value for variable WB_Usr. This value is required when WB_System variable is not empty.
Check your spelling, and make sure that you did not forget to put the field in form/config file.

23.3.25 Error 5034 - Required form field(s) missing!

This error is generated when user submits the form without entering the value for field (or fields) defined in WB_Required variable.

23.3.26 Error 5035 - Required WB_UID member field <field name> missing! Unable to identify record!

This error is generated when WB_UID is defined and user tries to add, update or delete record but does not provide a value for all fields defined in WB_UID.
When WB_UID is defined none of the form fields (WBF_Fields) used in WB_UID variable cannot be left empty when executing add, update or delete command.

23.3.27 Error 5036 - Syntax error in $WBFN: <function>!

This error is generated when report function $WBFN receives unrecognized argument(s).

23.3.28 Error 5037 - Test mode disabled!

This error does not exist since WhizBase v. 4.000

23.3.29 Error 5038 - Unable to execute mail operation! WB_MailServer missing!

This error is generated when WhizBase does not get valid WB_MailServer variable and email-related operation is required.

23.3.30 Error 5039 - Unable to select mail list mode. Received values for both WB_ToField and WB_BCCField. Please remove one!

This error is generated when user tries to execute Mail List command (WB_Command=L) with values set both for WB_ToField and WB_BCCField. When executing L command WhizBase checks these fields to determine what kind of list to generate - if both fields have value WhizBase cannot decide what to do (there is no default method).

23.3.31 Error 5040 - User not found!

This error does not exist since WhizBase v. 4.000

23.3.32 Error 5041 - Unable to send mail to mailing list. Both WB_ToField and WB_BCCField are empty!

This error is generated when user tries to execute Mail List command (WB_Command=L) with no values for both WB_ToField and WB_BCCField variables.

23.3.33 Error 5042 - Report template file <TemplateName> not found!

This error is generated when WBSP engine is not able to find external template file defined in variable WB_TempName.

23.3.34 Error 5043 - Required field WB_Config missing!

This error does not exist since WhizBase v. 4.000

23.3.35 Error 5044 - Configuration file <ConfigName> not found!

This error does not exist since WhizBase v. 4.000

23.3.36 Error 5045 - Absolute path not allowed! File name: <FileName>

This error is generated when author tries to use absolute path on virtual server that does not have absolute paths enabled in file wbsp.ssc.

23.3.37 Error 5046 - Error in script file <ScriptFile>

This error is generated by scripting host when script specified in $WBRun function contains the error. WBSP provides detailed error report and processed script code saved in temporary file.

23.3.38 Error 5047 - Multipart content not allowed!

This error is generated when multipart form sends the data to WBSP file that does not have WB_AllowMultipart variable set to True.

23.3.39 Error 5048 - Server <ServerName> does not support file upload!

This error is generated when author tries to use file upload on virtual server that does not have file upload enabled in file wbsp.ssc.

23.3.40 Error 5049 - File too large!

This error is generated when visitor tries to upload file that is larger than value specified in WB_MaxFSize variable.

23.3.41 Error 5050 - Ilegal file type!

This error is generated when visitor tries to upload file with file extension specified in WB_Disallow variable.

23.3.42 Error 5051 - Ilegal scripting language!

This error is generated when author tries to use $WBRun function on virtual server that does not have file execute enabled in file wbsp.ssc, or using scripting language that is not included in variable Execute in wbsp.ssc.

23.3.43 Error 5052 - Ilegal script method!

This error does not exist since WhizBase v. 4.000

23.3.44 Error 5053 - Server <server name> does not support WBSP!

This error is generated when author tries to use WBSP files on virtual server that does not allow execution of WBSP files - variable DisableWB set to True in wbsp.ssc.

23.3.45 Error 5054 - WB_Order does not start with WB_ChangeHFOn

This error is generated when both variables exist and fields specified in WB_ChangeHFOn are not the same as starting fields in WB_Order.

23.3.46 Error 5055 - This WBSP project is not registered for use with <HostName> virtual host!

This error is generated when using protected WBSP file on non-authorized domain.

23.3.47 Error 5056 - $WBMREPL arguments do not match!

This error is generated when $WBMREPL function receives different number of comma-separated elements in source and target array.

23.3.48 Error 5057 - Required form field WB_FileName missing!

This error is generated when WBSP tries to execute command DF or WF and neither form nor configuration file have specified value for variable WB_FileName.

23.3.49 Error 5058 - Required form field WB_KeyName missing!

This error is generated when WBSP tries to execute command WF and neither form nor configuration file have specified value for variable WB_KeyName.

23.3.50 Error 5059 - Required form field WB_Section missing!

This error is generated when WBSP tries to execute command WF and neither form nor configuration file have specified value for variable WB_Section.

23.3.51 Error 5060 - Unable to delete file <filename>! File not found

This error is generated when WBSP tries to execute command DF and specified file does not exist.

23.3.52 Error 5061 - Error writing file <filename>

This error is generated when WBSP tries to execute command WF and fails for any reason.

23.3.53 Error 5062 - Server does not support writing files!

This error is generated when author tries to write file on virtual server that does not have file commands enabled in file wbsp.ssc.

23.3.54 Error 5063 - Server does not support deleting files!

This error is generated when author tries to delete file on virtual server that does not have file commands enabled in file wbsp.ssc.

23.3.55 Error 5064 - Wrong IP range string:<IPAddrRange>

This error is generated when file defined in WB_IPListFile contians invalid value for IP range. Valid formats are:
A.B.C.D
A.B.*.*
A.B.C.D - B.D.D.D
A.B.C.D/N

It is invalid to combine wildcard (*) and LowAddr-HiAddr range like this: A.B.*.* - C.D.D.D

23.3.56 Error 5065 - IP authentication failed!

This error is generated when user with invalid IP address (REMOTE_ADDR) tries to access WBSP file protected with IP address verification system.

23.3.57 Error 5066 - WB_UID for multi update must not contain field list! Unable to identify records!

This error is generated when variable WB_UID contains more than one field, and WB_Command is set to MU (Multi update).

23.3.58 Error 5067 - Different size of WBF form field arrays! Field <field name>.

This error is generated when WhizBase receives an arrays of form fields used for addition or updating of multiple records in a single pass, and when one (or more) of the WBF_ form field arrays have different number of elements.

23.3.59 Error 5068 - Invalid IP address <IP Address>! Dotted decimal values must be between 0 and 255

This error is generated when the IP list file contain an IP address with invalid decimal value (less then 0 or greater than 255).

23.3.60 Error 5069 - WB_KeyName and WB_KeyValue arrays do not match!

This error is generated when the user tries to write more than one key values using write file command (WB_Command=WF) and separator character (WB_Separator) and WB_KeyValue and WB_KeyName variables contain different number of elements separated by character defined in WB_Separator.

23.3.61 Error 5070 - Invalid value for the WB_DBObject!

This error is generated when variable ADOConnectionString is set to false and the value forwarded to WB_DBObject is invalid. WB_DBObject can contain one of the following values: d35, d36, A35, A40 or A07.

23.3.62 Error 5071 - Invalid value for the ADO connection string or WB_DBObject!

This error is generated when WB_DBObject receives unrecognized argument(s). The cause of this error can also be a syntax error in ADO Connection String.

23.3.63 Error 5072 - Security string missing

This error is generated when the user tries to start WBSP web page that does not start with the security string defined in CGISecurityString variable.

23.3.64 Error 5073 - Invalid extension

This error is generated when the user tries to execute WBSP in CGI mode with file extension different than the one defined in ActivateCGIByExt variable.

23.3.65 Error 5074 - Invalid numeric argument for $WBErr function! Use numbers from 6000 to 65535!

This error is generated when WBSP tries to process function $WBERR with errnum argument outside of 6000-65535 range.

23.3.66 Error 5075 - Syntax error

This error is generated when WBSP tries to process nested function with syntax error (usually illegal use of # character).

23.3.67 Error 5076 - WB_Redirect required!

This error is generated when WBSP file that uses WB_Destination variable does not have WB_Redirect variable defined.

23.3.68 Error 5077 - Too many instances for server <server name>!

This error is generated when number of instances of WBSP.EXE started by single virtual server is greater than number defined in server configuration variable MaxInstances.

23.3.69 Error 5078 - $WBCASE function syntax error

This error is generated when:
1. $WBCASE function does not have all five arguments
or
2. arguments "condition list" and "result list" do not have same number of items.

23.3.70 Error 5079 - Script time out error

This error is generated when single WBSP page takes more time to execute than it is defined in WB_TimeOut variable.

23.3.71 Error 5080 - Invalid time out interval

This error is generated when WB_TimeOut variable receives value less than 0 or greater than 86400.

23.3.72 Error 5081 - Invalid assign method! Use WBAAdd[] function.

This error is generated when you try to set value of an array element using $wbsetv instead of $wbaadd function.

23.3.73 Error 5082 - Invalid path!

This error is generated when WhizBase receives an invalid path to any file in any variable or function.

23.3.74 Error 5083 - Empty SMS list recordset!

This error is generated when user tries to execute SMS List command (WB_Command=SMS with WB_SMSField) and resulting recordset is empty.

23.3.75 Error 5084 - Required form field 'WB_SMSPort' missing!

This error is generated when user tries to execute WB_Command SMS and WB_SMSPort variable is empty. For more information please read the explanation for WB_SMSPort variable (section FormFields).

23.3.76 Error 5085 - Invalid character passed to WBCAPTCHA function!

This error is generated when WhizBase function $WBCAPTCHA receives an invalid character in *SourceText* parameter. Valid characters are 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ.

23.3.77 Error 5086 - Duplicate Sub Definition! Sub name: *nameofsubroutine*!

This error is generated when WhizBase finds subroutine definition with specified name that is already used for some other subroutine. The scope of subroutines is main wbsp file and all sub reports and include files called from main wbsp file.

23.3.78 Error 5087 - Undefined subroutine *nameofsubroutine*!

This error is generated when the subroutine called from WhizBase function $WBSUB does not exist. Check if you have spelled the subroutine name correctly, and if the subroutine is defined in main wbsp file or in any sub reports and include files called from main wbsp file.

23.3.79 Error 5088 - Invalid hash algorithm!

This error is generated when WhizBase function $WBHASH receives an invalid algorithm type in *AlgType* parameter. Valid types are MD5, SHA1, SHA256, SHA384, SHA512.

23.3.80 Error 5089 - Invalid algorithm!

This error is generated when WhizBase functions $WBENCRYPT OR $WBDECRYPT receive an invalid algorithm type in *AlgType* parameter. Valid types are AES, AES192, AES256, BF (Blowfish), CAST, DES, RC2, RC4, RC5, 3DES (TripleDES), UC (UNIXcrypt).

23.3.81 Error 5090 - Unrecognized encrypted format! Check the input type!

This error is generated when WhizBase function $WBDECRYPT receives input data that does not match *InputType* parameter (e.g. passing hexadecimal input data with InputType parameter set to B).

23.3.82 Error 5091 - Error processing JSON

This error is generated when WhizBase tries to process JSON code using any of these functions: $WBJSON, $WBJSONLEN or $WBJSONELEM and JSON parser returns an error. WhizBase will also display JSON source code and original error description.

23.3.83 Error 5092 - Missing parameter function name in $WBIRUN function!

This means that function $WBIRUN is missing second parameter, the name of function that should be executed.

24. Appendixes

24.1 Named Formats (format string definitions)

Format Name	Description
General Number	Display number with no thousand separator.
Currency	Display number with thousand separator, if appropriate; display two digits to the right of the decimal separator. Output is based on system locale settings.
Fixed	Display at least one digit to the left and two digits to the right of the decimal separator.
Standard	Display number with thousand separator, at least one digit to the left and two digits to the right of the decimal separator.
Percent	Display number multiplied by 100 with a percent sign (%) appended to the right; always display two digits to the right of the decimal separator.
Scientific	Use standard scientific notation.
Yes/No	Display No if number is 0; otherwise, display Yes.
True/False	Display False if number is 0; otherwise, display True.
On/Off	Display Off if number is 0; otherwise, display On.
General Date	Display a date and/or time. For real numbers, display a date and time, for example, 4/3/93 05:34 PM. If there is no fractional part, display only a date, for example, 4/3/93. If there is no integer part, display time only, for example, 05:34 PM. Date display is determined by your system settings.
Long Date	Display a date according to your system's long date format.
Medium Date	Display a date using the medium date format appropriate for the language version of the host application.
Short Date	Display a date using your system's short date format.
Long Time	Display a time using your system's long time format; includes hours, minutes, seconds.
Medium Time	Display time in 12-hour format using hours and minutes and the AM/PM designator.
Short Time	Display a time using the 24-hour format, for example, 17:45.

24.2 User-Defined Formats (format string definitions)

Character	Description
@	Character placeholder. Display a character or a space. If the string has a character in the position where the at symbol (@) appears in the format string, display it; otherwise, display a space in that position. Placeholders are filled from right to left unless there is an exclamation point character (!) in the format string.
&	Character placeholder. Display a character or nothing. If the string has a character in the position where the ampersand (&) appears, display it; otherwise, display nothing. Placeholders are filled from right to left unless there is an exclamation point character (!) in the format string.
<	Force lowercase. Display all characters in lowercase format.
>	Force uppercase. Display all characters in uppercase format.
!	Force left to right fill of placeholders. The default is to fill placeholders from right to left.
0	Digit placeholder. Display a digit or a zero. If the expression has a digit in the position where the 0 appears in the format string, display it; otherwise, display a zero in that position. If the number has fewer digits than there are zeros (on either side of the decimal) in the format expression, display leading or trailing zeros. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, round the number to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, display the extra digits without modification.
#	Digit placeholder. Display a digit or nothing. If the expression has a digit in the position where the # appears in the format string, display it; otherwise, display nothing in that position. This symbol works like the 0 digit placeholder, except that leading and trailing zeros are not displayed if the number has the same or fewer digits than there are # characters on either side of the decimal separator in the format expression.
.	Decimal placeholder. In some locales, a comma is used as the decimal separator. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only number signs to the left of this symbol, numbers smaller than 1 begin with a decimal separator. To display a leading zero displayed with fractional numbers, use 0 as the first digit placeholder to the left of the decimal separator. The actual character used as a decimal placeholder in the formatted output depends on the Number Format recognized by your system.
%	Percentage placeholder. The expression is multiplied by 100. The percent character (%) is inserted in the position where it appears in the format string.
,	Thousand separator. In some locales, a period is used as a thousand separator. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is

	specified if the format contains a thousand separator surrounded by digit placeholders (0 or #). Two adjacent thousand separators or a thousand separator immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing it by 1000, rounding as needed." For example, you can use the format string "##0,," to represent 100 million as 100. Numbers smaller than 1 million are displayed as 0. Two adjacent thousand separators in any position other than immediately to the left of the decimal separator are treated simply as specifying the use of a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format recognized by your system.
E- E+ e- e+	Scientific format. If the format expression contains at least one digit placeholder (0 or #) to the right of E-, E+, e-, or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents.
- + $ ( )	Display a literal character. To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (" ").
\	Display the next character in the format string. To display a character that has special meaning as a literal character, precede it with a backslash (\). The backslash itself is not displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\). Examples of characters that cannot be displayed as literal characters are the date-formatting and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, / and :), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !).
"ABC"	Display the string inside the double quotation marks (" "). To include a string in format from within code, you must use Chr(34) to enclose the text (34 is the character code for a quotation mark (")).
:	Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings.
/	Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings.
c	Display the date as ddddd and display the time as ttttt, in that order. Display only date information if there is no fractional part to the date serial number; display only time information if there is no integer portion.
d	Display the day as a number without a leading zero (1 – 31).

dd	Display the day as a number with a leading zero (01 – 31).
ddd	Display the day as an abbreviation (Sun – Sat).
dddd	Display the day as a full name (Sunday – Saturday).
ddddd	Display the date as a complete date (including day, month, and year), formatted according to your system's short date format setting. For Microsoft Windows, the default short date format is m/d/yy.
dddddd	Display a date serial number as a complete date (including day, month, and year) formatted according to the long date setting recognized by your system. For Microsoft Windows, the default long date format is mmmm dd, yyyy.
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday).
ww	Display the week of the year as a number (1 – 54).
m	Display the month as a number without a leading zero (1 – 12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01 – 12). If m immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan – Dec).
mmmm	Display the month as a full month name (January – December).
q	Display the quarter of the year as a number (1 – 4).
y	Display the day of the year as a number (1 – 366).
yy	Display the year as a 2-digit number (00 – 99).
yyyy	Display the year as a 4-digit number (100 – 9999).
h	Display the hour as a number without leading zeros (0 – 23).
hh	Display the hour as a number with leading zeros (00 – 23).
n	Display the minute as a number without leading zeros (0 – 59).
nn	Display the minute as a number with leading zeros (00 – 59).
s	Display the second as a number without leading zeros (0 – 59).
ss	Display the second as a number with leading zeros (00 – 59).
t t t t t	Display a time as a complete time (including hour, minute, and second), formatted using the time separator defined by the time format recognized by your system. A leading zero is displayed if the leading zero option is selected and the time is before 10:00 A.M. or P.M.
AM/PM	Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 P.M.
am/pm	Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 P.M.
A/P	Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 P.M.
a/p	Use the 12-hour clock and display a lowercase A with any hour

	before noon; display a lowercase P with any hour between noon and 11:59 P.M.
AMPM	Use the 12-hour clock and display the AM string literal as defined by your system with any hour before noon; display the PM string literal as defined by your system with any hour between noon and 11:59 P.M. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. For Microsoft Windows, the default format is AM/PM.

24.3 Format expression rules

A format expression for strings can have one section or two sections separated by a semicolon (;).

If you use	**The result is**
One section only	The format applies to all string data.
Two sections	The first section applies to string data, the second to Null values and zero-length strings ("").

A user-defined format expression for numbers can have from one to four sections separated by semicolons. If the format argument contains one of the named numeric formats, only one section is allowed.

If you use	**The result is**
One section only	The format expression applies to all values.
Two sections	The first section applies to positive values and zeros, the second to negative values.
Three sections	The first section applies to positive values, the second to negative values, and the third to zeros.
Four sections	The first section applies to positive values, the second to negative values, the third to zeros, and the fourth to Null values.

24.4 HTML form and input elements

24.4.1 HTML from

The Form element is used to delimit a data input form. There can be several forms in a single document, but the Form element cannot be nested. (i.e. a form cannot contain another form)

<FORM ACTION="_URL_" METHOD="GET|POST" ENCTYPE="MIME type">
The ACTION attribute is a URL specifying the location to which the contents of the form data fields are submitted to elicit a response. As mentioned before, this could be simply a direction to an e-mail address, but generally, would be used to point towards some kind of server based CGI script/application that handles the forwarding of form data. If the ACTION attribute is missing, the URL of the document itself is assumed. The way data is submitted varies with the access protocol of the URL to

which the form data is sent and with the values of the METHOD and ENCTYPE attributes.

Generally, the METHOD attribute specifies a method of accessing the URL specified in the ACTION attribute and will be either GET or POST. The GET method is ideal for form submission where the use of the form data does not require external processing. For example, with database searches, there is no lasting effect caused by the query of the form (that is, the query runs its search through the database and reports the results). However, where the form is used to provide information for example, that updates a database, then the POST method should be used, with the ACTION attribute pointing to a CGI script that executes the form data processing.

The ENCTYPE specifies the media type used to encode the form data. The default ENCTYPE is the MIME type 'application/x-www-form-urlencoded'

The <FORM> element can also accept the TARGET attribute (as in <A> elements), to specify what window is used for any form feedback. It can take the following values :

window_name
The name of any window specified by a <FRAME> element, or by using the window.open scripting method. If a window_name is used which does not correlate to a previously defined window, then a new window is created and NAMED according the the window name used in the TARGET attribute. This new window can then be referenced using its new name.
_self
Using this reserved keyword value, would cause any form feedback page to be loaded into the window that currently contains the form.
_parent
Using this reserved keyword value, would cause any form feedback page to be loaded into the window that is the parent of the window currently containing the form. i.e. if the form's window is part of a framed document, it would load into the window controlled by the <FRAMESET> element definitions that control the form's current window.
_top
Using the reserved keyword value would cause the form feedback page to be loaded into the topmost window, clearing any currently existing framed windows.
_blank
Using this reserved keyword value would cause the form feedback page to be loaded into a newly created window. Using this value is the same as using TARGET="window_name" where the window_name used is not a previously defined window. NOTE: Unlike using the window_name using a previously undefined window name, using _blank will not name the new window for future use.
The <FORM> can also take the NAME attribute, which can be used to set the name of the element for scripting purposes.

24.4.2 Input elements

The Input element represents a field whose contents may be edited or activated by the user.

Attributes of the Input element:

ALIGN
Vertical alignment of the image. For use only with TYPE=IMAGE in HTML level 2. The possible values are exactly the same as for the ALIGN attribute of the <IMG ...> element.

CHECKED
Indicates that a checkbox or radio button is selected. Unselected checkboxes and radio buttons do not return name/value pairs when the form is submitted.

MAXLENGTH
Indicates the maximum number of characters that can be entered into a text field. This can be greater than specified by the SIZE attribute, in which case the field will scroll appropriately. The default number of characters is unlimited.

NAME
Symbolic name used when transferring the form's contents. The NAME attribute is required for most input types and is normally used to provide a unique identifier for a field, or for a logically related group of fields. The name given to the element can also be used to reference it for scripting purposes.

SIZE
Specifies the size or precision of the field according to its type. For example, to specify a field with a visible width of 24 characters:
INPUT TYPE="text" SIZE="24"

SRC
To be used with the TYPE=IMAGE , this attribute represents a URL specifying an the desired image.

TYPE
Defines the type of data the field accepts. Defaults to free text. Several types of fields can be defined with the type attribute:

BUTTON: This can be used to embed buttons directly into HTML documents, that add functionality when used in conjunction with Visual Basic Script, or JavaScript. The NAME attribute is used to give the button a unique name, which can be used to set its function in the script. The VALUE attribute specifies the text that is displayed on the button in the document.

CHECKBOX: Used for simple Boolean attributes (where a field will be chosen, or not), or for attributes that can take multiple values at the same time. The latter is represented by a number of checkbox fields each of which has the same name. Each selected checkbox generates a separate name/value pair in the submitted data, even if this results in duplicate names. The default value for checkboxes is "on". It requires the NAME and VALUE attributes, optional attributes being CHECKED.

HIDDEN: With this input type, no field is presented to the user, but the content of the field is sent with the submitted form. This value may be used to transmit state information about client/server interaction.

IMAGE: An image field upon which you can click with a pointing device, causing the form to be immediately submitted. The co-ordinates of the selected point are measured in pixel units from the upper-left corner of the image, and are returned (along with the other contents of the form) in two name/value pairs. The x-co-ordinate is submitted under the name of the field with .x appended, and the y- co-ordinate is submitted under the name of the field with .y appended. The NAME attribute is required. The image itself is specified by the SRC attribute, exactly as for the Image element.

PASSWORD: is the same as the TEXT attribute, except that text is not displayed as it is entered.

RADIO: is used for attributes that accept a single value from a set of alternatives. Each radio button field in the group should be given the same name. Only the selected radio button in the group generates a name/value pair in the submitted data. Radio buttons require an explicit VALUE and NAME attribute. CHECKED is an optional attribute and can be used to specify which options are selected for initial form display.

RESET: is a button that when pressed resets the form's fields to their specified initial values. The label to be displayed on the button may be specified just as for the SUBMIT button.

SUBMIT: is a button that when pressed submits the form. You can use the VALUE attribute to provide a non- editable label to be displayed on the button. The default label is browser-specific. If a SUBMIT button is pressed in order to submit the form, and that button has a NAME attribute specified, then that button contributes a name/value pair to the submitted data. Otherwise, a SUBMIT button makes no contribution to the submitted data.

TEXT: is used for a single line text entry fields. It should be used in conjunction with the SIZE and MAXLENGTH attributes to set the maximum amount of text that can be entered. For textual input that requires multiple lines, the <TEXTAREA> element for text fields which can accept multiple lines. Explicit VALUE and NAME attributes are also required.

TEXTAREA: is used for multiple-line text-entry fields. Use in conjunction with the SIZE and MAXLENGTH attributes. It is better to use the <TEXTAREA> element for such text entry boxes.

FILE: is used with forms of ENCTYPE "multipart/form-data".
This allows the inclusion of files with form information, which could prove invaluable for example, for companies providing technical support, or service providers, requesting data files.

VALUE
When used with TYPE= ... attributes, this attribute sets the initial displayed value of the field if it displays a textual or numerical value. If the TYPE= ... attribute is one which only allows Boolean values (i.e. chosen, or not chosen) then this specifies the value to be returned when the field is selected.

24.5 SQL patterns

Character(s) in pattern		**Matches in expression**
DAO	***ADO***	
? (question mark)	_ (underscore)	Any single character
* (asterisk)	% (percent sign)	Zero or more characters
#		Any single digit (0-9)
[charlist]		Any single character in charlist
[!charlist]		Any single character not in charlist

A group of one or more characters (charlist) enclosed in brackets ([ ]) can be used to match any single character in expression and can include almost any characters in the ANSI character set, including digits. In fact, the special characters left bracket ([), question mark (?), underscore(_), number sign (#), percent sign (%) and asterisk (*) can be used to match themselves directly only by enclosing them in brackets. The right bracket (]) cannot be used within a group to match itself, but it can be used outside a group as an individual character.
In addition to a simple list of characters enclosed in brackets, charlist can specify a range of characters by using a hyphen (-) to separate the upper and lower bounds of the range. For example, [A-Z] in pattern results in a match if the corresponding character position in expression contains any of the uppercase letters in the range A through Z. Multiple ranges are included within the brackets without any delimiting. For example, [a-zA-Z0-9] matches any alphanumeric character.
Other important rules for pattern matching include the following:
An exclamation mark (!) at the beginning of charlist means that a match is made if any character except the ones in charlist are found in expression. When used outside brackets, the exclamation point matches itself.
The hyphen (-) can appear either at the beginning (after an exclamation mark if one is used) or at the end of charlist to match itself. In any other location, the hyphen is used to identify a range of ANSI characters.
When a range of characters is specified, they must appear in ascending sort order (from lowest to highest). [A-Z] is a valid pattern, but [Z-A] is not.
The character sequence [ ] is ignored; it is considered to be a zero-length string. The following examples show how you can use Like to test expressions for different patterns.

Kind of matching	**With this pattern**	**This expression returns True**	**This expression returns False**
Multiple characters	a*a	"aa", "aBa", "aBBBa"	"aBC"
Special character	a[*]a	"a*a"	"aaa"
Multiple characters	ab*	"abcdefg", "abc"	"cab", "aab"
Single character	a?a	"aaa", "a3a", "aBa"	"aBBBa"
Single digit	a#a	"a0a", "a1a", "a2a"	"aaa", "a10a"
Range of characters	[a-z]	"f", "p", "j"	"2", "&"
Outside a range	[!a-z]	"9", "&", "%"	"b", "a"
Not a digit	[!0-9]	"A", "a", "&", "~"	"0", "1", "9"
Combined	a[!b-m]#	"An9", "az0", "a99"	"abc", "aj0"
Single special to double	[?]	"ae", "AE", "?"	"Ä", "A"