MGateway Service Integration Gateway (SIG)

Integrating Services to Applications

MGateway Ltd.
Chris Munt

Revision History:

11 March 2020
21 December 2020
11 February 2021
18 August 2021
2 December 2021
27 May 2023

Contents

1 Introduction

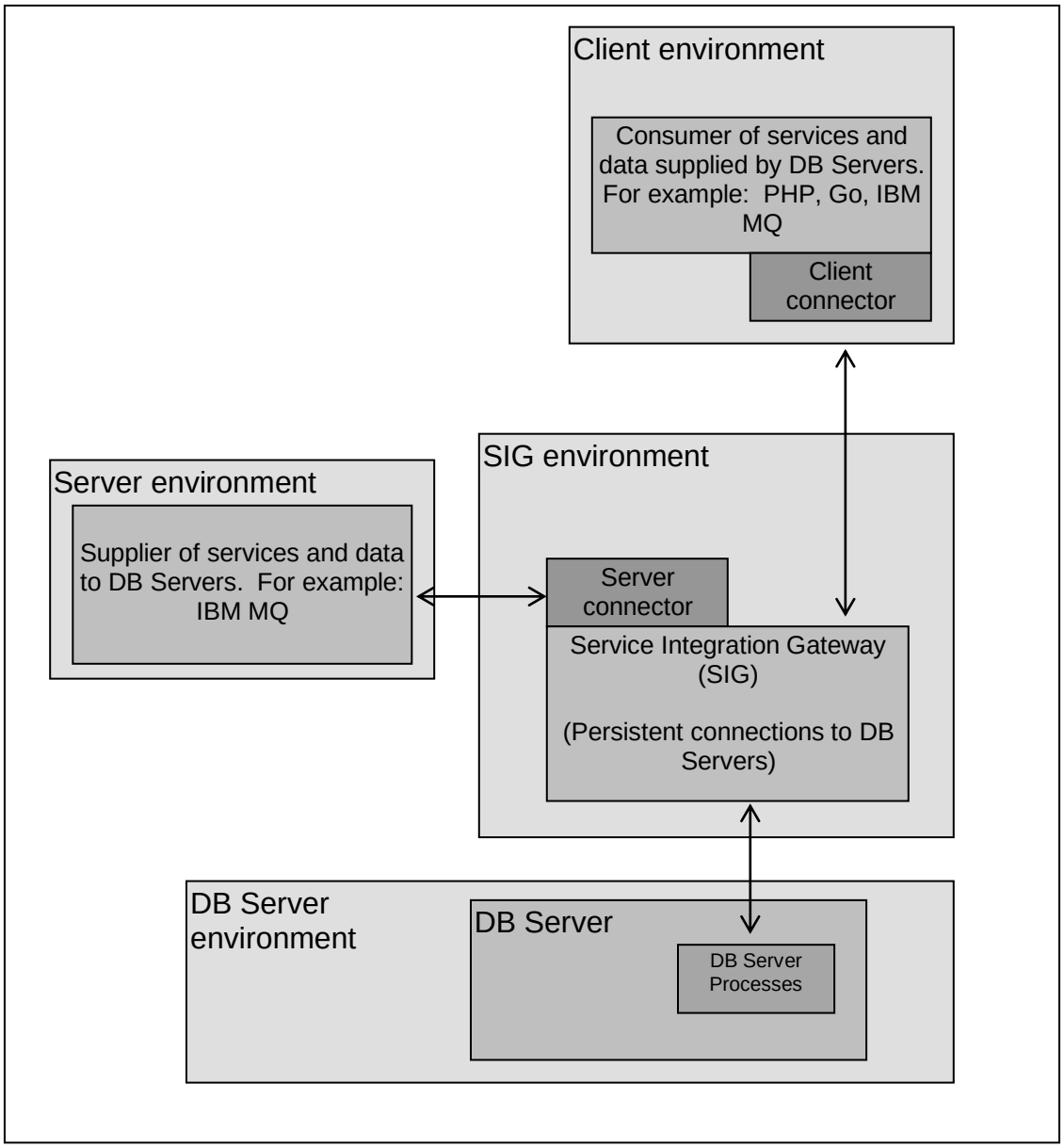
The MGateway Service Integration Gateway (**SIG**) is an Open Source network-based service developed for InterSystems Caché/IRIS and the YottaDB Database Servers. It will also work with the GT.M database and other M-like DB Servers. Its core function is to manage connectivity, process and resource pooling for M-like DB Servers. Persistent connections to DB Servers are permanently available to all services dependent upon the **SIG.**

The following products are able to take advantage of the connection pool provided by the **SIG**: PHP (**mg_php**), Go (**mg_go**), Python (**mg_python**), Ruby (**mg_ruby**), and IBM MQ (formerly WebSphere MQ).

The **SIG** is responsible for managing and maintaining the lines of communication between participating software components. Individual components provide specific services to web applications. For example, in **mg_php**, M-based DB Servers provide a database service to the PHP environment. In **mg_ibmmq**, DB Servers provide a database service to the IBM MQ environment. For cases where DB Servers are acting as a client to IBM MQ, the **SIG** manages and implements the communication between DB Server software and IBM MQ's message queues.

2 General Architecture

The various components making up the Service Integration Gateway (SIG) are shown in the following diagram.



3 Installing the SIG

There are three parts to the Service Integration Gateway installation and configuration.

- The SIG executable (**mgsi** or **mgsi.exe**).
- The database (or server) side code: **zmgsi.**
- A network configuration to bind the former two elements together.

3.1 Pre-requisites

It is assumed that you have a suitable DB Server installed:

Either:

- InterSystems Caché or IRIS http://www.intersystems.com

Or:

- YottaDB https://www.yottadb.com

3.2 Installing the DB Superserver

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

3.2.1 InterSystems Caché or IRIS

Log in to the %SYS Namespace and install the **zmgsi** routines held in **/isc/zmgsi_isc.ro**.

```
do $system.OBJ.Load("/isc/zmgsi_isc.ro","ck")
```

Change to your development Namespace and check the installation:

```
do ^%zmgsi

MGateway Ltd - Service Integration Gateway
Version: 4.5; Revision 37 (9 March 2025)
```

3.2.2 YottaDB

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

```
/usr/local/lib/yottadb/r138
```

The primary default location for routines:

```
/root/.yottadb/r1.38_x86_64/r
```

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

```
/root/.yottadb/r1.38_x86_64/r
```

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

```
cd /usr/local/lib/yottadb/r138
./ydb
```

Check the installation:

```
do ^%zmgsi

MGateway Ltd - Service Integration Gateway
Version: 4.5; Revision 37 (9 March 2025)
```

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

3.3 Starting the DB Superserver

The SIG executable (**mgsi** or **mgsi.exe**), by default, listens on TCP port **7040**.

The default TCP server port on which the DB Superserver (**%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 DB Superserver to be listening on port **7041** of the local server (localhost).

3.3.1 Starting the DB Superserver from the DB command prompt

- For InterSystems DB servers the concurrent TCP service should be started in the **%SYS** Namespace.

Start the DB Superserver using the following command:

```
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).

3.3.1.1 Using InterSystems TLS configurations

DB Superserver version 4.4 (and later) can accept secured connections from clients over TLS. This facility is only available with InterSystems DB Servers. To use an InterSystems TLS ***Server*** configuration, specify this configuration name as the second argument to the DB Superserver start function.

```
Do start^%zmgsi(0,"MyInterSystemsTLSServerConfiguration")
```

3.3.1.2 Restricting the type of requests accepted

DB Superserver version 4.5.26 (and later) includes an option to restrict the type of requests accepted. A list of acceptable request types can be specified in the third argument to the DB Superserver start-up command.

```
Do start^%zmgsi(<port>,<tls>,<request_types>)
```

request_types is a comma-separated list containing one or more of the following.

http (or https): Allow http requests (via **mg_web**).
globals: Allow manipulation of global records (e.g. via **mg-dbx** or **mg_php**).
functions: Allow calls to M functions (e.g. via **mg-dbx** or **mg_php**).
classes: Allow manipulation of InterSystems classes (e.g. via **mg-dbx** or **mg_php**).

Example 1: Allow only the processing of http requests via **mg_web**:

```
Do start^%zmgsi(0,"","http")
```

Example 2: Allow access to M globals and functions:

```
Do start^%zmgsi(0,"","globals,functions")
```

In the absence of a third argument, all request types will be allowed. In other words, the default behaviour is equivalent to:

```
Do start^%zmgsi(0,"","http,globals,functions,classes")
```

3.3.2 Starting YottaDB Superserver processes via the xinetd daemon

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

```
/usr/local/lib/yottadb/r138/zmgsi_ydb
```

Content:

```
#!/bin/bash
cd /usr/local/lib/yottadb/r138
export ydb_dir=/root/.yottadb
export ydb_dist=/usr/local/lib/yottadb/r138
export
ydb_routines="/root/.yottadb/r1.38_x86_64/o*(/root/.yottadb/r1.38_x86_6
4/r /root/.yottadb/r) /usr/local/lib/yottadb/r138/libyottadbutil.so"
export ydb_gbldir="/root/.yottadb/r1.38_x86_64/g/yottadb.gld"
$ydb_dist/ydb -r xinetd^%zmgsis
```

- Note that you should, if necessary, modify the permissions on this file so that it is executable. For example:

```
chmod a=rx /usr/local/lib/yottadb/r138/zmgsi_ydb
```

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/r138/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
```

3.4 The SIG executable

In this section we will look at the procedure for building the SIG from its source code and creating a new installation or upgrading an existing one.

3.4.1 UNIX Systems

Building the SIG from source

Invoke the build procedure from the /src directory (i.e. the directory containing the **Makefile** file):

```
make
```

Installing the SIG executable

The SIG executable (**mgsi**) may be installed in a directory of your choice (e.g. /usr/mgsi/bin/). The SIG configuration and log file will be written to the same location.

Before you start: If you are upgrading an existing installation of this software, you must close down the SIG first. To do this, invoke the following command from within the SIG directory:

```
./mgsi -stop
```

Or:

```
kill -TERM `cat mgsi.pid`
```

The SIG binary (mgsi) can be installed in a directory of your choice. For example:

```
/usr/mgsi/bin/
```

The SIG provides a web-based management interface through which it can be configured and monitored. The SIG contains a built-in HTTP (web) server for this purpose. This built-in web server is accessible through the service's listening TCP port (usually 7040).

http://[server]:7040/mgsi/mgsisys.mgw

3.4.2 Windows Systems

Building the SIG from source

Invoke the build procedure from the /src directory (i.e. the directory containing the **Makefile.win** file):

```
nmake /f Makefile.win
```

Installing the SIG executable

The SIG executable (**mgsi.exe**) may be installed in a directory of your choice (e.g. C:\mgsi\bin\). The SIG configuration and log file will be written to the same location.

Before you start: If you are upgrading an existing installation of this software, you must first close down the SIG. In addition to closing down these components you must also remove the SIG from the Windows Services Database if you plan to install the new version of this module in a different location. To do this, run the following command and follow the instructions printed on the screen.

```
mgsi -stop
```

The SIG executable (mgsi.exe) can be installed in a directory of your choice. For example:

```
C:\mgsi\bin\
```

4 Operating the SIG

4.1 UNIX Systems

Start the core software from the SIG 'bin' directory (e.g. /usr/mgsi/bin/) by entering the following command directly from the UNIX shell Command prompt:

```
./mgsi
```

This command will write messages to the command window indicating progress then exit to the Command Line. The SIG service will run in the background as a UNIX daemon process. The UNIX process number of this daemon process will be recorded in file: **mgsi.pid**.

When the SIG is started for the first time, the default configuration will be created in the configuration file: mgsi.ini.

Use the following command to subsequently shut down the SIG daemon:

```
./mgsi -stop
```

Use the following command to list the options supported by the SIG daemon:

```
./mgsi -h
```

Having started the SIG as described above, use the following URL to access the web-based configuration and management suite for the SIG:

http://[server]:7040/mgsi/mgsisys.mgw

The management forms allow you to configure the core SIG software and connectivity to other environments – for example, connectivity to the DB Servers. The SIG will automatically configure itself for access to a DB Server operating on the same host (i.e. 127.0.0.1 or localhost). The local DB Server is named 'LOCAL' within the SIG configuration.

Setting the SIG service to automatically start-up at system boot-time.

The outline procedure given below is based on the Red-Hat method for managing the startup and closedown of daemon processes at system startup and closedown. The procedure described should be the same (if not identical) to the procedure for other Linux systems and, indeed, other UNIX systems.

- The first step is to create a start-up (and close-down) script in `'/etc/rc.d/init.d/'` directory. Call the script file, say, `'mgsi'`. The following script will be sufficient for starting mgsi at system start-up time and closing it down at system shutdown time. The paths shown below assume that you have a default installation of the SIG. Modify the paths accordingly if this is not the case.

```
#!/bin/sh
#
# Startup script for MGateway's SIG Daemon
#
# chkconfig: 345 85 15
# description: Service Integration Gateway.
# processname: mgsi
# pidfile: /usr/mgsi/bin/mgsi.pid
# config:  /usr/mgsi/bin/mgsid.ini
# config:  /usr/mgsi/bin/mgsi.ini
# config:  /usr/mgsi/bin/mgsi.log


# Source function library.
. /etc/rc.d/init.d/functions

# See how we were called.
case "$1" in
  start)
        echo -n "Starting MGateway's SIG Daemon: "
        PWD=`pwd`
        cd /usr/mgsi/bin/
        daemon /usr/mgsi/bin/mgsi -s
        cd $PWD
        echo
        ;;
  stop)
        echo -n "Shutting down MGateway's SIG Daemon: "
        kill -TERM `cat /usr/mgsi/bin/mgsi.pid`
        echo
        ;;
  *)
        echo "Usage: $0 {start|stop}"
        exit 1
esac

exit 0
```

- Create a symbolic link in directory '/etc/rc.d/rc3.d' for the start-up call. Using the following scheme will result in the SIG service starting at the same time as Apache. Apache is usually started according to the link 'S85httpd'. The 'S' component of the link name refers to 'start-up' and the number indicates the order in which processes will be started.

```
ln -s ../init.d/mgsi S85mgsi
```

- Create a symbolic link in '/etc/rc.d/rc2.d' for the close-down call (the 'K' component of the link name refers to 'Kill process'):

```
ln -s ../init.d/mgsi K15mgsi
```

4.2 Windows Systems

Start the core software from the SIG 'bin' directory (c:\mgsi\bin\) by entering the following command directly from the Windows Command Line:

```
mgsi -start
```

The first time this module is run, it will perform two tasks: Firstly, it will install itself as a Windows Service in the Windows Service database and, secondly, it will start it up. The command will write messages to the command window indicating progress then exit to the Command Line.

Subsequent use of this command will invoke a simple character-based management suite from which you can restart, pause, stop or uninstall the SIG service. Just follow the instructions printed on the screen. Alternatively, once the SIG service is installed and operational you can subsequently manage it from the Windows Services Management console. In particular, you can use the Windows Services Management Console to indicate that SIG should be automatically started when the operating system is started.

When the SIG is started for the first time, the default configuration will be created in the configuration file**: mgsi.ini**.

Having started the SIG as described above, use the following URL to access the web-based configuration and management suite for the SIG:

http://[server]:7040/mgsi/mgsisys.mgw

The management forms allow you to configure the core SIG software and connectivity to other environments – for example, connectivity to DB Servers. The SIG will

automatically configure itself for access to a DB Server operating on the same host (i.e. 127.0.0.1 or localhost). The local DB Server is named 'LOCAL' within the SIG configuration.

5 Configuration

In this section, the various SIG configuration options will be discussed.

5.1 Service Integration Gateway

These options allow you to configure the Service Integration Gateway and monitor its status.

5.1.1 Main Configuration

- **Internal HTTP Service Status**

This allows you to set HTTP access to these forms as 'Enabled' or 'Disabled'.

- **Authorized Managers**

This is a comma-separated list of IP addresses. These represent the client machines from which the SIG web-based systems management forms can be accessed.

Example:

```
123.12.12.3,123.12.13.5
```

The following directive will grant systems management access to all clients (not recommended for routine use).

```
*.*.*.*
```

5.1.2 Viewing the System Status

This option shows the current status of the SIG. All connections to other services are shown – for example, connections to DB Servers opened for the **mg_php** component. Also, the services provided by the SIG installation are shown. For example, the service that listens for requests from the PHP environment.

The following columns are shown in the status form:

- **Connection Number**

This is the internal connection number for connections held to remote services.

- **Service Type**

The type of server connected to (for example, YottaDB), or, in the case of services provided to other clients (for example, PHP and JSP), the type of service offered. Connections to other services are shown in BLUE. Services supplied to other clients are shown in RED.

- **Server Name**

In the case of connections to other servers, the name of the server as defined in the SIG configuration.

- **IP Address**

In the case of connections to other servers, the IP address of the remote server as defined in the SIG configuration. Services supplied are defined as 'Local'.

- **TCP Port**

In the case of connections to other servers, the TCP port number on which the remote server is listening. For example, the YottaDB listener (%zmgsi) usually listens on TCP port 7041. For cases where a Unix super server ([x]inetd) is used, the port listed will be that on which the super server is configured to listen.

Services are usually supplied on port 7040.

- **Server NameSpace**

This represents the directory in which the service is operating. For connections to InterSystems DB Servers, this will take the value of the NameSpace.

- **Server Process ID**

The process ID for the service – for example the YottaDB process (or job) number ($Job).

- **Status**

Indicates whether a connection to a foreign service is busy (in use) or 'free' to accept further requests. In the case of locally supplied services, the status should read 'Enabled'.

- **Requests**

The number of requests (or 'hits') processed by a connection to a remote service, or the total number of requests received by a service supplied locally.

- **Close**

Use this button to forcing a particular connection to close.

5.1.3 Viewing the Event Log

This log shows all error conditions encountered within the SIG environment. Further diagnostic information will be captured depending on the level of logging required.

The log can be displayed in forward or reverse date order. Click on the hyperlink to reverse the order of the log.

5.2 Managing Connections to DB Servers

These options allow you to configure and manage connections to the DB Servers.

5.2.1 Configuration

This is the general configuration. The parameters specified in this section will be applied to all DB Servers.

- **Server Response Timeout**

The amount of time the SIG will wait for a response from the DB Server before returning a timeout error to the client.

- **Timeout for Queued Requests**

The amount of time the SIG will hold a request in a queue waiting for a connection to the DB Server to become available. Requests will typically be queued at times when all available DB Server connections are found to be in use.

- **Timeout for Inactive Connections**

The amount of time the SIG will keep an idle connection to a DB Server open before automatically closing it down.

- **Default DB Server**

The DB Server that will be used to serve client requests, in cases where a specific DB Server is not specified as part of the client's request.

- **Event Log Directive**

The parameters specified here controls what (and how much) information is written to the Event Log. The following two directives are currently available:

x – Record all software exceptions and other errors conditions.
i – Information – Verbose mode of operation.

Serious error conditions will always be written to the Event Log. The 'x' directive will instruct the SIG to provide more information about the errors. The 'i' directive will instruct the SIG to record information relevant to the progress of critical tasks, such as making new connections to DB Servers. The information captured is useful for diagnosing connectivity problems.

Both Event Log directives can be specified:

```
xi
```

5.2.2 Configuring Access to DB Servers

This option allows you to configure connectivity to specific DB Servers.

- **Service Status**

Use this facility for enabling or disabling access to an operational DB Server.

- **IP Address**

This is the IP address (logical or physical) of the DB Server.

- **TCP Port (for %zmgsi)**

This is the number of the TCP port on which the DB Servers is listening (usually 7041).

- **NameSpace**

This is the UCI (or InterSystems NameSpace) to be used on the DB Server.

- **User Name**

This is the User Name assigned to this DB Server for the purpose of authenticating the SIG.

- **Password**

This is the Password assigned to this DB Server for the purpose of authenticating the SIG.

- **Password (Confirm)**

Any Password entered above must be re-entered and verified here.

On the DB Server, the User Name and Password must be set as follows:

```
Set status=$$setunpw^%zmgsis("MyServiceUserName"," MyServicePassword")
```

The Username/Password is stored in the following global:

```
^%mgsi(0,"password")="dG9tY2F0"
^%mgsi(0,"username")="MyServiceUserName"
```

The password is obscured using B64 encoding on InterSystems DB Servers.

- **Maximum Number of Connections**

This is the maximum number of connections that the SIG installation can make to the DB Server. Under normal circumstances, the SIG will start as many connections to the DB Server as it needs in order to satisfy the current load. This parameter is used to prevent the SIG from taking over the DB Server during periods of high demand. Once the specified level of concurrent usage is reached, the SIG will automatically start queuing incoming requests.

- **NLS Translation Table Name**

This is the name of the InterSystems DB Server NLS (National Language Support) translation table to be applied to all communications between the Service Integration Gateway (and ultimately PHP) and the DB Server. This setting will be applied to the DB Server processes using the following function call:

```
$$SetIO^%NLS(tablename)
```

- **Optional Parameters**

The values for the parameters in this section will, by default, by inherited from the global DB Server configuration. These default values can be overridden here.

- **Failover and Load-Balancing**

In this section one or more DB Servers can be defined for the purpose of Load Balancing and/or Failover.

- **Alternative DB Servers**

This option allows you to define how Alternative DB Servers are to be used. There are three setting to choose from:

  - ***Use for Load-Balancing***: Use the DB Servers for balancing the load. The distribution of requests is made on a round-robin basis. Failover is implied with this option. If a DB Server becomes unresponsive it will be marked 'offline' and the load will be distributed to other DB Servers.
  - ***Use for Fail-Over only***: Use the Alternative DB Servers only if the primary DB Server becomes unresponsive. With this setting, if a DB Server becomes unresponsive it will be marked 'offline' and the load will be distributed to other DB Servers.
  - ***Disabled***: Do not use the Alternative Servers.

A list of Alternative DB Servers is defined, each of which can be marked as 'Enabled' or 'Disabled'. The SIG Management form will always present prompts for three Alternative Server. As these are filled, more slots will be presented as the configuration is saved and the form refreshed.

5.2.3 Testing Connections to the DB Server

Use this facility to test connection to a newly configured DB Server. It is recommended that you run this facility against newly created DB Server configurations to be sure that the connection is correctly configured.

5.2.4 Closing Connections to the DB Server

This option allows you to close down connections to all DB Servers or just an individual DB Server.

5.3 Configuring the SIG to respond to IBM MQ message queues.

This configuration suite defines how DB Servers can be configured to send messages to IBM MQ queues and also respond to request messages received by IBM MQ.

5.3.1 Configuration

This option allows you to check the value of the MQSERVER environment variable available to **mg_ibmmq**. This field must report a valid value otherwise the SIG will not be able to communicate with the target IBM MQ installation.

It is essential that the MQSERVER environment variable is globally defined on the computer hosting the SIG. The structure of this variable is as follows:

```
MQSERVER=ChannelName/TransportType/ConnectionName(SocketNumber)
```

For example, consider the following IBM MQ host configuration:

```
IBM MQ Channel name = S_dell3 (IBM MQ usually sets up a channel name of
the following form at installation time: S_<IBM_MQ_hostname>)

Transport type = TCP

Connection Name = dell3 (usually the name of the computer hosting IBM
MQ)

Socket Number = 1414 (the default TCP port number used by IBM MQ is
1414)
```

For these values, the MQSERVER variable should be set as follows:

```
MQSERVER=S_dell3/TCP/dell3(1414)
```

5.3.2 Configuring Services to IBM MQ

Use this option to configure each queue to be processed by **mg_ibmmq** (in the context of a server).

- **Service Name**

A user-defined name for the service. The service name will be used for management purposes within the SIG and will be used to identify the service in the SIG's systems status form. It must be made up of alphanumeric characters and be less than 32 characters in length.

- **Status**

The ability to listen for and process request messages arriving on a particular queue can be 'enabled' or 'disabled'. When an active service is disabled it will take up to sixty seconds for the service to gracefully disengage itself from the IBM MQ interface. When the service is closed it will be marked as 'Disabled' in the SIG's systems status form.

- **Queue Manager Name**

The name of the queue manager responsible for maintaining the queue to be used.

- **Queue Name**

The name of the queue from which request messages are expected.

- **Server**

The name of the DB Server to be used to process the request messages received. The server is selected from the drop-down list of configured DB Servers.

- **Server Routine**

The full name of the DB Server routine to be used to process the request messages received.

- **Maximum Array Size**

The maximum amount of message data that can be safely uploaded into local storage on the target DB Server. Messages that exceed the size specified here will be automatically uploaded into global (permanent) storage. Interrogate the global flag supplied to your server code to determine whether the message is available from local or global storage. It should be noted that this parameter only applies to messages received by the SIG operating as a server. The client functions described in the previous section should use their dedicated flag (request("global")) to specify whether or not global storage should be used.

6 Miscellaneous services provided by the SIG.

This section describes miscellaneous services provided by the SIG – notably access to cryptographic functions provided by the OpenSSL libraries, high-resolution time functions and Base-64 encoding/decoding.

6.1 OpenSSL functions

These functions are embedded in the SIG and provide access to cryptographic services provided by the OpenSSL library. It should be noted that modern M systems (such as recent versions of InterSystem Cache and IRIS) provide direct access to these services.

The OpenSSL libraries are usually pre-installed with many UNIX systems (notably Linux) but can be obtained via the following web site:

http://www.openssl.org/

Pre-built binaries for Windows can also be obtained via the Windows section of this site.

The SIG, when started, will attempt to bind to the OpenSSL libraries, which will usually be installed in the following locations:

Windows: ssleay32.dll and libeay32.dll in C:\windows\system32

UNIX: libssl.so and libcrypto.so in /usr/lib/

The Gateway will write the following message (or similar) to the Event Log if it is able to locate a working OpenSSL installation:

Windows:

```
Initialization
The SSL libraries (ssleay32.dll and libeay32.dll) are loaded - Version:
OpenSSL 0.9.8g 19 Oct 2007
```

UNIX:

```
Initialization
The SSL libraries (libssl.so and libcrypto.so) are loaded - Version:
OpenSSL 0.9.6b [engine] 9 Jul 2001
```

A failure message will be written if the OpenSSL libraries are either not present or unusable. The SIG will still operate under these circumstances but OpenSSL based services will, of course, not be available.

The following functions are provided. Set the 'b64' flag (1 or 0) depending on whether or not you wish the result to be Base-64 encoded. The final parameter (the context flag) should be set to 0.

Cryptographic hash: SHA256

```
Set result=$$sha256^%zmgsis(string,  b64, context)
```

Cryptographic hash: SHA1

```
Set result=$$sha1^%zmgsis(string,  b64, context)
```

Cryptographic hash: SHA

```
Set result=$$sha^%zmgsis(string,  b64, context)
```

Cryptographic hash: MD5

```
Set result=$$md5^%zmgsis(string,  b64, context)
```

Message Authentication: SHA256 based: HMAC-SHA256

```
Set result=$$hmacsha256^%zmgsis(string,  key,  b64, context)
```

Message Authentication: SHA1 based: HMAC-SHA1

```
Set result=$$hmacsha1^%zmgsis(string,  key,  b64, context)
```

Message Authentication: SHA based: HMAC-SHA

```
Set result=$$hmacsha^%zmgsis(string, key, b64, context)
```

Message Authentication: MD5 based: HMAC-MD5

```
Set result=$$hmacmd5^%zmgsis(string, key, b64, context)
```

Example: Generate a SHA256 HMAC using the SIG as the service provider and base-64 encode the result:

```
Set result=$$hmacsha256^%zmgsis("my data","my key",1,0)
```

6.1.1 Miscellaneous system functions

These functions are embedded in the SIG and do not require third party libraries.

Encode to Base-64

```
Set result=$$b64^%zmgsis(string, context)
```

Decode from Base-64

```
Set result=$$db64^%zmgsis(string, context)
```

Date and Time niladic functions:

Displayable Time to millisecond resolution:

```
Set result=$$time^%zmgsis(context)
```

Date and Time to millisecond resolution, formated in internal M form:

```
Set result=$$zts^%zmgsis(context)
```

7 License

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