MiniM Web Access

Eugene Karataev
mailto:support@minimdb.com
http://www.minimdb.com

February 6, 2014

Contents

Chapter 1

Setup and Administration

MiniM Web Access is CGI module for web server, page processor and special tag handling module to work with MiniM Database Server.

1.1 Common principles

HTTP client, for example web browser, send http query to web server. Web server determines handler for mwa page and call page processor. For MWA page administrator must setup MWA CGI processor. Through page handling MWA processor determines parameters to connect to MiniM Database Server, runs process on the server and writes to MUMPS local variables query parameters - HTTP CGI environment and page parameters.

After prepare local variables page processor scans mwa page. All special tags begins with <? and ends with ?> are processed as special MWA tags and handled depended of tag type. After open symbols <? need to be MWA special tag name. If this tag is not MWA tag, this tag is not processed. Page processor make need action to interoperate with MiniM process and all character sequence from (and including) <? to ?> are removes or replaces with need character sequence. All character sequence outside of special MWA tags does not processed and sends to web server without any changes.

MWA tags can have no parameters, can have mandatory parameters and can have optional (not mandatory) parameter.

In processing stage page processor creates output content, which sends to web server and web server send response to HTTP client. MWA page processor can change page content and HTTP response headers.

MWA page processor requires MiniM database "TEMP" still in non-journaling state.

1.2 Setup Apache for Windows

All need files to setup MiniM Web Access are in /mwa subdirectory of MiniM Database Server installation. MWA files are:

nph-minimwa.exe	CGI module and MWA page processor
mwa.ini	MWA applications settings template file
samples	Subdirectory with samples

To setup Apache web server to process MWA pages administrator must enable MWA page processor to execute. Administrator can use MWA page processor in MiniM installation directory or in separate directory or subdirectory. In the same subdirectory with MWA page processor nph-minimwa.exe need to be the setting's file mwa.ini. This file need to be edited to setup MWA applications and connection options. Recommended way in order to avoid changes is to place nph-minimwa.exe and mwa.ini files into separate subdirectory. Later we assime MWA page processor is installed into /Apache/cgi-bin subdirectory of Apache Web Server installation. And Apache setup procedure is:

Copy nph-minimwa.exe and mwa.ini files into /Apache/cgi-bin subdirectory.

In Apache /conf/ subdirectory edit file httpd.conf and setup file handler type:

```
AddType application/mwa .mwa
```

Add handler type association with MWA processor:

```
Action application/mwa "/cgi-bin/nph-minimwa.exe"
```

Another way to accosiate MWA processor with .mwa extension is declaring handler type:

```
AddHandler mwascript .mwa
Action mwascript /cgi-bin/nph-minimwa.exe
```

For Apache version 2 and later need to make additional setup steps.

Uncomment configuration line if one present or add if not:

```
LoadModule ext_filter_module modules/mod_ext_filter.so
```

Define file handler filter (cmd parameters need to be in one line):

```
ExtFilterDefine mwa mode=output \
  cmd="E:\Program Files\Apache Software
Foundation\Apache2.2\cgi-bin\nph-minimwa.exe" \
  intype=application/mwa
```

Enable defined output handler filter

```
SetOutputFilter mwa
```

Now MWA page handler is ready to work under Apache Web Server.

1.3 Setup Apache for Linux

All installation files to setup Apache for Linux are in mwa subdirectory of MiniM installation. MiniM for Linux contains MWA processor for Linux:

minimwa	CGI module and MWA page processor
mwa.ini	MWA applications settings template file
samples	Subdirectory with samples

To setup MWA handler in Apache for Linux need to determine directory of Apache configuration file httpd.conf / apache.conf / apache2.conf. In depending of Linux version / edition Apache can be installed in different directories, for example

Debian Linux:	/etc/apache2
Red Hat Linux:	/etc/httpd/conf

You need to decide what directory will be used by minimwa executable to run from. Here we assume that this module will be located in cgi-bin subdirectory. In depending of Linux version / edition this may be located in:

Red Hat Linux:	/var/www/cgi-bin
Debian Linux:	/usr/lib/cgi-bin

Executable module minimwa and configuration file mwa.ini must be copied into this subdirectory.

Executable module minimwa must have executable attribute:

```
chmod +x minimwa
```

Next must be checked in the file httpd.conf that webserver loads "actions" mosule. In the case of webserver does not do this, determine in the modules subdirectory loading file for this module. For example, in the Debian Linux in the subdirectory /etc/apache2/mods-available present file actions.load with commands for loading "actions" module. Webserver need to be configured to execute commands from this file. Need add to configuration file httpd.conf command

```
LoadModule actions_module /usr/lib/apache2/modules/mod_actions.so
```

as described in the file actions.load, or determine location of modules directory and describe module loading, or make link from subdirectory mods-enabled. All this actions are depends from the Linux and Apache version.

Next need to define handler type

```
AddHandler mwahandler .mwa
```

and setup handler

```
Action mwahandler /cgi-bin/minimwa
```

After all done, webserver must be restarted:

```
apachectl -k restart
```

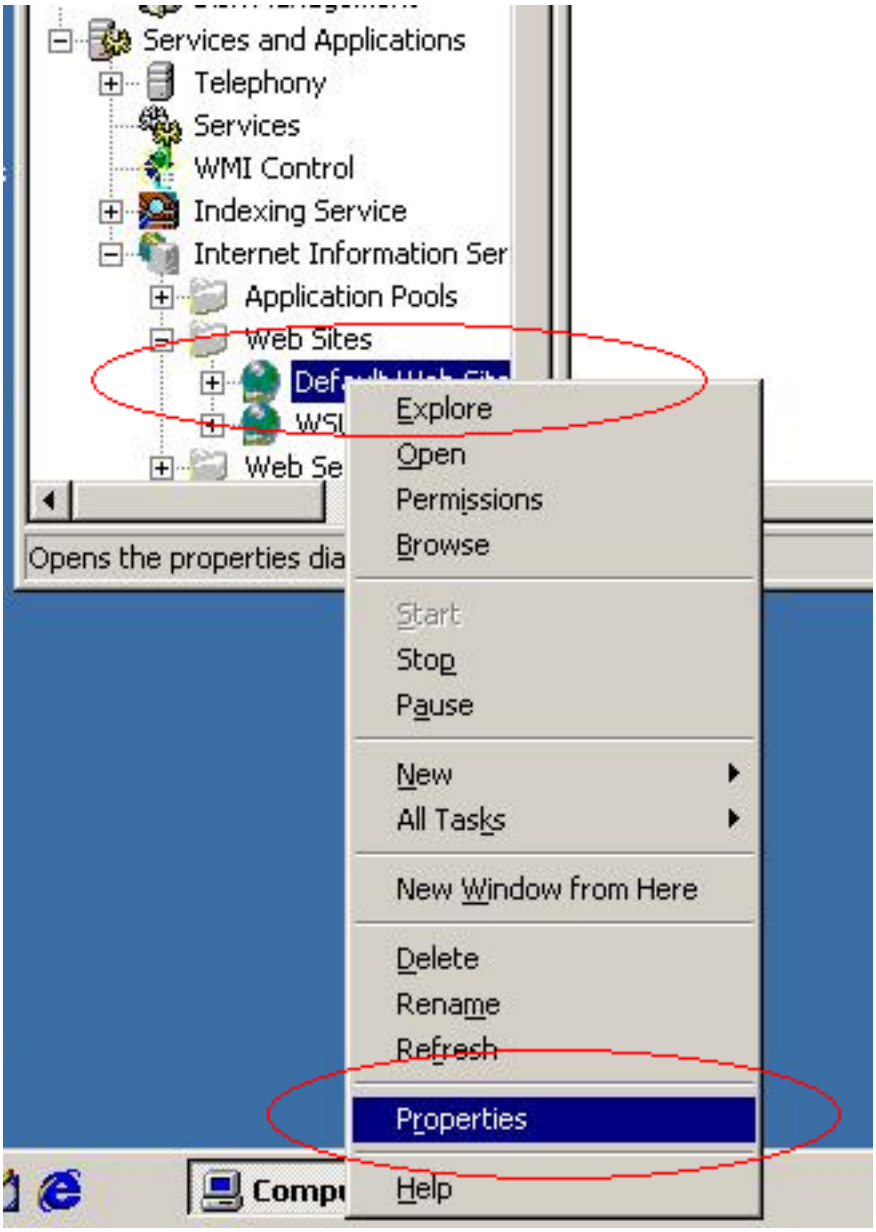
1.4 Setup IIS Web Server

Before IIS Web Server setup administrator must check web server is installed and add from Windows installation disk if need. To do this call applet "Install applications" and select "Add Windows components". Select setup Internet Information Services (IIS). Next follow installation instructions. This procedure may depends from Windows version. After installing IIS Web Server make setup MWA processor.

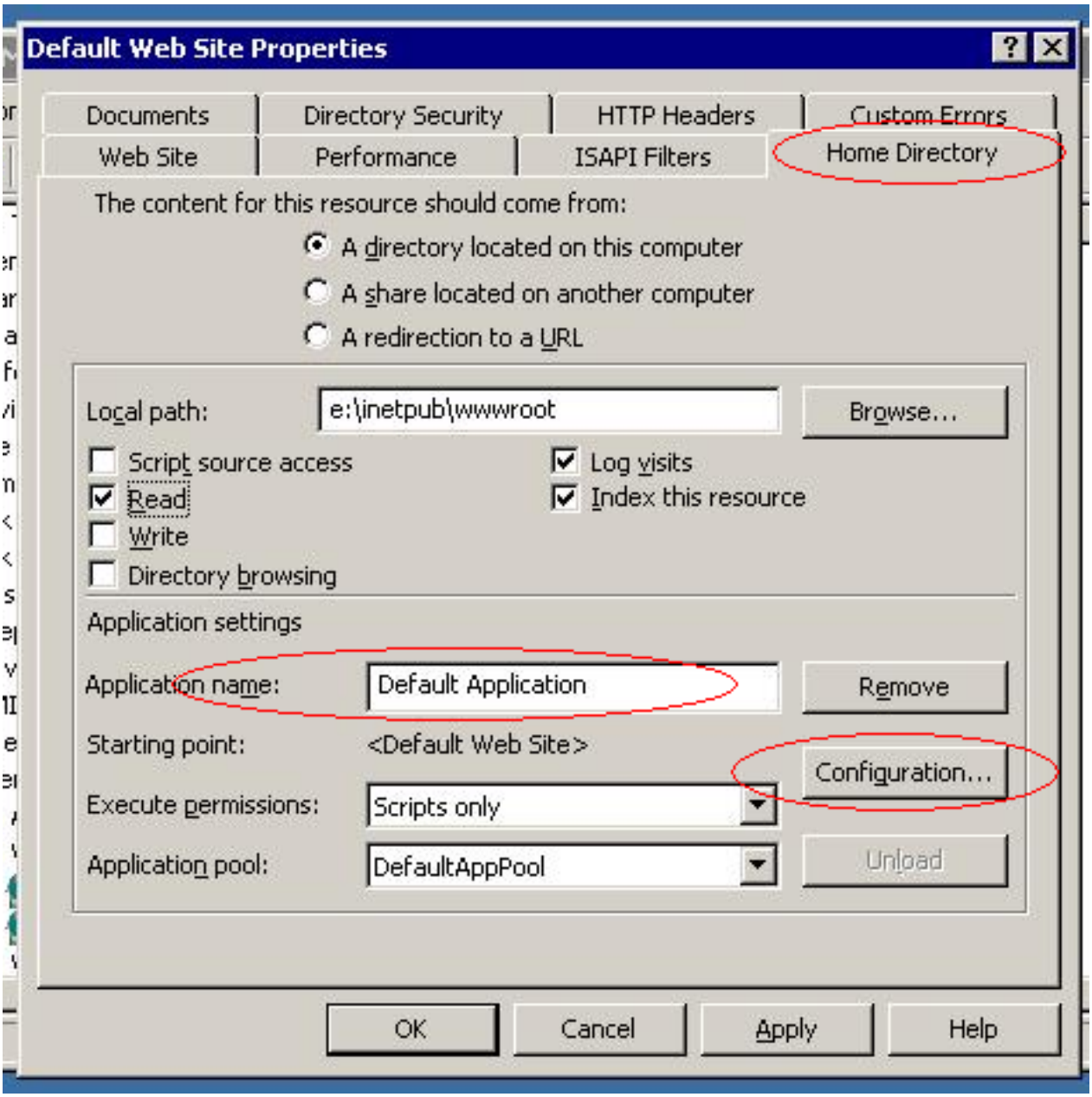
Copy into IIS Web Server /scripts subdirectory (by default it is system disk and InetPub subdirectory) nph-minimwa.exe and mwa.ini files from MiniM Database Server installation subdirectory /mwa.

Run from Control Panel applet "Computer Management".

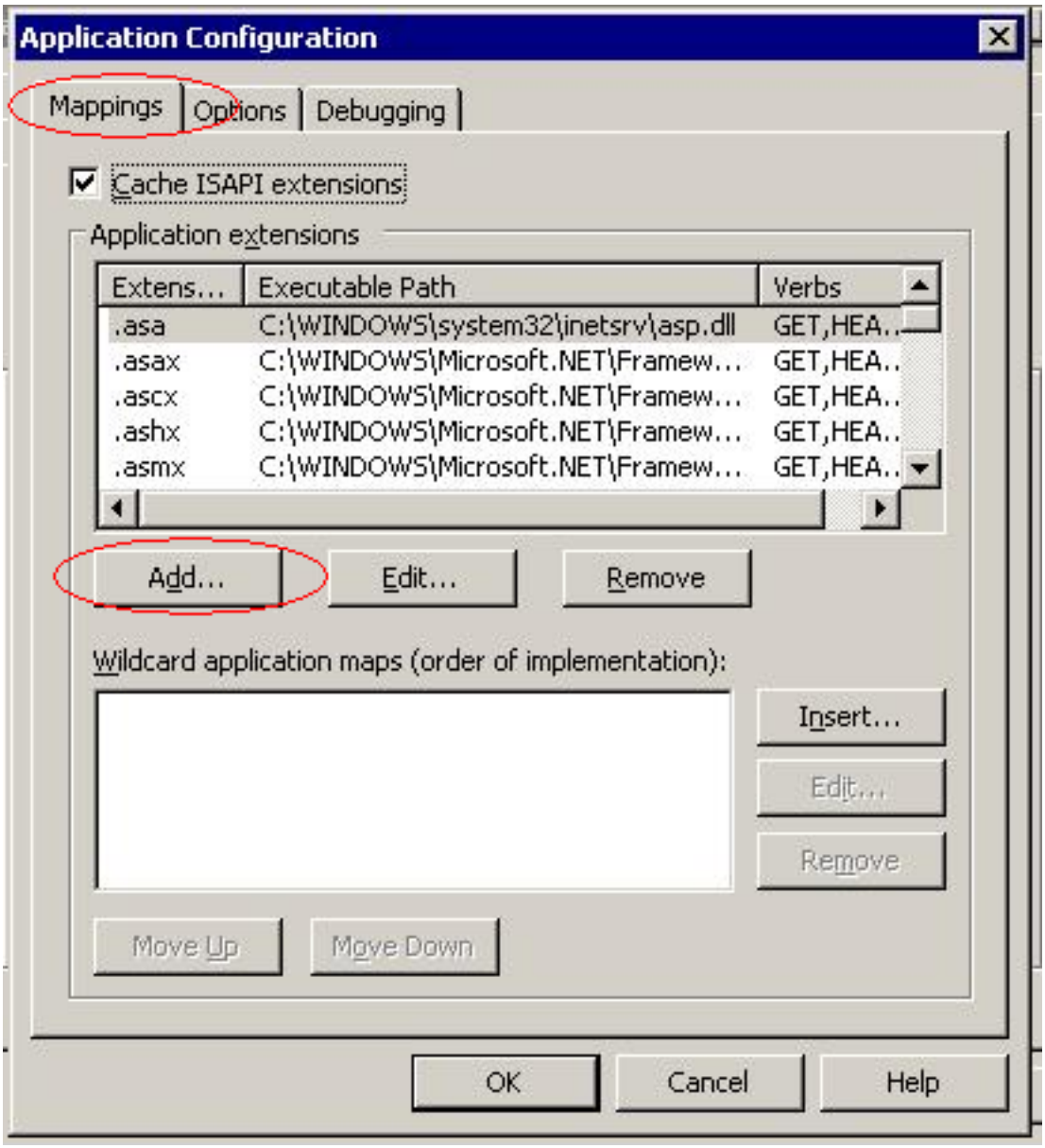
Inside "Services and applications" group select Internet Information Services. IIS Web Server can handle one web site (by default) or many web sites. Select need web site, click right mouse button and select from context menu "Properties" action:



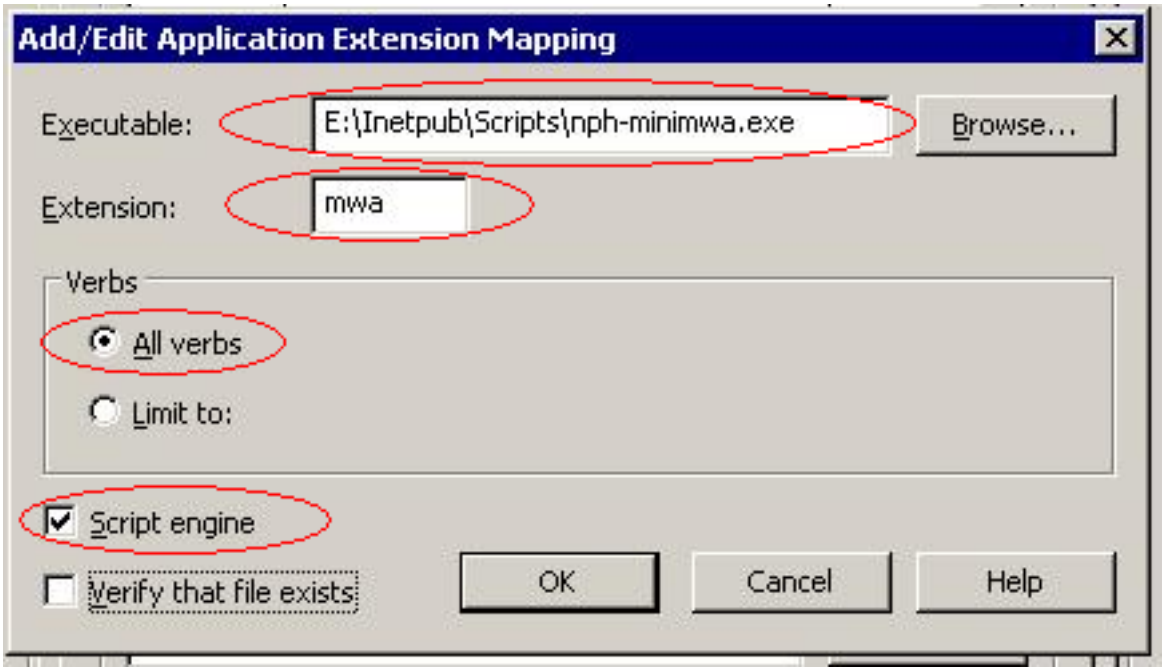
In the setup dialog select page "Home Directory" and check application "Default Application" is created. If one does not exist, this application need to be created.



Click button "Configuration" to create script's mapping rule. Using the Application Configuration dialog select Mapping page and press "Add" button.



Using the Application Extension Mapping dialog specify early copied file nph-minimwa.exe, specify "mwa" as file extension and check that handler must handle all query verbs and it is a script engine.



Now MWA page handler is ready to work under IIS Web Server.

1.5 MWA settings file mwa.ini

File mwa.ini is the MWA settings file. Here are described connection options and available virtual MWA applications.

Section [::], key ConnectLimit specify limit to number of simultaneous connections from MWA page processor to MiniM Database Server. Here assumed that all connections are handled inside this limit independently from available and used MiniM Database Servers. If web server need to handle more than specified in ConnectLimit queries concurrently, some part of ones are placed into internal quiery order until current quieries are handled and still wait available connect. MWA page processor does not use number of concurrent connections to MiniM Database Server more then ConnectLimit. How much connections required to handle by MWA page processor can be determined experimentally. This number can depends from number of CPU processors, for example.

Section [::], key ShowProcess specify show or not MiniM process's system variables on the MWA error page.

Section [::], parameter ShowCGI specify show or not HTTP query parameters on the MWA error page.

Both this settings can help to programmers to find out errors. While used in the production this additional info can be disabled. It is recommended.

MWA error page is produced by MWA page processor if any errors occured while page processing.

To setup options to connect to MiniM Database Server administrator must add all need connection descriptions. Section names are names of virtual Web Server subdirectory. MWA page processor use this options to connect to MiniM while handle pages from this directory and subdirectories, if special subdirectory options not specified. Root Web Server subdirectory is specified by section name with one "/" symbol.

For example section [/] defines connection options for MWA pages from root subdirectory, section [/lib/] defines special separate connection options for /lib/ virtual directory and /lib/ subdirectories. Connection may vary, for example, by MiniM database name to connect to.

Section [/dirname/], key Host specifies the network computer name with MiniM Database Server. To connect to local MiniM installation on the same computer can be specified name "localhost" without quotes.

Section [/dirname/], key Address specifies IP address of computer with MiniM Database Server. Administrator can specify one of Host or Address keys, by the administrator choice. Must be specified at least one key, Host or Address.

Section [/dirname/], key Port specifies IP port number which handled by internal MiniM superserver ^%srv. By default this is port 5000, and administrator can change this port.

Section [/dirname/], key Database specifies MiniM database name to connect to handle MWA page.

Sample mwa.ini file can be, for example:

```
[::]
ConnectLimit = 10
ShowProcess = 1
ShowCGI = 1

[/]
Host = localhost
Port = 5000
Database = USER
```

The Section [:CGI:] is designed for advanced CGI handling settings. If this section is present in mwa.ini file MWA page processor creates additional CGI environment variables. Lines in this section must specify names of this additional variables.

Additional CGI variables names must be listed one per line as ini key names. Symbol "=" is optional. All line after symbol "=" is ignored. For example, additional Apache Web Server CGI variables can be listed as:

```
[:CGI:]
HTTPS =
HTTP_TE =
INSTANCE_META_PATH =
SSL_CLIENT_I_DN =
SSL_SERVER_I_DN =
SSL_SERVER_S_DN =
```

By default MWA page processor handles only base CGI environment variables names. In depending on the applicable Web server and included in its capabilities, this list can be changed by the administrator for a full the necessary information.

1.6 Setup Apache Web Server Examples

To setup MiniM Web Access examples for Apache administrator must enable script execution for this subdirectory. Assume that the MiniM Database Server is installed to directory

```
C:\Program Files\MiniM\
```

and administrator must declare virtual subdirectory in Apache configuration file httpd.conf as

```
<Directory "C:/Program Files/MiniM/mwa/samples/">
  Options All
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
Alias /mwasamples/ "C:/Program Files/MiniM/mwa/samples/"
```

Or administrator can copy this directory into web server subdirectory and assign options *Options All*.

Once this settings are made MiniM Web Access examples are available as virtual directory /mwasamples/ and web browser can be pointed to

```
http://localhost/mwasamples/index.html
```

Administrator can specify separate connection options for this virtual subdirectory, for example

```
[/mwasamples/]
Host = localhost
Port = 5000
Database = USER
```

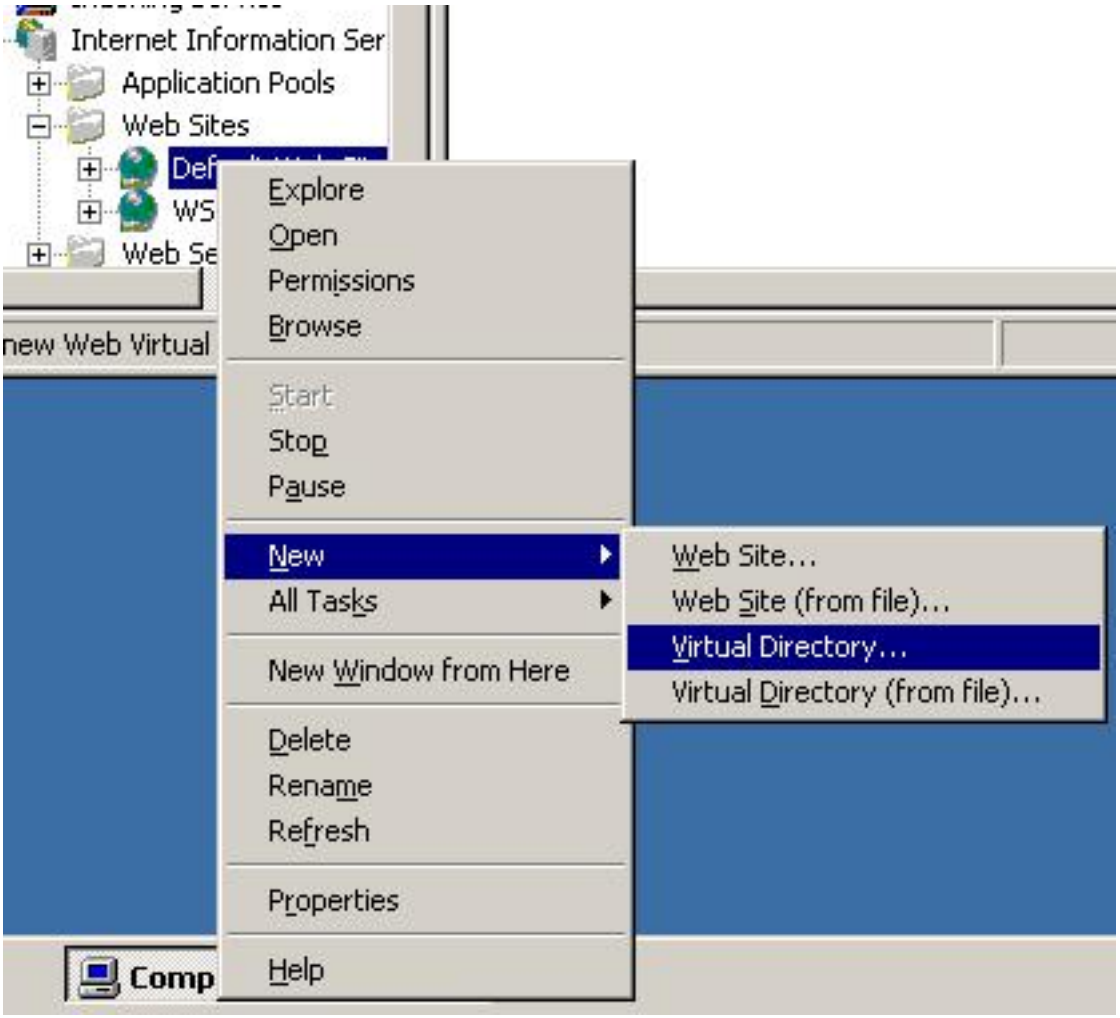
If administrator does not setup separate connection options MWA page processor use default connection options from root directory [/].

For MiniM Web Access examples does not required to import any routines or globals into database and examples does not modify database. While examples runs can be modified only TEMP database. MWA page processor use TEMP database to handle some MWA tags.

1.7 Setup IIS Web Server Examples

To setup examples for IIS Web server administrator must create for selected web site virtual subdirectory /mwasamples/ and as phisical directory specify subdirectory /mwa/samples from MiniM Database Server installation.

Administrator must run applet "Computer management" or other IIS administration tool dependent of IIS version, select appropriate web site and in context menu select action to create virtual subdirectory.



Using virtual subdirectory wizard must be specified alias of virtual subdirectory as *mwasamples* and as phisical path need to be selected /mwa/samples subdirectory from MiniM Database Server installation. Also need to be checked options to allow to read and execute scripts.

Check IIS Web Server is running. Once this settings are made MiniM Web Access examples are available as virtual directory /mwasamples/ and web browser can be pointed to

```
http://localhost/mwasamples/index.html
```

Administrator can specify separate connection options for this virtual subdirectory, for example

```
[/mwasamples/]
Host = localhost
Port = 5000
Database = USER
```

If administrator does not setup separate connection options MWA page processor use default connection options from root directory [/].

For MiniM Web Access examples does not required to import any routines or globals into database and examples does not modify database. While examples runs can be modified only TEMP database. MWA page processor use TEMP database to handle some MWA tags.

Chapter 2

Query parameters

Before to process page MWA page processor reads CGI environment variables and query parameters to processed mwa page and writes ones into MUMPS local variables. On execution mwa tags this parameters can be used as need, changed or deleted. CGI and query parameters are writes only once. After mwa page processed MiniM process halted and this variables are expired.

2.1 CGI environment variables

The values of CGI environment variables stored in local variable

```
%CGI
```

As an index are used CGI variable names and as values are user CGI variable values. For example:

```
%CGI("REQUEST_METHOD")="GET"
```

CGI environment variables contains information about currently processed MWA page, about web server about http client and other.

To list all available CGI environment variables programmer can insert into MWA page tag to write out local variable %CGI:

```
<?exec zw %CGI ?>
```

Here MWA page responce can be for example like this:

```
%CGI("AUTH_PASSWORD")=""
%CGI("AUTH_TYPE")=""
%CGI("HTTP_ACCEPT")="text/html, application/xml;q=0.9,
  application/xhtml+xml, image/png, image/jpeg,
  image/gif, image/x-xbitmap, */*;q=0.1"
%CGI("HTTP_ACCEPT_CHARSET")="iso-8859-1, utf-8,
  utf-16, *;q=0.1"
%CGI("HTTP_ACCEPT_ENCODING")="deflate, gzip, x-gzip,
  identity, *;q=0"
%CGI("HTTP_ACCEPT_LANGUAGE")="ru-RU,ru;q=0.9,en;q=0.8"
%CGI("HTTP_AUTHORIZATION")=""
%CGI("HTTP_CACHE_CONTROL")="no-cache"
%CGI("HTTP_CONNECTION")="Keep-Alive, TE"
%CGI("HTTP_CONTENT_ENCODING")=""
%CGI("HTTP_CONTENT_LENGTH")="-1"
%CGI("HTTP_CONTENT_TYPE")=""
%CGI("HTTP_CONTENT_VERSION")=""
%CGI("HTTP_COOKIE")=""
%CGI("HTTP_DERIVED_FROM")=""
%CGI("HTTP_FROM")=""
%CGI("HTTP_HOST")="localhost"
%CGI("HTTP_REFERER")=""
%CGI("HTTP_TE")="deflate, gzip, chunked,
  identity, trailers"
%CGI("HTTP_TITLE")=""
%CGI("HTTP_USER_AGENT")="Opera/9.63
  (Windows NT 5.0; U; ru) Presto/2.1.1"
%CGI("PATH_INFO")="/mwa/hello.mwa"
%CGI("PATH_TRANSLATED")="i:\apache\htdocs\mwa\hello.mwa"
%CGI("QUERY_STRING")="p=123"
%CGI("REMOTE_ADDR")="127.0.0.1"
%CGI("REMOTE_HOST")=""
%CGI("REQUEST_METHOD")="GET"
%CGI("SCRIPT_NAME")="/cgi-bin/nph-minimwa.exe"
%CGI("SERVER_PORT")="80"
%CGI("SERVER_PROTOCOL")="HTTP/1.1"
%CGI("SERVER_SOFTWARE")="Apache/1.3.12
  (Win32) MiniM Web Access/0.8"
```

The exact value and recommendations about each variable CGI environment are specified by the document RFC 3875:

```
http://www.ietf.org/rfc/rfc3875
```

2.2 Page parameters

Query parameters to page can be passed by HTTP client using several ways, most recently used are GET and POST methods. If used GET method query parameters are part of url specification and specified syntactically. For example:

```
http://localhost/mwadir/page.mwa?name1=value1
```

Here passed variable *name1* with value *value1*.

Using GET method HTTP client place query string into HTTP request header. This method can be used to pass not great data. Generally web servers limits url length, about 2 kilobyte. This limitation depends from web server and from web server settings.

Using POST method HTTP client place query parameters to page into HTTP request content, which follows by HTTP request header. This method can be used to pass data with any length, and can be transferred files.

Web clients, proxy servers and web servers have the agreement if page is response to GET method web server must have the same response, and infrastructure internet software allow to cache page content, and responses by POST method are not cached. Note that MWA page processor always insert into HTTP response special headers with cache disallow:

```
Pragma: no-cache
Cache-Control: no-cache
```

This directives specifies that http client and proxy servers disallow to cache page content.

Before page processing MWA page processor analizes query parameters format, place of page parameters and encoding method. In any case, if page parameters are present, ones are placed into %KEY local variable in decoded format. If HTTP client transfer data with encoding type one of the following:

```
text/plain
application/x-www-form-urlencoded
multipart/form-data
```

then MWA processor splits content to several named parts.

%KEY variable structure is:

```
%KEY(name,sequence,nsegment)=segment
```

Here *name* is parameter name. If name specified as empty string, this name and value is not used.

sequence specifies order number of the parameter with the same name. HTTP protocol allow to pass several parameters with the same name. In this case they are differs by order sequence. In the most cases parameter's name is passed only once by query. Sequence number is integer number from 1 (1, 2, 3, ...). If parameter name presents in query only once, MWA page processor uses $sequence = 1$.

nsegment specifies number of value's segment. In order limiting MUMPS variable by total length (32K) big parameter value must be splitted into several segments. *nsegment* is integer number starting from 1. In the most cases parameter's data does not exeeds 32K limit and value can be obtained as

```
%KEY(name,1,1)
```

If HTTP client transfer data using POST or PUT method with the following encoding type:

```
application/octet-stream
```

then MWA processor writes entire content to the %KEY variable like this content have name of one space symbol:

```
%KEY(" ",1,nsegment)
```

Different HTTP clients can differently encode query parameters, for example we can have query

```
http://localhost/mwadir/page.mwa?p="123"
```

this query can be sent as is or can be used special URI encoding

```
http://localhost/mwadir/page.mwa?p=%22123%22
```

MWA page processor in both cases translates parameter's names and values to common decoded representation form.

To see how MWA page processor decode page parametrs insert into MWA page tag like this

```
<?exec zw %KEY?>
```

Chapter 3

MWA Tags

MWA tag is character sequence from symbols <? to ?> including ones. Immediately after symbols <? must follow tag name (or tag type). MWA page processor uses only MWA tags, all other tags are not processed. For example, sequence <?xml version= "1.0" ?> writes out as is.

MWA tags are case insensitive.

MWA tag can have mandatory parameter, can have optional parameter and can have no parameters. Spaces before and after tag parameter are not functional. Common MWA tag syntax is:

```
<?tagname [parameter] ?>
```

MWA page processor make action dependent from tag type, removes all tag's character sequence, and replaces tag's content to empty string or to tag expression. Some tag are conditional tags: if/elif/else/endif and while/endwhile, which can control processing algorithm and can include sequence between tags into page content ones, zero or many times.

Tag's parameters can be a MUMPS command sequence or a calculated MUMPS expression. To execute commands or to calculate expression MWA page processor calls MiniM process. MWA page processor does not have own variables or functions.

MWA page processor does not limit tag's parameters except parameter must be spesified in single line. If parameter must contain character sequence "?>" this sequence must be splitted, for example using concatenation operator.

3.1 BIN

MWA tag *bin* have one mandatory parameter, and this must be one or more MUMPS commands. This commands are executed while tag processed. Commands can write any data into current output device and all this bytes are user to replace tag.

Syntax

```
<?bin commands?>
```

Here commands has two special meaning: 1) line feed as *write !* produces one byte $c(10) and 2) tabular output *write ?N* produces N spaces.

Examples

```
<?bin s i="" f  s i=$o(^glb(i)) q:i=""  w $g(^glb(i))?>
<?bin d OUT^PAGE15()?>
```

All intersepted commands output used to replace tag.

Tags *exec* and *bin* are very similar except the *exec* tag work like text mode and the *bin* tag work like bynary mode.

3.2 ELIF

The *elif* tag have optional parameter, and this parameter is seen as a calculated MUMPS expression. The *elif* tag must follow by previous *if* tag or other *elif* tag.

The tag parameter evaluated and compared with 0. If result is nonzero, MWA page processor produces to output all page content up to next *elif*, *else* or *endif* tags, and all other alternativs, devined by *elif* and *else* tags, are ignored. Instead of *elif* tag is produced empty string. If result is zero, MWA page processor ignores all page content up to next *elif*, *endif*, or *else* tag.

Syntax

```
<?elif expression?>
```

Example

```
<?if $d(%KEY("city"))?>
text about city
<?elif $d(%KEY("country"))?>
text about country
<?else?>
text about nothing
<?endif?>
```

Functional braces defined by *if* - *elif* - *else* - *endif* tags, can be embedded into each other and into *while* - *endwhile* tags.

3.3 ELSE

The *else* tag has not parameter. The *else* tag is replaced by empty string. This tag must follow after previous *if* or *elif* tag and after this tag must be *endif* tag.

If previous logical alternative is not true, next page content after *else* tag is inserted into page responce, otherwise not.

Syntax

```
<?else?>
```

Example

```
<?if $d(%KEY("city"))?>
text about city
<?elif $d(%KEY("country"))?>
text about country
<?else?>
text about nothing
<?endif?>
```

Functional braces defined by *if* - *elif* - *else* - *endif* tags, can be embedded into each other and into *while* - *endwhile* tags.

3.4 ENDIF

The *endif* tag has not parameter. The *elif* tag is replaced by empty string. Before this tag required previous *if*, *elif* or *else* tag. The *endif* tag is end of logical functional braces.

Syntax

```
<?endif?>
```

Example

```
<?if $d(%KEY("city"))?>
text about city
<?elif $d(%KEY("country"))?>
text about country
<?else?>
text about nothing
<?endif?>
```

Functional braces defined by *if* - *elif* - *else* - *endif* tags, can be embedded into each other and into *while* - *endwhile* tags.

3.5 ENDWHILE

The *endwhile* tag have optional parameter, it must be MUMPS command sequence. This commands are executed if previous *while* tag have nonzero evaluated parameter.

If previous *while* tag has nonzero parameter, the *endwhile* tag reposition MWA page processor to previous *while* tag and one executes again. Otherwize the *endwhile* tag is ignored and MWA page processor continue to process page after this tag.

Tags *while* and *endwhile* allow programmer to organize cycles.

Syntax

```
<?endwhile?>
<?endwhile commands?>
```

Example

```
<?exec set counter=1?><?while counter<=5?>
<p>Counter is: <?eval counter?>
<?endwhile set counter=counter+1?>
```

Functional braces defined by *while-endwhile* tags, can be embedded into each other and into *if - elif - else - endif* tags.

3.6 EVAL

The *eval* tag have mandatory parameter and this must be MUMPS calculated expression. MWA page processor call MiniM process to evaluate expression and result used to replace the entire *eval* tag.

Syntax

```
<?eval expression?>
```

The *expression* can contain any MUMPS operators, functions, variables and result used in the string representation, as raw byte sequence.

Example

```
Date now is <?eval $zd($h,13)?>
```

Instead of *eval* name can be used the *m* name.

3.7 EXEC

The *exec* tag have mandatory parameter, this must be MUMPS command sequence. MWA page processor executes ones and all device output is inserted instead of entire *exec* tag. If no any output present, tag is replaced by empty string.

The *exec* tag has two special meaning: 1) line feed (*write !*) inserts symbols $C(13,10) and 2) tabular output (*write ?N*) inserts spaces with padding as need from current line position to specified position. Current position calculates from tag start and from each line feed command (*write !*).

Syntax

```
<?exec commands?>
```

Example

```
<?exec s counter=counter+1?>
<?exec d WriteJS^HTMLAPP()?>
<?exec s i="" f  s i=$o(page(i)) q:i=""  w $g(page(i)),!?>
```

Instead of *exec* tag can be used *x* name.

Tags *exec* and *bin* are very similar except the *exec* tag work like text mode and the *bin* tag work like bynary mode.

3.8 HEAD

The *head* tag have mandatory parameter, it must be a MUMPS command sequence.

MWA page processor call MiniM process to execute parameter's commands and replace entire tag to empty string. Page processor intercept all output and all output counts as strings and each string must be a pair as:

```
name=value
```

All this string from the *head* tag MWA page processor use as HTTP responce headers. Here *name* used as HTTP responce header name and *value* used as value. This additional HTTP headers adds to current HTTP responce header.

Syntax

```
<?head commands?>
```

Example

```
<?head w "Content-type=text/plain"?>
<?head w "Content-encoding=abcdef"?>
<?head w "Content-type=text/xml"?>
```

Head tags can be used in any page place and any times. Later by place *head* tage can replace previous *head* tag result.

If MWA page does not contain any *head* tag, MWA page processor does not change HTTP responce headers and use header with values:

```
HTTP 1.0 200 OK
Content-type: text/html
```

Additionally the *head* tag can create 2 special reserved values: StatusCode and ReasonString. By default, MWA page processor create HTTP responce with settings:

```
StatusCode = 200
ReasonString = OK
```

This mean HTTP resource found and HTTP responce created.

Programmer can change this settings, for example:

```
<?head w "StatusCode=204",!,"ReasonString=No Content",!?>
```

There is existent RFC 2616 document about how to setup HTTP responce headers:

```
http://www.ietf.org/rfc/rfc2616.txt
```

Note that MWA page processor supports only HTTP 1.0 protocol, after page processing CGI module ends to work and disconnects from MiniM process.

3.9 IF

The *if* tag have mandatory parameter, it must be a calculated MUMPS expression. MWA page processor calls MiniM process to evaluate expression and compare with 0.

If result is zero, MWA page processor ignores all page content up to next *elif*, *else* or *endif* tag. If result is nonzero, page content up to next *elif*, *else* or *endif* tag is inserted into responce content and other logical alternatives with *elif* and *else* tags are ignored.

Syntax

```
<?if expression?>
```

Example

```
<?if $d(%KEY("city"))?>
text about city
<?elif $d(%KEY("country"))?>
text about country
<?else?>
text about nothing
<?endif?>
```

Functional braces defined by *if* - *elif* - *else* - *endif* tags, can be embedded into each other and into *while* - *endwhile* tags.

3.10 INCLUDE

The *include* tag have mandatory parameter, it must be a file name. File name used relatively from currently processed MWA page. MWA page processor replace the *include* tag to specified file content with processing this included file. Included file also can contain MWA tags.

While used the *include* tag internal MWA page processor line counter counts from current generated HTTP responce. And after *include* or *insert* tags this counter does not correlate with internal MWA position.

Included file name can have any name and extension.

Syntax

```
<?include filename?>
```

Example

```
<?include pageheader.mwa?>
Page content here...
<?include pagefooter.mwa?>
```

3.11 INSERT

The *insert* tag have mandatory parameter, and it must be a file name. File name used relatively from currently processed MWA page.

The *insert* tag is replaced with specified in parameter file's content as is. Unlike the *include* tag this file does not processed. File can have any name and extension. File content inserts only instead of the *insert* tag, all other MWA page also is processed.

Syntax

```
<?insert filename?>
```

Example

```
<?if '$d(%KEY)?>
<?insert noparameters.html?>
<?else?>
Analize parameters...
<?endif?>
```

3.12 WHILE

The *while* tag have mandatory parameter, it must be MUMPS calculated expression.

MWA page processor call MiniM process, evaluate parameter and compare with 0. If expression is not 0, MWA processor still process page until next appropriate *endwhile* tag. Otherwise all content to next *endwhile* tag is skipped.

Tags *while* and *endwhile* allow programmer to organize cycles.

Syntax

```
<?while expression?>
```

Example

```
<?exec set counter=1?><?while counter<=5?>
<p>Counter is: <?eval counter?>
<?endwhile set counter=counter+1?>
```

Functional braces defined by *while-endwhile* tags, can be embedded into each other and into *if - elif - else - endif* tags.