A web server extension for M/Caché/IRIS and YottaDB
mg_web

M/Gateway Developments Ltd.
Chris Munt

Revision History:
>21 July 2020
>3 August 2020

Contents

1 Introduction

mg_web provides a high-performance minimalistic interface between three popular web servers (Microsoft IIS, Apache and Nginx) and M-like DB Servers (YottaDB, InterSystems IRIS and Cache).

HTTP requests passed to the DB Server via **mg_web** are processed by a simple function of the form:

```
Response = DBServerFunction(CGI, Content, System)
```

Where *CGI* represents an array of CGI Environment Variables, *Content* represents the request payload and *System* is reserved for **mg_web** use.

A simple 'Hello World' function would look something like the following pseudo-code:

```
DBServerFunction(CGI, Content, System)
{
      // Create HTTP response headers
      Response = "HTTP/1.1 200 OK" + crlf
      Response = Response + "Content-type: text/html" + crlf
      Response = Response + crlf
      //
      // Add the HTML content
      Response = Response + "<html>" + crlf
      Response = Response + "<head><title>" + crlf
      Response = Response + "Hello World" + crlf
      Response = Response + "</title></head>" + crlf
      Response = Response + "<h1>Hello World</h1>" + crlf
      return Response
}
```

In production, the above function would, of course, be crafted in the scripting language provided by the DB Server.

2 Web Server Components

In this section we discuss the process for building and configuring the **mg_web** component for all supported web servers.

2.1 mg_web for Microsoft IIS

mg_web for IIS is implemented as an *IIS Native Module*.

2.1.1 Building from source

It is assumed that you have Visual C++ installed.

Copy the contents of **/src/** and **/src/iis/** to a directory of your choice. You should now have the following files in that directory.

```
Makefile.win
mg_web.c
mg_web.h
mg_websys.h
mg_web_iis.cpp
```

To build **mg_web** for IIS (**mg_web_iis.dll):**

```
nmake -f Makefile.win
```

The following command will simply copy the module (**mg_web_iis.dll**) to the **c:\inetpub\mgweb\** directory. You will need to create this directory first.

```
nmake -f Makefile.win install
```
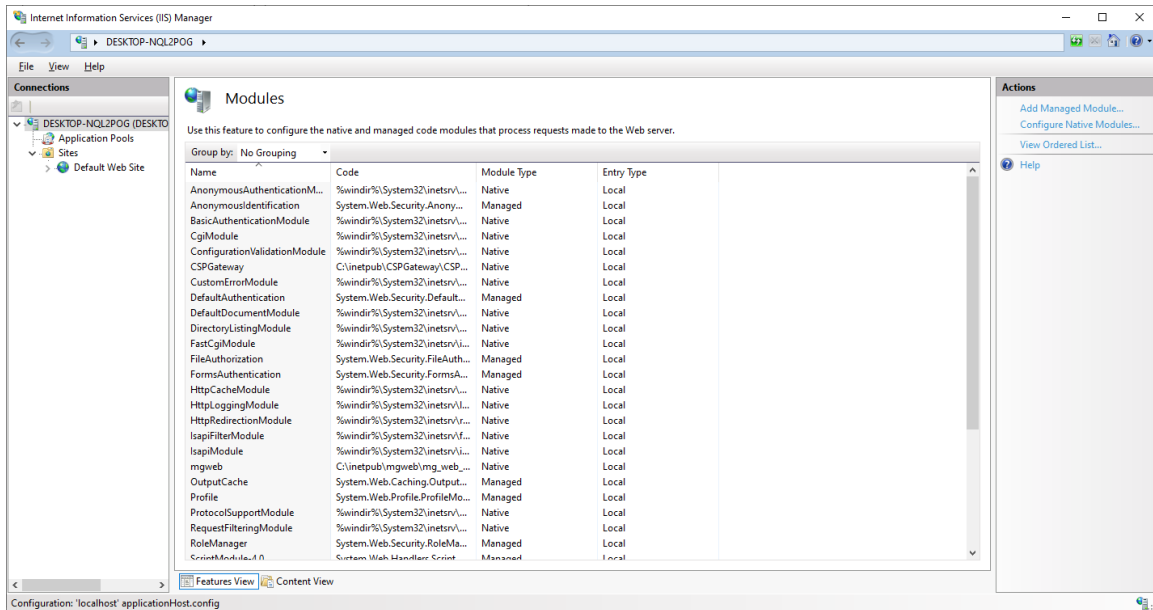
### 2.1.2  Web Server configuration

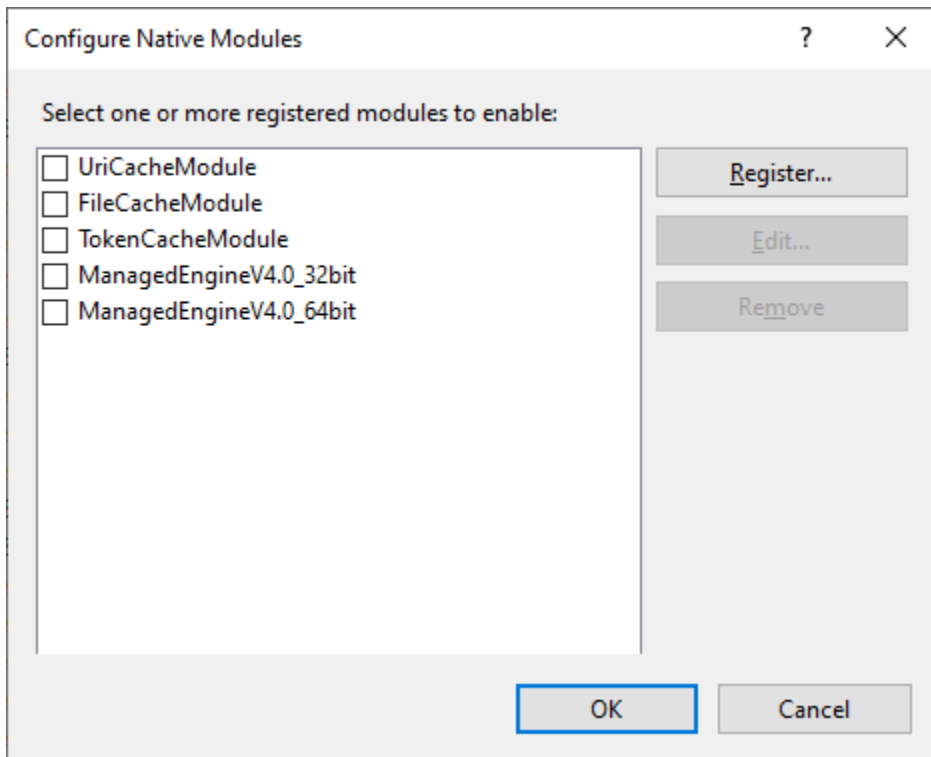The **mg_web** module must be registered as an IIS *'Native Module'*.

Open the **IIS Control Panel** with focus on the root of the IIS installation in the left-hand panel. Open the **Modules** Control Panel.
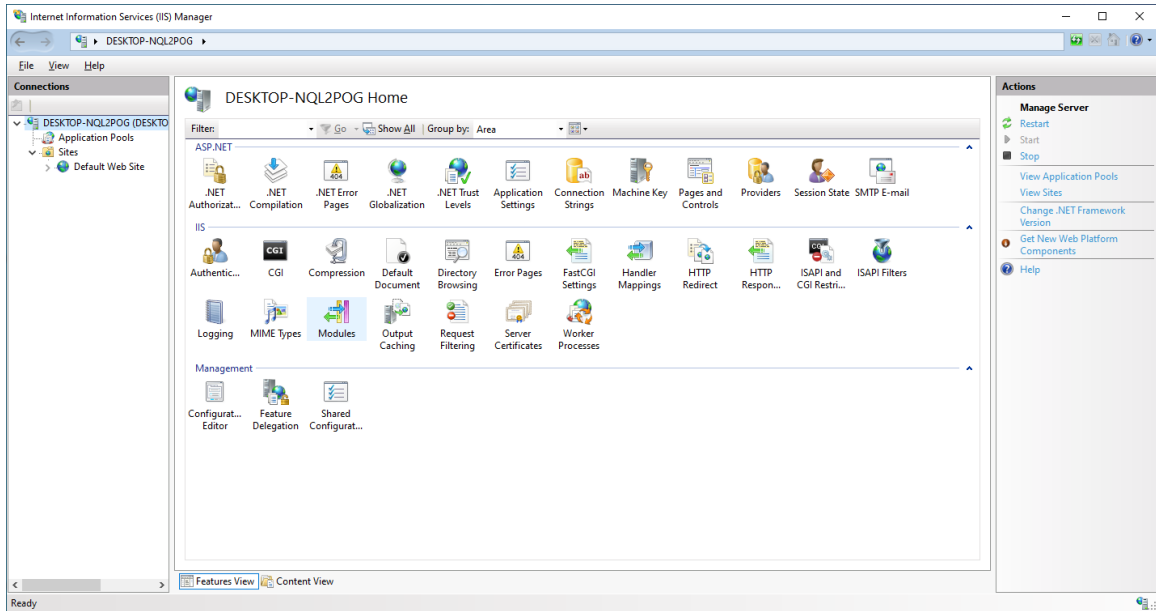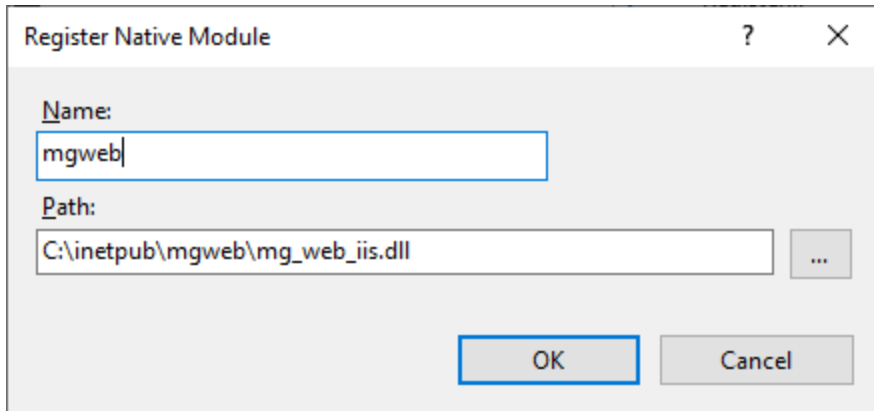
The *Modules* Control Panel.  Choose the '*Configure Native Modules*' option in the right-hand panel.



The '*Configure Native Modules*' Control Panel.  Note the *Register* button.

Press the *Register* button to add the **mg_web** module for IIS (**mg_web_iis**).



You can assign a name of your choice to the registration – **mgweb** is used in the above example. When the **mg_web** module is registered it can be associated with a particular (virtual) application path and/or specific file types.

To create a new virtual path for applications, right-click on the appropriate web site in the left-hand panel and choose the '*Add Application*' option. In the example below, virtual application path *mgweb* was added beneath the *Default Web Site*.
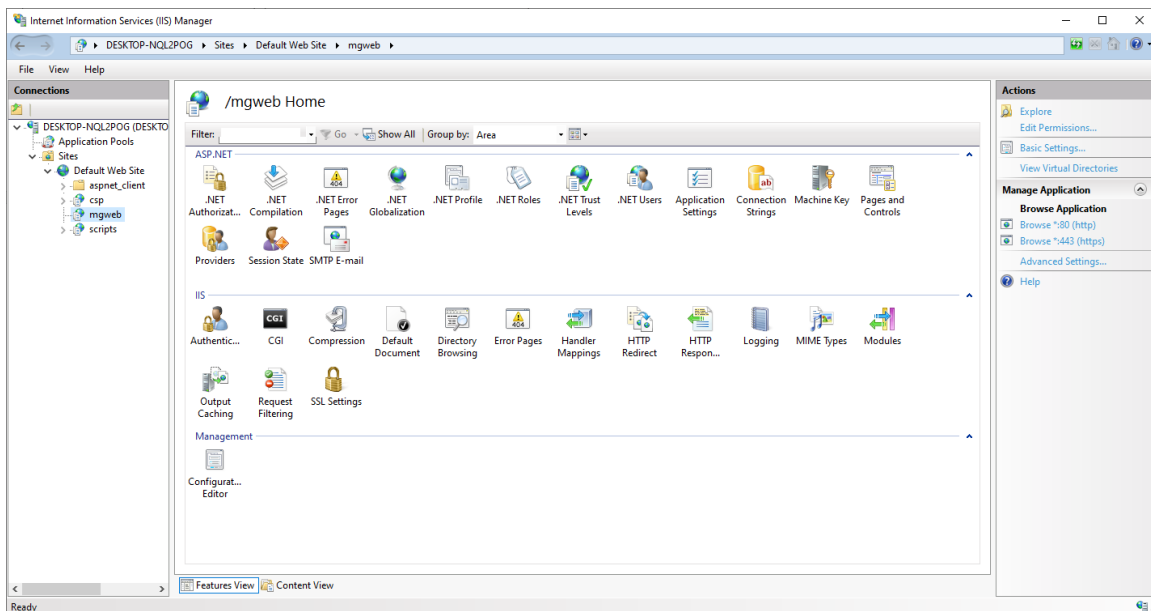
The properties for the virtual application path are shown in the following example. You have to assign an alias (e.g. *mgweb*) and choose an *application pool* to process **mg_web** requests (again named *mgweb* in this example).

Having created a virtual application path for hosting **mg_web** requests, the next task is to map specific file types to the **mg_web** extension. In the following example, we will configure IIS to pass all requests for files of type *.mgw* to **mg_web** for processing.

With the focus on the virtual application path previously created (*mgweb*) in the left-hand panel, open the '*Handler Mappings*' Control Panel and choose '*Add Module Mapping*' in the right-hand panel.
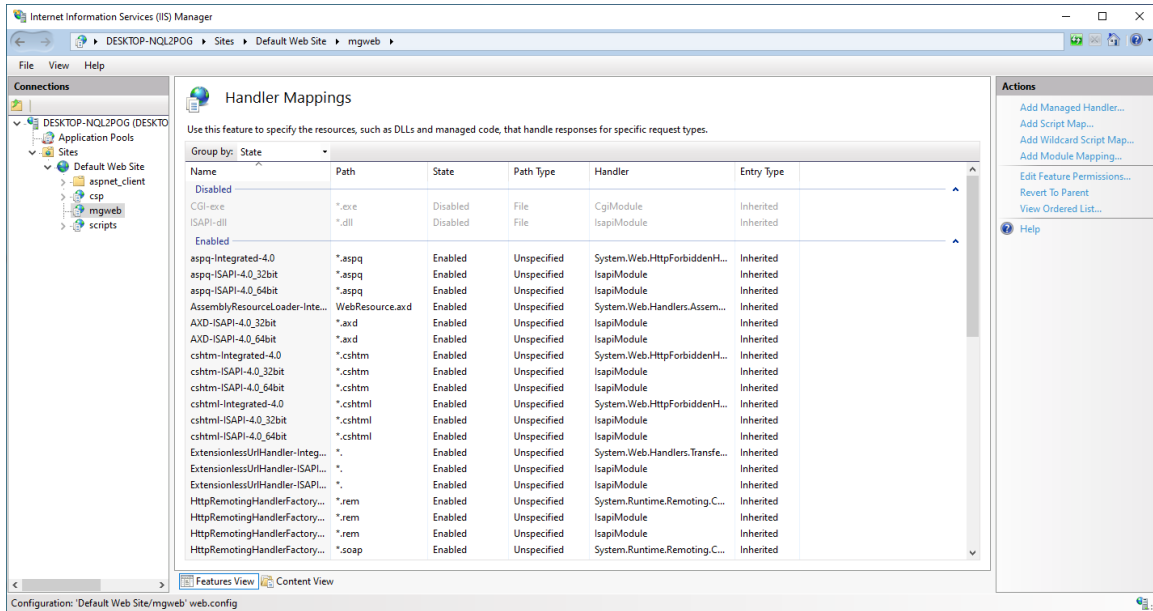
The module name is **mgweb** and we wish to process all files of type *.mgw* with **mg_web** This is defined in the *Request Path* text box.  You can name the Module Mapping with a name of your choice – **mgweb** is used in the example below.
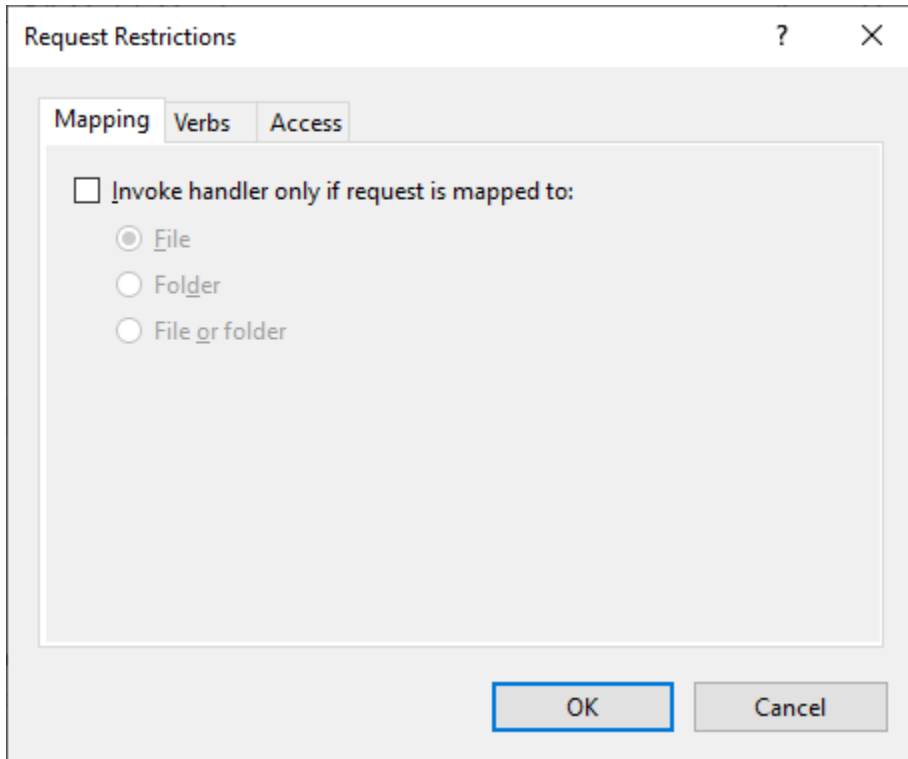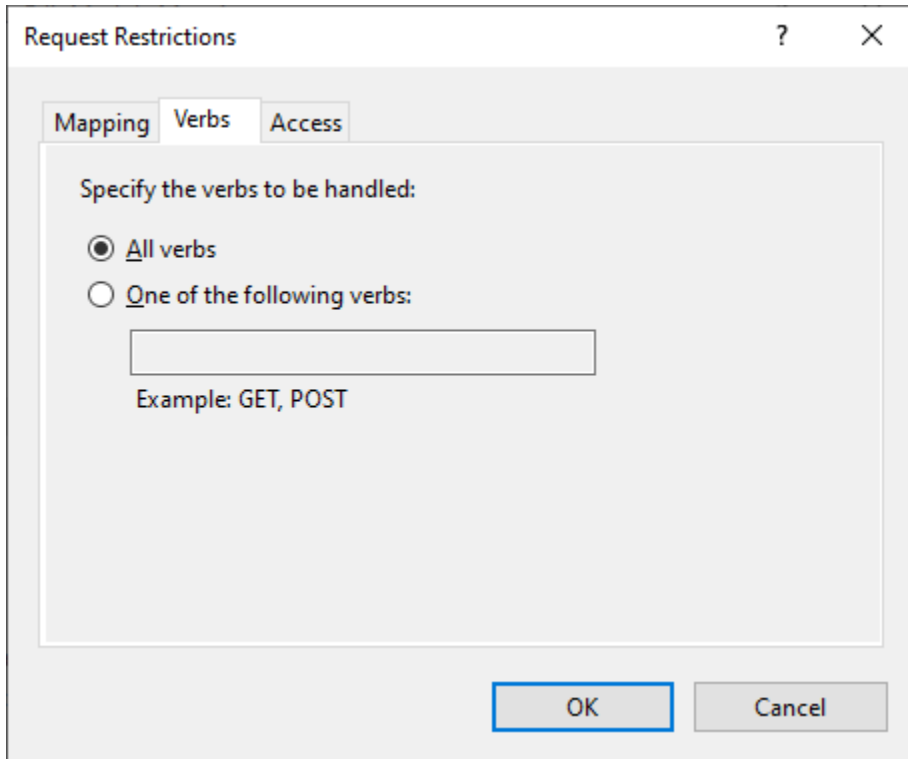
Press the **Request Restrictions** button and make sure that '***Invoke handler only if request is mapped to'*** is unchecked.  Web resources served my **mg_web** do not physically exist on the web server.

Check that all verbs (i.e. HTTP methods) can be served by **mg_web**.

Check that '*Script'* is selected in the *Access* panel.

Request Restrictions                                    ?    ✕

Mapping   Verbs   Access

Specify the access required by the handler:

○ None

○ Read

○ Write

◉ Script

○ Execute

OK        Cancel

The **mg_web** configuration and log files (**mgweb.conf** and **mgweb.log**) will be expected to exist in the same directory as that hosting the **mg_web** module. The IIS worker processes must have permission to access these files. Create these files and use the following commands to grant full access to IIS.

```
cacls C:\inetpub\mgweb\mgweb.conf /E /G IIS_IUSRS:F
cacls C:\inetpub\mgweb\mgweb.log /E /G IIS_IUSRS:F
```

Final contents of the **mg_web** directory (**C:\inetpub\mgweb**):

```
mg_web_iis.dll
mgweb.conf
mgweb.log
```

Finally, restart IIS from the main Windows Services Control Panel (*World Wide Web Publishing Service*).
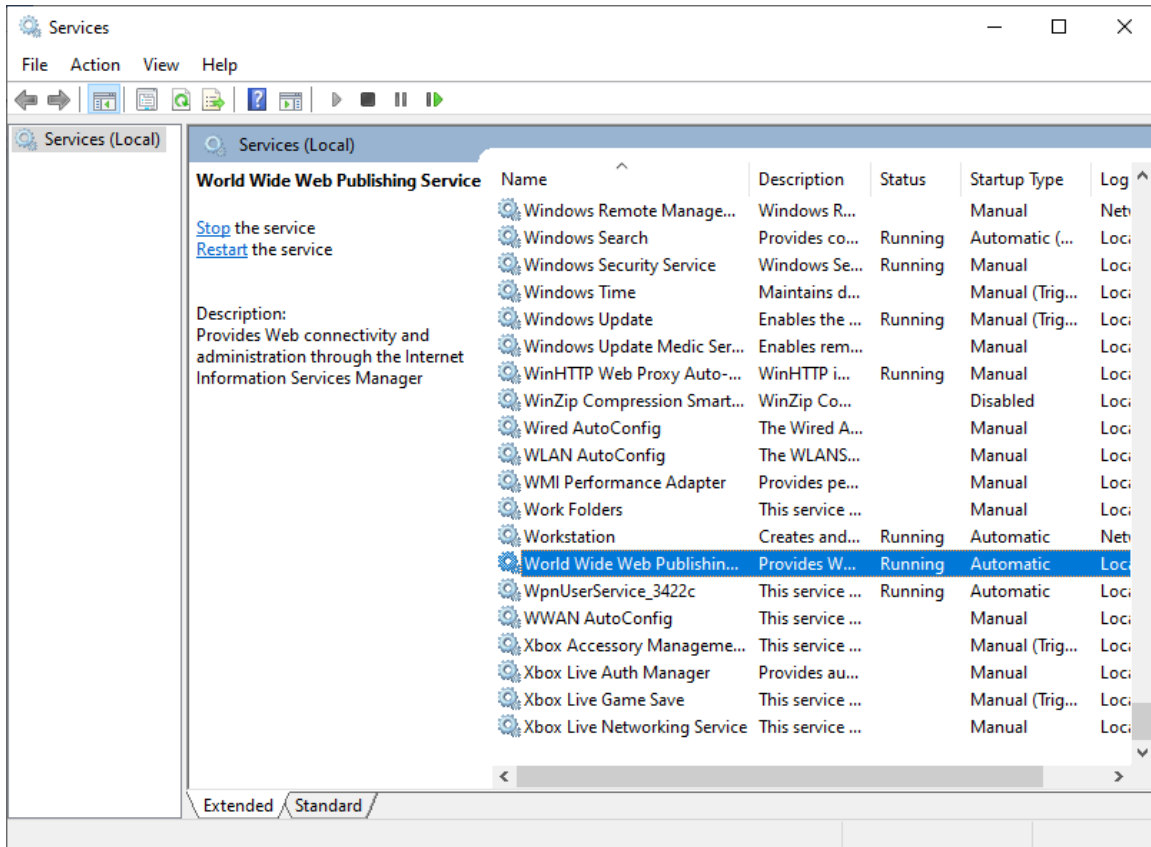
IIS will now pass all requests for files of type *.mgw* in path *mgweb* to **mg_web** for processing. For example:

http://localhost/mgweb/myfile.mgw

　　　　　　　　　　　　　　　15

2.2 mg_web for Apache

mg_web for Apache is implemented as an *Apache Extension Module*.

2.2.1 Building from source

It is assumed that you have a suitable C/C++ compiler installed. For example, GCC for Linux and Visual C++ for Windows.

Copy the contents of **/src/** and **/src/apache/** to a directory of your choice. You should now have the following files in that directory.

```
mg_web.c
mg_web.h
mg_websys.h
mg_web_apache.c
```

To build **mg_web** for Apache (**mg_web_apache.so** or **mg_web_apache.dll**)**:**

Using the Apache extension compiler tool - *APache eXtenSion tool* (**apxs**):

```
apxs -a -i -c mg_web_apache.c mg_web.c
```

This will compile the **mg_web** extension for Apache, install the module in the Apache *modules* directory, and add the following line to the Apache configuration file (***httpd.conf***):

```
LoadModule mg_web_module     modules/mg_web_apache.so
```

2.2.2 Web Server configuration

Check that the **mg_web** module is registered in the Apache configuration file (***httpd.conf***).

```
LoadModule mg_web_module     modules/mg_web_apache.so
```

Add the full path of the **mg_web** configuration file (***mgweb.conf***) and log file (***mgweb.log***) to the Apache configuration file (***httpd.conf***). For example:

```
MGWEBConfigFile c:/Apache2433/conf/mgweb.conf
MGWEBLogFile c:/Apache2433/logs/mgweb.log
```

Create a location through which **mg_web** requests will be processed. For example, **mg_web** can be active for a whole path (*mgweb* in this example):

```
<Location /mgweb>
    MGWEB On
</Location>
```

Alternatively, **mg_web** can be set to configure only specific file types.

```
<Location /mgweb>
    MGWEBFileTypes .mgw .mgweb
</Location>
```

Finally, restart Apache with the new configuration and the web server will now (for example) pass all requests for files of type *.mgw* in path *mgweb* to **mg_web** for processing.

http://apachehost/mgweb/myfile.mgw

2.3 mg_web for Nginx

mg_web for Nginx is implemented as a *Nginx Addon Module*. Unlike the other web server solutions where **mg_web** is created as a dynamically loaded library, **mg_web** functionality is built directly into the Nginx core executable.

2.3.1 Building from source

It is assumed that you have a suitable C/C++ compiler installed. For example, GCC for Linux and Visual C++ for Windows. Additionally, on Windows Nginx is built using the MSYS toolkit and that should be installed. The Nginx instructions for building this web server under Windows can be found here.

http://nginx.org/en/docs/howto_build_on_win32.html

Copy the contents of **/src/** and **/src/nginx/** to a directory of your choice For example, /**opt/mgweb/**. You should now have the following files in that directory.

```
config
mg_web.c
mg_web.h
mg_websys.h
mg_web_nginx.c
```

To build **mg_web** into Nginx:

For UNIX systems, add the **mg_web** module directory to the pre-build configuration step. For example:

```
./configure --prefix=/opt/nginx1180 \
            --with-threads \
            --add-module=/opt/mgweb
```

Note that the '--with-threads' option must be included if **mg_web** is to take advantage of Nginx thread pooling (recommended).

Having run the configuration step, the Nginx web server with **mg_web** included can be built using:

```
make
make install
```

For Windows systems, using the MSYS environment, the process is very similar. Add the **mg_web** module directory to the pre-build configuration step. For example:

```
auto/configure --with-cc=cl --builddir=objs --prefix= \
--conf-path=conf/nginx.conf --pid-path=logs/nginx.pid \
--http-log-path=logs/access.log                  --error-log-
path=logs/error.log \
--sbin-path=nginx.exe \
--http-client-body-temp-path=temp/client_body_temp \
--http-proxy-temp-path=temp/proxy_temp \
--http-fastcgi-temp-path=temp/fastcgi_temp \
--with-cc-opt=-DFD_SETSIZE=1024 \
--without-http_rewrite_module \
--without-http_gzip_module \
--with-select_module \
--add-module=/opt/mgweb
```

Having run the configuration step, the Nginx web server with **mg_web** can be built using:

```
nmake -f objs/Makefile
```

2.3.2 Web Server configuration

Add the full path of the **mg_web** configuration file (**mgweb.conf**) and log file (**mgweb.log**) to the Nginx configuration file (*nginx.conf*). These directives should be added to the *http* section of *nginx.conf*. For example:

```
MGWEBConfigFile /opt/nginx1180/conf/mgweb.conf
MGWEBLogFile /opt/nginx1180/logs/mgweb.log
```

Create a location through which **mg_web** requests will be processed. These directives should be added to the *server* section of *nginx.conf*. For example, **mg_web** can be active for a whole path:

```
        location /mgweb {
            MGWEB On;
            MGWEBThreadPool default;
        }
```

Alternatively, **mg_web** can be set to configure only specific file types.

```
        location /mgweb {
            MGWEBFileTypes .mgw .mgweb;
            MGWEBThreadPool default;
        }
```

Note that in both cases **mg_web** is configured to use a Nginx thread pool called *default*.

Finally, restart Nginx with the new configuration and the web server will now (for example) pass all requests for files of type *.mgw* in path *mgweb* to **mg_web** for processing.

http://nginxhost/mgweb/myfile.mgw

3 DB Server Components

The installation package contains two DB Server routines (i.e. M routines): **%zmgsi** and **%zmgsis**. In this section we will look at the procedure for installing them.

3.1 Installation for InterSystems Caché or IRIS

Log in to the Manager Namespace (%SYS) and install the **zmgsi** routines held in either **/m/zmgsi_cache.xml** or **/m/zmgsi_iris.xml** as appropriate.

```
do $system.OBJ.Load("/m/zmgsi_cache.xml","ck")
```

Change to your development UCI and check the installation:

```
do ^%zmgsi

M/Gateway Developments Ltd - Service Integration Gateway
Version: 3.3; Revision 11 (3 August 2020)
```

3.2 Installation for YottaDB

The instructions given here assume a standard 'out of the box' installation of **YottaDB** deployed in the following location:

```
/usr/local/lib/yottadb/r122
```

The primary default location for routines:

```
/root/.yottadb/r1.22_x86_64/r
```

Copy all the routines (i.e. all files with an 'm' extension) held in the GitHub **/yottadb** directory to:

```
/root/.yottadb/r1.22_x86_64/r
```

Change directory to the following location and start a **YottaDB** command shell:

```
cd /usr/local/lib/yottadb/r122
./ydb
```

Check the installation:

```
do ^%zmgsi
```

```
        M/Gateway Developments Ltd - Service Integration Gateway
        Version: 3.3; Revision  11 (3 August 2020)
```

Note that the version of **zmgsi** is successfully displayed.

3.3 Setting up the DB Server network service

The default TCP server port on which the DB Server (**%zmgsi)** listens is **7041**. If you wish to use an alternative port then modify the following instructions accordingly. The SIG will, by default, expect the database server to be listening on port **7041** of the local server (localhost).

3.3.1 InterSystems Caché or IRIS

Start the M-hosted concurrent TCP service in the Manager UCI (%SYS):

```
        do start^%zmgsi(0)
```

To use a server TCP port other than 7041, specify it in the start-up command (as opposed to using zero to indicate the default port of 7041).

3.3.2 YottaDB

Network connectivity to **YottaDB** is managed via the **xinetd** service. First create the following launch script (called **zmgsi_ydb** here):

```
        /usr/local/lib/yottadb/r122/zmgsi_ydb
```

Content:

```
        #!/bin/bash
        cd /usr/local/lib/yottadb/r122
        export ydb_dir=/root/.yottadb
        export ydb_dist=/usr/local/lib/yottadb/r122
        export
ydb_routines="/root/.yottadb/r1.22_x86_64/o*(/root/.yottadb/r1.22_x86_6
4/r /root/.yottadb/r) /usr/local/lib/yottadb/r122/libyottadbutil.so"
        export ydb_gbldir="/root/.yottadb/r1.22_x86_64/g/yottadb.gld"
        $ydb_dist/ydb -r xinetd^%zmgsis
```

Create the **xinetd** script (called **zmgsi_xinetd** here):

```
        /etc/xinetd.d/zmgsi_xinetd
```

Content:

```
service zmgsi_xinetd
{
     disable          = no
     type             = UNLISTED
     port             = 7041
     socket_type      = stream
     wait             = no
     user             = root
     server           = /usr/local/lib/yottadb/r122/zmgsi_ydb
}
```

- Note: sample copies of **zmgsi_xinetd** and **zmgsi_ydb** are included in the /unix directory.

Edit the services file:

```
/etc/services
```

Add the following line to this file:

```
zmgsi_xinetd              7041/tcp                              # ZMGSI
```

Finally restart the **xinetd** service:

```
/etc/init.d/xinetd restart
```

4 General mg_web configuration (mgweb.conf)

The **mg_web** configuration file (*mgweb.conf*) contains the instructions for connecting to each DB Server and which web paths should be routed to each DB Server.

There are general settings such as request timeouts:

```
timeout 30
```

You can control the amount of information written to the Event Log:

```
log_level <directives>
```

Where the *directives* may include:

- **e** - Log error conditions.
- **f** - Log the internal **mg_web** functions invoked while processing a web request. Also, the basic framing information for the request and response buffers will be recorded.
- **t** - Log the contents of the request and response data buffers transmitted between **mg_web** and the DB Server.

Example:

```
log_level eft
```

You can also define lists of CGI environment variables to be sent to the DB Server with each request. For example, the following directive will instruct **mg_web** to send to the DB Server all CGI environment variables derived from client HTTP request headers (*HTTP**) and the web server *Server Software* with each request.

```
<cgi>
   HTTP*
   SERVER_SOFTWARE
</cgi>
```

Note that, by default, **mg_web** will only send the following CGI environment variables to the DB Server with each request.

```
REQUEST_METHOD
SCRIPT_NAME
QUERY_STRING
```

4.1 Defining Servers

The following examples will illustrate how DB Server access should be defined for **mg_web**.

Network based access to InterSystems IRIS (or Cache) listening on TCP port 7041:

```
<server local>
    type IRIS
    host localhost
    tcp_port 7041
    username _SYSTEM
    password SYS
    namespace USER
</server>
```

API based access to InterSystems IRIS (or Cache):

```
<server local>
    type Cache
    path /opt/cache20181/mgr
    username _SYSTEM
    password SYS
    namespace USER
</server>
```

Network based access to YottaDB listening on TCP port 7041:

```
<server local>
    type YottaDB
    host localhost
    tcp_port 7041
</server>
```

API based access to YottaDB:

```
<server local>
    type YottaDB
    path /usr/local/lib/yottadb/r122
    <env>
        ydb_dir=/root/.yottadb
        ydb_rel=r1.22_x86_64
        ydb_gbldir=/opt/webapps/yottadb.gld
        ydb_routines=/opt/webapps
        ydb_ci=/usr/local/lib/yottadb/r122/zmgsi.ci
    </env>
</server>
```

The routine interface file (named *zmgsi.ci* in the above example) must contain the following line:

```
ifc_zmgsis: ydb_string_t * ifc^%zmgsis(I:ydb_string_t *,
I:ydb_string_t *, I:ydb_string_t *)
```

4.2 Defining Paths

The following examples will illustrate how web paths should be defined for **mg_web**.

The root path (effectively the default mapping):

```
<location />
   function web^%zmgweb
   servers local
</location>
```

Further examples:

```
<location /mgweb/path1>
   function web1^%zmgweb
   servers local1
</location>

<location /mgweb/path2>
   function web2^%zmgweb
   servers local2
</location>
```

A hierarchical system of inheritance for the paths is applied. For Example:

http://webserver/mgweb/path1/file.mgw

This request will be routed to DB Server **local1**

http://webserver/mgweb/path2/file.mgw

This request will be routed to DB Server **local2**

http://webserver/mgweb/path2/abc/file.mgw

This request will be routed to DB Server **local2**

http://webserver/mgweb/file.mgw

This request will be routed to DB server **local**

---

http://webserver/xyz/file.mgw

This request will be routed to DB server **local** (assuming the web server is configured to pass requests with a path of **/xyz** to **mg_web**).

4.3 Complete example mgweb.conf

```
timeout 30

<cgi>
   HTTP*
   SERVER_SOFTWARE
</cgi>

<server local>
   type IRIS
   host localhost
   tcp_port 7041
   username _SYSTEM
   password SYS
   namespace USER
</server>

<location />
   function web^%zmgweb
   servers local
</location>
```

4.4 Reporting configuration errors

It is essential that the **mg_web** event log file is specified correctly and that the web server worker processes are granted full access to it as any configuration errors will be reported in the log.

When a web server worker process successfully links to the **mg_web** library, a message such as that shown below will be written to the **mg_web** event log.

```
>>> Time: Thu Jul 23 16:21:44 2020; Build: 1.0.1 pid=9364;tid=27368;
   mg_web: worker initialization
   configuration: C:/inetpub/mgweb/mgweb.conf
```

If a configuration error is detected it will be reported after the initialization message. For example, if parameter *tcp_port* is not specified correctly, a sequence of messages similar to those shown below will be reported.

```
>>> Time: Thu Jul 23 16:21:44 2020; Build: 1.0.1 pid=9364;tid=27368;
    mg_web: worker initialization
    configuration: C:/inetpub/mgweb/mgweb.conf
>>> Time: Thu Jul 23 16:21:44 2020; Build: 1.0.1 pid=9364;tid=27368;
    mg_web: configuration error
    Invalid 'server' parameter 'tcpport' on line 11
```

5 The mg_web DB Server function

The signature of DB Server functions for **mg_web** is as follows:

```
web^%zmgweb(%cgi,%content,%sys)
   ; process request and generate response
   Quit response
```

Where:

```
%cgi:     List of CGI Environment Variables
%content: The request payload (if any)
%sys:     Reserved for mg_web use.
```

Of course, the function may be named as you wish but must match the corresponding *function* entry in the **mg_web** configuration file (*mgweb.conf*).

There are a number of 'helper' functions available to **mg_web** functions. These are described below.

Parse content of type 'application/x-www-form-urlencoded' OR a QUERY_STRING to return an array of name/value pairs:

```
Set %status=$$nvpair^%zmgsis(.%nv,%content)
```

Where:

```
%nv:      An array of name/value pairs
%content: The request payload or a QUERY_STRING
```

This function will un-escape all components before placing them in the name/value pair array.

URL decoding function (URL unescaping):

```
Set %decoded=$$urld^%zmgsis(%encoded)
```

Where:

```
%encoded: URL-escaped item.
%decoded: URL-unescaped item.
```

URL encoding function (URL escaping):

```
Set %decoded=$$urle^%zmgsis(%decoded)
```

Where:

```
%decoded: URL-unescaped item.
%encoded: URL-escaped item.
```

Determine the maximum string length for this DB Server installation:

```
Set %max=$$getmsl^%zmgsis()
```

Where:

```
%max:      Maximum string length.
```

6 License

Copyright (c) 2019-2020 M/Gateway Developments Ltd,
Surrey UK.
All rights reserved.

http://www.mgateway.com
Email: cmunt@mgateway.com

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.