

HandicapMaster[®] Software

Remote Request API and Integration - Guide

This document provides advice to developers wishing to use the Remote Request API to integrate HandicapMaster[®] with their Systems

Author: Stephen Jack
Last Updated: 9 June, 2009
Version: Issue 5 (for version 7.2 software)

© HandicapMaster Ltd 2009

Confidential

Contents

Contents.....	2
Introduction	4
Document Conventions.....	4
Per User data location.....	4
Document History.....	5
The Remote Request API	6
What is the Remote Request API?.....	6
Overview of the Remote Request API	6
Available Remote Request Facilities	7
Insert Member	7
Delete Member	7
Status of queued request	7
Using the Remote Request DLL File	8
Software requirements for running the Remote Request API DLL.....	8
Databases Supported by the Remote Request API DLL.....	9
Installation	10
Configuring Remote Request DLL	10
Example of calling DLL from Visual Basic	11
Using the Remote Request API Stored Procedures	12
Databases Supported by the Remote Request API Stored Procedures ..	12
Example of running the Stored Procedures.....	12
Request Format	13
XML Element Tags	13
JET 4.0 Database Message format examples	16
SQL SERVER Database Message format examples	18
Return Values.....	19
Format of messages returned (API DLL only)	20
Values for the RETURNCODE value	20
Housekeeping of requests queue.	21
Logging	21
HandicapMaster Log.....	22
DLL logs	22
Message Processing	22
Reading messages	22
Message Housekeeping.....	23
Stored procedures	24
Remote Request API Procedures	24
PROCEDURE: RemoteRequest_1	24
PROCEDURE: RemoteRequestStatus_1	25
Managing Membership Records.....	26
PROCEDURE: GetRecentMembers_1.....	26
PROCEDURE: SetMemberContactID_1	29
PROCEDURE: GetNationalNumber_1	30
Player Handicap Records.....	31
PROCEDURE: GetShortHcapRecord_1	31
PROCEDURE: GetHandicap_1.....	32
Player Handicap Certificate	34

PROCEDURE: GetHandicapCertificate_1.....	34
Managing Competition Fees	35
PROCEDURE: GetCompFeesPayable_1	35
PROCEDURE: SetCompFeesAllPaid_1.....	36
PROCEDURE: SetCompFeesPaid_1	37
PROCEDURE: GetCompFeesRefundable_1	38
PROCEDURE: SetCompFeesAllRefunded_1	39
PROCEDURE: SetCompFeesRefunded_1.....	40
SQL Database Security Requirements	41
Example configuration of Login in SQL Server	41
APPENDIX 1	42
Deprecated Message formats.....	42
JET 4.0 Database Message format examples	42
INSERT request (Jet 4.0 Database)	42
DELETE request (Jet 4.0 Database)	42
STATUS request (Jet 4.0 Database).....	43
SQL SERVER Database Message format examples.....	43
INSERT request (SQL Server Database)	43
DELETE request (SQL Server Database)	44
STATUS request (SQL Server Database).....	44

Introduction

This document is intended to provide information to customers with technical information to enable them to develop software within their applications to integrate with the facilities of HandicapMaster.

HandicapMaster reserve the right to update and modify the integration interface at any time.

This document contains an overview of the integration interface. This is followed by more technical content which is aimed at a technical audience including applications development staff.

Document Conventions

Per User data location

Within this document the location of per user files is indicated by the use of

“{UserDataDir}”

For Windows Vista please read this to be

`C:\Users\{User}\AppData\Roaming`

For Windows 2000 and XP please read this to be the Application Data path on Windows 2000/XP which is normally

`C:\Documents and Settings\{User}\Application Data`

On Windows 98/ME, this path is usually

`C:\WINDOWS\Application Data`

Within this document the location of application program files is indicated by the use of

“{ProgDir}”

For Windows default location please read this to be

`C:\Program Files\HandicapMaster7`

All trademarks acknowledged.

Document History

Issue 1

This is the initial version for integration with HandicapMaster 7.2.

Includes new tag for 'National Number', for EGU Central Database of Handicaps.

Issue 2

Added EMAILADDRESS tag to allow external system to set member's e-mail address in HandicapMaster.

Issue 3

Updated comments for NATIONALNUMBER tag with respect to English National system. Added 'GetNationalNumber_1' stored procedure.

Issue 4

Added column COMP_UNIQUE_ID to 'GetCompFeesPayable_1' and 'GetCompFeesRefundable_1' stored procedures.

Issue 5

Added 'SetCompFeesPaid_1' and 'SetCompFeesRefunded_1' stored procedures.

The Remote Request API

What is the Remote Request API?

The Remote Request API (Application Programme Interface) allows an external system to update details in HandicapMaster.

This document describes Version 2 of the Remote Request API. The facilities provided are limited to the insertion and deletion of membership details.

It is intended that future releases will support additional requests.

This interface has a number of advantages. These include:

- ◆ No manual intervention for its operation after the initial configuration
- ◆ Incremental changes to membership details instead of a “bulk load” (as implemented by the CSV method within HandicapMaster).

Thus it will provide a seamless means of integrating HandicapMaster with your existing applications.

In addition to the Remote Request API a number of stored procedures have been implemented to allow 3rd party applications to extract information from the HandicapMaster database.

Overview of the Remote Request API

The Remote Request API is a pre-programmed option in HandicapMaster and includes methods for your application to communicate with HandicapMaster.

There are two methods by which your application may communicate with HandicapMaster:

- a) An ActiveX DLL called “HM7RemoteRequest.DLL”, or
- b) Using Database Stored Procedures.

Your application may use the ActiveX DLL interface or the Stored Procedures to queue requests to HandicapMaster. These are then periodically processed by HandicapMaster. As such the interface is asynchronous. Multiple requests may be placed in the queue, and they will be processed in a First In, First Out (FIFO) manner.

A timer and queue limit is used to prevent a large number of requests being processed at the expense of the interactive use of HandicapMaster. For further details please see the section titled ‘Message Processing’

Available Remote Request Facilities

The following types of request will be processed by the Remote Request API.

Insert an “object” in to the HandicapMaster database
Delete an “object” from the HandicapMaster database
Status of a request previous queued.

Currently the only “object” supported is a “MEMBER”

Insert Member

This will either add a new member into the HandicapMaster database, or amend an existing member’s details.

Delete Member

This option will move a member to the Past Member’s list. This removes them from future competitions etc, but retains their historical data for handicapping purposes. Note: It is a CONGU requirement that the handicap history be retained for the current season and the previous 2 seasons. So the records are not deleted but placed in the “Past Members” list.

Status of queued request

This allows the 3rd Party application to determine the status of a successfully queued request.

If a request cannot be queued, the calling application will be immediately aware of this through the values returned from the call to the ActiveX DLL.

The 3rd Party application does not need to check the status of a previously queued request, but it may wish to do so to enhance the user experience.

Note: As the queuing and processing of requests is asynchronous there may be a delay between a request being queued, it being processed and the status of the queued request being updated. An immediate status check is not recommended as this will probably just indicate the request placed in the queue is still to be processed.

Using the Remote Request DLL File

If using the Remote Request DLL file, this must be installed alongside the 3rd Party applications on the Client's system. This system must have access to the HandicapMaster database, either located on the same system, or for Jet 4.0 (Access 2003) via Network shares in a Windows network environment, or via the network for SQL Server databases. To provide this database access the "HM7Database.DLL" is required.

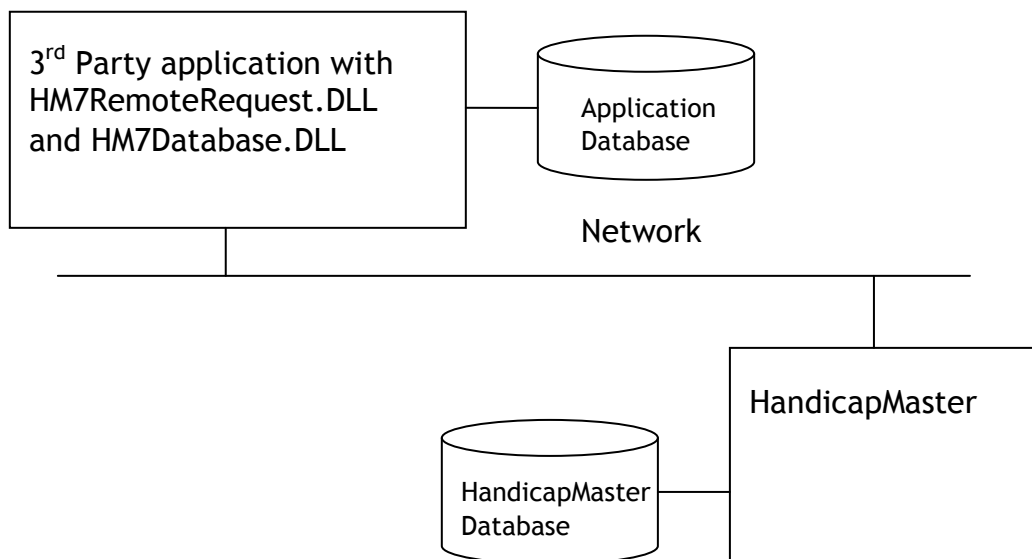
The "HM7RemoteRequest.DLL" may be optionally configured via an "INI" file. Details are provided below.

When a request is received by the Remote Request DLL, some initial validation is performed. If the request syntax is valid then it is placed in a queue for processing by HandicapMaster.

The 3rd Party application is then passed back the result of the attempt to queue the request. This is either the ID of the request in the queue, if the request syntax is valid, or error information if the request syntax is invalid.

The request ID can be used by the 3rd Party application to request status information. However this is not a requirement to using the interface.

The following diagram illustrates a typical installation:



Software requirements for running the Remote Request API DLL

The 3rd Party application will need to be able to make calls to the ActiveX DLL "HM7RemoteRequest.DLL". In addition the requests must be encoded in an XML format. Details of the message format are provided below.

The Remote Access DLL makes reference to the following HandicapMaster and Microsoft components, which MUST be available on the client computer:

HandicapMaster HM7Database DLL

Default file when HandicapMaster installed is

C:\Program Files\HandicapMaster7\HM7Database.dll

It is suggested that 3rd Party application package this with their application in addition to the "HM7RemoteRequest.DLL".

Microsoft ActiveX Data Objects 2.0 Library

(file \Program Files\Common Files\system\ado\msado20.tlb)

Microsoft XML, version 3.0

(file \Windows\System32\msxml3.dll)

(file \Windows\System\msxml3.dll for Windows 98 with I.E. 6 or better)

Databases Supported by the Remote Request API DLL

HandicapMaster (version 7 or greater) supports 2 different database engines, Jet 4.0 (Access 2002 / 2003) and SQL Server 2005.

This remote request facility can use either of these databases, however due to differences in the way these operate the configuration of the facility will be different.

Installation

The Remote Request facility (when using the API DLL file) requires the addition of the “HM7RemoteRequest.DLL” and “Hm7Database.DLL” into a folder where it can be accessed by the 3rd Party application. If an INI file is to be used for configuration, then it MUST be located in the “{UserDataDir}\HandicapMaster7” folder.

Configuring Remote Request DLL

The following entries may be included in the “HMRemoteRequest.INI”

INI file section “[HMRemoteRequest]”:

Key	Details	Jet Default Value	SQL Server Default Value
Catalog	If Jet 4.0 location of .mdb database file If SQL Server the Database Catalogue	Full path and filename of HandicapMaster database file. Note: this can be a network share.	Same value as used in DBconfig.ini file on system running HandicapMaster
DataSource	If Jet 4.0 location of .mdw security file If SQL Server the name of the database	Full path and filename of HandicapMaster database security file. Note: this can be a network share.	Same value as used in Dbconfig.ini file on system running HandicapMaster
DBDatabaseName			
DBServer			
Trace	Indicates whether debug trace mode is activated. Any value (other than “”) will set this. So by default the key should be missing. If enabled a log file called Rrtrace.txt will be created in the folder “{UserDataDir}\HandicapMaster7” This should only be enabled if requested by HandicapMaster Support as it will reduce performance.	(no default)	(no default)
ReportAll	Indicates whether successful Requests are reported in the log. Any value (other than “”) will set this. So by default the key is missing. The log file is located in the folder “{UserDataDir}\HandicapMaster7” and is called “HMRemoteRequest.log” Errors and warnings are always reported	(no default)	(no default)
Log	Full path and filename of log file used to report Errors and Warnings. If this key is not set then the log file is defaulted to {UserDataDir}\HandicapMaster7\	(no default)	(no default)

	HMRemoteRequest.log		
--	---------------------	--	--

NOTES:

- ◆ The INI file is not required to use the Remote Request API. In this case the DBDatabaseName and DBServer (or deprecated: Catalog and DataSource) values **MUST** be contained in the request sent from the 3rd Party application. Details of this are provided below.
- ◆ The XML tags DBDatabaseName and DBServer have **NO** equivalent in the INI file.

An example of an HMRemoteRequest INI file for a Jet 4.0 database with full logging enabled: (Windows XP)

```
[HMRemoteRequest]
Catalog= "C:\Documents and Settings\{user}\Application Data\HandicapMaster7\Database1\handicap.mdb"
DataSource= "C:\Program Files\HandicapMaster7\handicap.mdw"
Trace=True
ReportAll=True
Log=C:\Test\testlog.log
```

An example of an HMRemoteRequest INI file for a SQL Server database with full logging enabled:

```
[HMRemoteRequest]
Catalog= 192.168.0.100\SQLEXPRESS
DataSource= handicap
Trace=True
ReportAll=True
Log=C:\Test\testlog.log
```

Example of calling DLL from Visual Basic

The following simplified code snippet may assist programmers with calling the DLL

```
Public Sub LaunchRemoteAccessProgram(ByVal strMessage As String)

    ' strMessage contains the previously formatted XML request

    Dim objPE As Object
    Dim strRetText As String

    ...
    Set objPE = CreateObject("HM7RemoteRequest.clsRR_Main")

    objPE.MESSAGE_DATA = strMessage
    objPE.Main ' call dll

    ' get results and process them

    strRetText = objPE.RETURN_TEXT

    ...

End sub
```

Using the Remote Request API Stored Procedures

As an alternative to the ActiveX DLL method to communicate with HandicapMaster, your application may also communicate by calling Stored Procedures in the Database. These are detailed in the Stored Procedures section below.

Databases Supported by the Remote Request API Stored Procedures

The Stored Procedures are only available when HandicapMaster is running using an SQL Server database.

This option is NOT available for HandicapMaster Premier Edition.

Example of running the Stored Procedures

The following example shows how the RemoteRequest_1 procedure could be called to allow a request to be posted to HandicapMaster:

```
declare @Err int;
declare @RequestID int;
declare @Request xml;

SET @Request = '<?xml version="1.0" encoding="UTF-8" ?>
<HMRemoteRequest>
<RequestType>INSERT</RequestType>
<Object>MEMBER</Object>
<ACCOUNTID>4523</ACCOUNTID>
<FIRSTNAME>Joe</FIRSTNAME>
<SURNAME>Bloggs</SURNAME>
<GENDER>M</GENDER>
</HMRemoteRequest>';

Exec @Err = RemoteRequest_1 4, @Request, @RequestID OUTPUT;

SELECT @RequestID AS REQUEST_ID, @Err AS RESULT;
```

To check the status of this request, use RemoteRequestStatus_1 procedure:

```
declare @Err int;
declare @RequestID int;
declare @Status int;
declare @ErrText nvarchar(255);

-- Refer to Request 10
SET @RequestID = 10;

-- Get Status of Request
EXEC @Err = RemoteRequestStatus_1 4, @RequestID, @Status OUTPUT,
@ErrText OUTPUT;

-- Show Result
SELECT @Err AS RESULT, @Status AS STATUS, @ErrText AS ERROR_TEXT;
```

Request Format

Requests **MUST** be encoded using XML. Each message consists of a XML root element and a number of child data elements. Each data element has a tag associated with it.

For general details of XML see the following:

<http://www.w3schools.com/xml/default.asp>

NOTE: element tags are CASE sensitive.
The data elements are case insensitive.

In the messages a start tag is <tag> and closing tag is </tag> for clarity the <> will be omitted from these tables.

Not all child elements are required in every request type.

XML Element Tags

The following tags are used in all request types unless stated otherwise.

Element Tag	Description	Example value
HMRemoteRequest	XML root entity REQUIRED	n/a
RequestType	Type of request REQUIRED	INSERT,DELETE or STATUS
Object	Object the request is processing.	MEMBER is currently the only supported value This is a required value except for STATUS RequestType.
REQUESTID	the id of a previously issued request	Number > 0 ONLY required for STATUS RequestType messages
ACCOUNTID	The account ID or contact ID of the member	REQUIRED for INSERT or DELETE Generated by the 3rd Party software. MUST be unique for each member Text up to 50 chars

The following tags are required when using the API DLL (if the details are not included in the INI file) but are not required when using the API Stored Procedures.

Element Tag	Description	Example value
DBDatabaseName	REQUIRED unless using INI file SQL Server: Database Instance Name Jet 4.0: Database file	SQL Server: handicap Jet 4.0 {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
DBServer	REQUIRED unless using INI file SQL Server: Database Server Jet 4.0: Database workgroup access file	SQL Server: 192.168.0.100\SQLEXPRESS Jet 4.0 {ProgDir}\HandicapMaster7\handicap.mdw

The following tags are supported, but are deprecated and retained for backwards compatibility only. Both of these are REQUIRED if using the API DLL (and details not included in the INI file). Not required if using the Stored Procedures.

Catalog	SQL Server: Database catalog Jet 4.0: Database file file	SQL Server: 192.168.0.100\SQLEXPRESS Jet 4.0 {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
DataSource	SQL Server: Database Instance Name Jet 4.0: Database Security file	SQL Server: handicap Jet 4.0 {ProgDir}\HandicapMaster7\handicap.mdw

The following Tags are ONLY used for the **INSERT** RequestType

Element Tag	Description	Example value and format details
FIRSTNAME	First name of member REQUIRED	Text up to 20 chars
SURNAME	Surname of member REQUIRED	Text up to 20 chars
MIDDLEINITIALS	Middle initials of member	Text up to 10 chars
TITLE	Title of Member	Text up to 6 chars
GENDER	Gender of member	“M” for male or “F” for female are the only supported values. Defaults to male gender if not supplied
DATEOFBIRTH	The member’s date of birth	Date format using locale
CATEGORY	Category of membership	Text up to 40 chars
CARDNUMBER	Member’s card number	Text up to 16 chars
MEMBERNUMBER	Members membership number	Text up to 10 chars
ADDRESS1	Line 1 of member’s address	Text up to 40 chars
ADDRESS2	Line 2 of member’s address	Text up to 40 chars
ADDRESS3	Line 3 of member’s address	Text up to 40 chars
ADDRESS4	Line 4 of member’s address	Text up to 40 chars
ADDRESS5	Postcode or Line 5 of member’s address	Text up to 10 chars
PHONE1	Member’s 1 st phone number	Text up to 20 chars
PHONE2	Member’s 2 nd phone number	Text up to 20 chars
PHONE3	Member’s 3rd phone number	Text up to 20 chars
LOCKERNO	Member’s Locker Number	Text up to 10 chars

PIN	Member's PIN Number	Text up to 6 chars
GOLFNETNUMBER or NATIONALNUMBER	Ireland GOLFnet card Number, or EGU CDH Lifetime Number	Always 8 digits for Ireland clubs. Always 10 digits for England clubs. NOTE: this value will be ignored for English Clubs as HandicapMaster is expected to interrogate the English national system and obtain a number for a member.
EMAILADDRESS	Email address of member	Text up to 50 characters. Validation of the e-mail address format is made, if the validation fails the e-mail address will not be updated. A warning message is logged if validation fails.

Note if data exceeds the length of the field it will be truncated to the maximum length.

There are currently 3 RequestType messages supported

INSERT
DELETE
STATUS

All message types **MUST** either contain the <DataSource> <Catalog> tags OR suitable values **MUST** be placed in the INI file if using the API DLL. Values for these tags in the request **ALWAYS** override any settings in the INI file. These 2 data elements are used by the HM7RemoteRequest DLL to access the database.

The examples below illustrate sample messages for Jet 4.0 Databases and then SQL Server databases

JET 4.0 Database Message format examples

These are using default locations which may be different in your configuration.

See Appendix 1 for examples of messages that contain the deprecated XML tags <DataSource> and <Catalog>

INSERT request (Jet 4.0 Database using API DLL)

```
<HMRemoteRequest>
  <DBServer>
    {ProgDir}\HandicapMaster7\HandicapMaster.mdw
  </DBServer >
  <DBDatabaseName>
    {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
  </DBDatabaseName >
  <RequestType>
    INSERT
```

```
</RequestType>
<Object>
  MEMBER
</Object>
<ACCOUNTID>
  4523
</ACCOUNTID>
<FIRSTNAME>
  Joe
</FIRSTNAME>
<SURNAME>
  Bloggs
</SURNAME>
```

And so on etc

```
</HMRemoteRequest>
```

DELETE request (Jet 4.0 Database using API DLL)

Only the ACCOUNTID is required to delete a member

```
<HMRemoteRequest>
  <DBServer>
    {ProgDir}\HandicapMaster7\HandicapMaster.mdw
  </DBServer>
  <DBDatabaseName>
    {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
  </DBDatabaseName>
  <RequestType>
    DELETE
  </RequestType>
  <Object>
    MEMBER
  </Object>
  <ACCOUNTID>
    4523
  </ACCOUNTID>
</HMRemoteRequest>
```

STATUS request (Jet 4.0 Database using API DLL)

```
<HMRemoteRequest>
  <DBServer>
    {ProgDir}\HandicapMaster7\HandicapMaster.mdw
  </DBServer>
  <DBDatabaseName>
    {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
  </DBDatabaseName>
  <RequestType>
    STATUS
  </RequestType>
  <REQUESTID>
    23
  </ REQUESTID >
</HMRemoteRequest>
```

SQL SERVER Database Message format examples

INSERT request (SQL Server Database using Stored Procedures)

```
<HMRemoteRequest>
  <RequestType>
    INSERT
  </RequestType>
  <Object>
    MEMBER
  </Object>
  <ACCOUNTID>
    4523
  </ACCOUNTID>
  <FIRSTNAME>
    Joe
  </FIRSTNAME>
  <SURNAME>
    Bloggs
  </SURNAME>
```

And so on etc

```
</HMRemoteRequest>
```

DELETE request (SQL Server Database using Stored Procedures)

Only the ACCOUNTID is required to delete a member

```
<HMRemoteRequest>
  <RequestType>
    DELETE
  </RequestType>
  <Object>
    MEMBER
  </Object>
  <ACCOUNTID>
    4523
  </ACCOUNTID>
</HMRemoteRequest>
```

STATUS request (SQL Server Database using API DLL)

```
<HMRemoteRequest>
  <DBServer>
    192.168.1.2\SQLEXPRESS
  </DBServer>
  <DBDatabaseName>
    handicap
  </DBDatabaseName>
  <RequestType>
    STATUS
  </RequestType>
  <REQUESTID>
    23
  </ REQUESTID >
</HMRemoteRequest>
```

Return Values

The Remote Access DLL returns to the caller the result of trying to place the request in the queue. It is strongly recommended that the calling application check this return information.

Note: this return information is DIFFERENT from a status request for a previously queued request.

Return information is encoded using XML. Each message consists of a XML root element and a number of child data elements. Each data element has a tag associated with it.

There are 3 message formats for the data being returned from the DLL. The following XML tags are supported:

Element Tag	Description	Example value
HMRemoteRequest	XML root entity ALWAYS returned	n/a
RETURNCODE	16 bit (signed integer) numeric encoded as text, indicates the status of the request ALWAYS returned	< 0 if an error trying to queue the request 0 If INSERT or DELETE request queued successfully. > 0 FOR STATUS request, the status of a request in the queue (see table below for possible values)
REQUESTID	The ID of the request placed in the queue. 32 bit (long) Numeric encoded as string	> 0 ONLY returned if an INSERT or DELETE request successfully queued
ERRORTXT	Error message if request not queued	Only returned if RETURNCODE < 0 i.e. error processing request
STATUSTXT	Status message	Text message containing status of request, only for STATUS requests Is "" (null) if no error

To summarise

RETURN_CODE = 0 If request queued successfully unless a status request when > 0
 < 0 if an error trying to process the request
 > 0 the status of a request in the staging table (only if you have issued a STATUS request)

Format of messages returned (API DLL only)

Error trying to queue the request i.e. RETURNCODE < 0

```
<HMRemoteRequest>
  <RETURNCODE>value</RETURNCODE>
  <ERRORTXT>processing error message text</ERRORTXT>
</HMRemoteRequest>
```

Request queued successfully i.e. RETURNCODE = 0

```
<HMRemoteRequest>
  <RETURNCODE>value</RETURNCODE>
  <REQUESTID>request id number</REQUESTID>
</HMRemoteRequest>
```

Result of STATUS request i.e. RETURNCODE > 0

```
<HMRemoteRequest>
  <RETURNCODE>status code of the request</RETURNCODE>
  <STATUSTEXT>status message text</STATUSTEXT>
</HMRemoteRequest>
```

Values for the RETURNCODE value

The following table describes the value of the RETURNCODE data element when the STATUS of the request in the queue is requested.

RETURNCODE Value	Description
1	Request in queue waiting to be processed
2	Request being actioned
3	Request successfully processed (no error) STATUSTEXT is ""
4	Error Processing request. Detail in STATUSTEXT

Housekeeping of requests queue.

HandicapMaster (when it is running) will periodically housekeep the Remote Request queue.

It will delete any requests that are more than 7 days old in the queue, regardless of their status.

Note on starting up HandicapMaster will process any outstanding requests, even if they have been queued more than 7 days.

Logging

There are a number of logs produced by this API and HandicapMaster. These logs are in general of little interest, they only become relevant when investigating possible issues with the interface.

One of the logs may be turned on at the request of HandicapMaster support. Another log can be configured to only display errors and warnings.

HandicapMaster Log

HMRRImport.log

Located in folder “{UserDataDir}\HandicapMaster7”, this log file contains details of the Remote requests processed by HandicapMaster. No configuration of this log is possible.

This log file will contain details of any insertion and deletion of members. This is probably the first log to examine to determine if a specific user has been processed successfully.

The log file is rotated, once it reaches a size of 512KBytes the original file is moved to a file with .old as the extension.

i.e. HMRRImport.log.old

If the “.old” file is present it is removed prior to the rotation of the log file.

DLL logs

HMRRemoteRequest.log

Located in folder “{UserDataDir}\HandicapMaster7”, this log file contains errors or warnings that are produced as the DLL processes requests.

It is also possible to turn on informational messages that will log details of successfully processed remote requests.

To change the location and name of this log file using an INI key value. The log file is rotated, once it reaches a size of 512KBytes the original file is moved to a file with .old as the extension.

i.e. HMRRemoteRequest.log.old

If the “.old” file is present it is removed prior to the rotation of the log file.

RRtrace.txt

Located in “{UserDataDir}\HandicapMaster7” this is only created/appended to if the appropriate INI file key is set.

This log file will only be of use to the developers of the DLL.

This log file is always appended to, there is NO housekeeping.

Only turn this on if requested to by HandicapMaster support.

Message Processing

Reading messages

This section describes the manner in which message requests are processed to prevent large number of messages impacting the user’s experience of using HandicapMaster interactively.

Messages received by the ActiveX DLL interface queues requests within a table within the HandicapMaster database.

A timer is activated every 30 seconds to process any requests that are queued. At this point up to 5 messages are processed by HandicapMaster and the appropriate status is set.

This process is repeated every 30 seconds.

There are 2 exceptions to the processing of messages in the queue.

1. When HandicapMaster is started.

On the computer that is configured to run the Remote Request interface when HandicapMaster is started all the messages in the queue that have not yet been processed will be read and processed. This is useful when initially setting up the interface as a significant number of messages may have been queued.

Note: ALL messages that were queued at some time in the past are processed before any housekeeping is performed.

After the initial run the timer activation and queue processing continues.

2. User manually activates the queue processing.

When the 'Load Membership Records' On the PLAYERS menu in HandicapMaster is selected then a maximum of 5 messages are immediately processed.

Message Housekeeping

When messages in the queue are older than 7 days they are removed from the queue regardless of the success or failure of the message to be processed. Prior to this time the 3rd part application may request the status of a queued request. This housekeeping is activated once every 30 minutes within HandicapMaster.

Stored procedures

This section of the document describes the stored procedures that have been implemented to complement the Remote Request facility.

Remote Request API Procedures

PROCEDURE: RemoteRequest_1

Parameters:

ParamVersion INT,
ParamMessage XML,
ParamRequestID INT

IN The Version of the interface being used. Must be value 4 when following this edition of the manual.
IN The request formatted in XML format as previously described.
OUT The allocated Request ID for the request submitted

Version compatibility

Available from Version 7.1.14 of HandicapMaster onwards

Databases supported

SQL Server Express (2005 or 2008)

Expected Use:

Enable 'Remote Requests' to be delivered to HandicapMaster.

Output

In addition to "ParamRequestID" parameter described above, this procedure will return an Error Code.

Error Code	Description
0	No Error
-5	REQUEST TYPE not recognised in request
-8	Version of Interface too low
-9	Version of Interface too high
-102	OBJECT not recognised in request

PROCEDURE: RemoteRequestStatus_1

Parameters:

ParamVersion INT,
 ParamRequestID INT,
 ParamStatus INT,
 ParamErrorText NVARCHAR(255)

IN The Version of the interface being used. Must be value 4 when following this edition of the manual.
 IN The Request ID for the request being queried
 OUT The Status of the request
 OUT Text returned from any Error encountered by HandicapMaster processing the request.

Version compatibility

Available from Version 7.1.14 of HandicapMaster

Databases supported

SQL Server Express (2005 or 2008)

Expected Use:

To query the status of any request previously posted to HandicapMaster.

Output

In addition to “ParamStatus” and “ParamErrorText” parameters described above, this procedure will return an Error Code.

Error Code	Description
0	No Error
-8	Version of Interface too low
-9	Version of Interface too high

Managing Membership Records

PROCEDURE: GetRecentMembers_1

Parameters:

ParamLastAmendmentDT datetime

IN a date value indicating the data and time after which changes to member details are required

Version compatibility

Available from Version 7.0.28 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

This query is typically used when the membership details are managed from within HandicapMaster.

3rd party application to periodically query the HandicapMaster database.

3rd party application to note the time of query, and then at a later time query any changes that are after the previous query time and use the information to maintain list of members.

Output

The following table will help to interpret some of the values in the fields.

The important one is HANDICAP STATUS as this will allow you to decide if a member has been deleted, i.e. the value = 0 (handicap deleted). This is set when a member is set to a past member within HandicapMaster.

FIELD	Description	Type	Size
LAST_AMENDMENT_DT	The date and time this member's record was last updated	date	
SURNAME	Member's surname	string/char	20
CHRISTIAN NAME	Members first name	string/char	20
MIDDLE INITIALS	Member's initials	string/char	10
TITLE	Member's title	string/char	6
MEMBER ID	HandicapMaster internal member ID. This can be reused so cannot be relied to be Unique!	Integer	
IS HOME CLUB	Flag to indicate if HOME player FALSE if AWAY player	Boolean	
HANDICAP STATUS	The status of the Members handicap: 0 = Handicap deleted 1 = Valid Handicap	Integer	

	2 = suspended Handicap 3 = lapsed Handicap 5 = no handicap (social member)		
CURRENT EXACT HANDICAP	The exact handicap for member	single	
CURRENT PLAYING HANDICAP	The playing handicap	integer	
GENTLEMAN	True if Male, False if female	Boolean	
TELEPHONE	Member's telephone number	string/char	20
CLUB_MEMBER_NUMBER	Club membership Number	string/char	10
ADDRESS_1	1 st line of address	string/char	40
ADDRESS_2	2nd line of address	string/char	40
ADDRESS_3	3rd line of address	string/char	40
ADDRESS_4	4th line of address	string/char	40
POST_CODE	Post code	string/char	10
DATE_JOINED	Date the member joined, may be null	date	
DATE_RENEWED	Date the Member's subscription was renewed if there are any subscriptions for the member	date	
DATE_EXPIRES	Date the Member's subscription will(or has) expired	date	
TELEPHONE2	2 nd Telephone number	string/char	20
TELEPHONE3	3rd Telephone number	string/char	20
EMAIL_ADDR	email address	string/char	50
NATIONAL_ID HOLDER	True for GOLFnet card holders (Ireland only) Currently False for all other regions.	Boolean	
NATIONAL_NUMBER	GOLFnet number (Ireland only)		
DATE_SUBSCRIPTION_PAID	Date a subscription payment has been made. Note there may be multiple payments over the year so this can change though the year.	date	
PIN_NUMBER	Player's HandicapMaster PIN number	Long	
PLAYER_HOLDS_CARD	TRUE if member has a card number. Can be TRUE even if GOLFnet card when using magnetic swipe reader instead of smart card reader.	Boolean	
CARD_NUMBER	Member card number	string/char	16
DATE_OF_BIRTH	The Member's Date of Birth (may be null)	date	
AGE_GROUP	The age group the member is in. Valid values are Adult = 1 Junior = 2 (age 18 or less)	long	

	Senior = 4 (age 65 or over by default, but can be changed by user) Other1 = 8 Other2 = 16 Other1 and Other2 are user defined age group names.		
--	--	--	--

PROCEDURE: SetMemberContactID_1

Parameters:

ParamContactID text
ParamMemberID integer

- IN The CONTACT_ID in the external system that is used to identify a member
- IN The MEMBER_ID in the HandicapMaster system that is used to identify a member

Version compatibility

Available from Version 7.0.28 of HandicapMaster

Databases supported

Jet 4.0 (Access)
SQL Server Express (2005 or 2008)

Expected Use:

This stored procedure allows a 3rd party to set the CONTACT_ID of MEMBER record. This may be required when a new member is added within HandicapMaster and 3rd party is getting its list of members from HandicapMaster using stored procedure GetRecentMembers_1.

Using this stored procedure will allow 3rd party applications to set the CONTACT_ID which initially will be 0 and this can then be used by the 3rd Party application in the next call to stored procedure GetRecentMembers_1 to identify if the record has been already added into its database.

NOTE: Although in HandicapMaster the [MEMBER ID] can be 'potentially' reused, if this stored procedure is used immediately after GetRecentMembers_1 then there is almost no chance of this happening.

Output

The CONTACT_ID of the member with the supplied MEMBER_ID will be updated.

PROCEDURE: GetNationalNumber_1

Parameters:

ParamContact_ID NVARCHAR(50)

IN The CONTACT_ID in the external system that is used to identify a member

Version compatibility

Available from Version 7.2.15 of HandicapMaster

Databases supported

SQL Server Express (2005 or 2008)

Expected Use:

Provide 3rd party application with current National Number (GOLFnet number in Ireland, EGU number in England, etc) for the specified player.

Output

This procedure will return a Status Code as an Integer:

Status Code	Description
0	No Error, valid handicap information returned
-2	CONTACT_ID not found, no information available.
-3	This player does not currently have a valid membership record (is a past member or visitor).

In addition the following 2 items are returned as part of the Stored Procedure:

FIELD	Description	Type	Size
STATUS_TEXT	Text message indicating status of request	NVARCHAR	255
NATIONAL_NUMBER	Member's National Number	INTEGER	

NOTE if the Status Code is less than 0 then the values returned are NULL as no information is available.

The STATUS_TEXT may be one of the following

STATUS TEXT	Status Code
Player holds a National Number	0
Player does not hold a National Number	0
CONTACT_ID not found, no information available.	-2
This player is not a current player.	-3

Player Handicap Records

PROCEDURE: GetShortHcapRecord_1

Parameters:

ParamContact_ID NVARCHAR(50)

IN The CONTACT_ID in the external system that is used to identify a member

Version compatibility

Available from Version 7.0.28 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Provide 3rd party application with short form of handicap record.

Output

The following table will help to interpret some of the values in the fields.

FIELD	Description	Type	Size
DATE	The date and time this item was added to the member's handicap record	date	
COMPETITION	description of the item in the handicap record includes such items as 'Handicap Brought Forward' etc.	string/char	40
GROSS SCORE	The member's Gross Score	Integer	
STAB/PAR SCORE	The member's Stableford or Par Score	Integer	
NETT DIFFERENTIAL	Handicap nett differential	integer	
HANDICAP ADJUSTMENT	Amount the handicap has been adjusted	single	
REVISED HANDICAP	New playing handicap	single	
MEMBER_ID	HandicapMaster member ID	Integer	

PROCEDURE: GetHandicap_1

Parameters:

ParamContact_ID NVARCHAR(50)

IN The CONTACT_ID in the external system that is used to identify a member

Version compatibility

Available from Version 7.1.31 of HandicapMaster

Databases supported

SQL Server Express (2005 or 2008)

Expected Use:

Provide 3rd party application with current Playing and Exact handicap of member for use in internal reporting and managing membership.

NOTE: Will NOT return values for Visitors, if a request is made for a Visitor's handicap the Status code -3 will be returned

Output

This procedure will return a Status Code as an Integer:

Status Code	Description
0	No Error, valid handicap information returned
-1	Handicap Not Available from this Edition of HandicapMaster
-2	CONTACT_ID not found, no handicap information available.
-3	This player does not currently have a Handicap Award.

In addition the following 6 items are returned as part of the Stored Procedure:

FIELD	Description	Type	Size
STATUS_TEXT	Text message indicating status of request	NVARCHAR	255
PLAYING_HANDICAP	Members Playing Handicap	INTEGER	
EXACT_HANDICAP	Members Exact Handicap	SINGLE	
IS_HOME_PLAYER	=1 if Player is Home Player, =0 if Player is Away Player.	BIT	
IS_STARRED	=1 if Handicap is Lapsed, =0 otherwise	BIT	
IS_DISABILITY_HANDICAP	=1 if Handicap is a Disability Handicap, =0 otherwise	BIT	

NOTE if the Status Code is less than 0 then the values returned are NULL as no valid handicaps are available.

The STATUS_TEXT may be one of the following

STATUS TEXT	Status Code
AWAY MEMBER CONGU® HANDICAP	0
HOME MEMBER CONGU® HANDICAP	0
CLUB HANDICAP	0
DISABILITY GOLF HANDICAP	0
Handicap Not Available from this Edition of HandicapMaster.	-1
CONTACT_ID not found, no handicap information available.	-2
This player does not currently have a Handicap Award.	-3

Conditions of use of this Procedure:

This information MUST NOT be used to generate Handicap Certificates or other CONGU® licensed reports.

Note: You are not licensed to use this procedure unless the conditions of use above are acknowledged and respected.

Player Handicap Certificate

PROCEDURE: **GetHandicapCertificate_1**

Parameters:

ParamContact_ID NVARCHAR(50)

IN The CONTACT_ID in the external system that is used to identify a member

Version compatibility

Available from Version 7.1.27 of HandicapMaster

Databases supported

SQL Server Express (2005 or 2008)

Expected Use:

Provide the wording required to produce a CONGU-compliant handicap certificate for the requested player.

Output:

Fields LINE_1, LINE_2, ... LINE_10 are generated. These are the paragraphs of text that must be placed on any Handicap Certificate that is created for the player.

FIELD	Description	Type	Size
LINE_xx	The date and time this item was added to the member's handicap record	NVARCHAR (255)	

Conditions of use of this Procedure:

It is understood that any system calling this procedure will display the paragraphs of text in the order that they are returned by the procedure (LINE_1 first, LINE_2 second, etc).

The text generated by the procedure **MUST NOT** be amended before being displayed on the Certificate.

No additional information may be displayed on the certificate.

Note: You are not licensed to use this procedure unless the conditions of use above are acknowledged and respected.

Managing Competition Fees

These Procedures are supplied to help manage competition fees collected by HandicapMaster to be processed by the 3rd Party Application.

PROCEDURE: **GetCompFeesPayable_1**

Parameters:

None

Version compatibility

Available from Version 7.1.10 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Gets a list of Competition Fees awaiting payment

Output

The following table will help to interpret some of the values in the fields.

FIELD	Description	Type	Size
SURNAME	Member's surname	string/char	20
CHRISTIAN NAME	Members first name	string/char	20
MIDDLE INITIALS	Member's initials	string/char	10
CONTACT_ID	Internal Number	string/char	
CLUB_MEMBER_NUMBER	Club membership Number	string/char	10
COMP_DATE	Date of Competition	Date	
COMP_NAME	Name of Competition	string/char	40
FEE_DESCRIPTION	Description of Fee payable	string/char	25
ACCOUNTING_CODE	Accounting Code of Fee	string/char	8
FEE_PAYABLE	Fee payable	Currency	
COMP_UNIQUE_ID	Unique Number of Competition	Long Integer	

PROCEDURE: SetCompFeesAllPaid_1

Parameters:

None

Version compatibility

Available from Version 7.1.10 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Marks all outstanding Competition Fees as paid.

Output

None.

PROCEDURE: SetCompFeesPaid_1

Parameters:

ParamCompetitionID INT

IN The COMP_UNIQUE_ID returned from a call to
'GetCompFeesPayable_1'

Version compatibility

Available from Version 7.2.32 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Marks all outstanding Competition Fees as paid for a specified competition.

Output

None.

PROCEDURE: GetCompFeesRefundable_1

Parameters:

None

Version compatibility

Available from Version 7.1.10 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Gets a list of Competition Fees awaiting refund.

Output

The following table will help to interpret some of the values in the fields.

FIELD	Description	Type	Size
SURNAME	Member's surname	string/char	20
CHRISTIAN NAME	Members first name	string/char	20
MIDDLE INITIALS	Member's initials	string/char	10
CONTACT_ID	Internal Number	string/char	
CLUB_MEMBER_NUMBER	Club membership Number	string/char	10
COMP_DATE	Date of Competition	Date	
COMP_NAME	Name of Competition	string/char	40
FEE_DESCRIPTION	Description of Fee payable	string/char	25
ACCOUNTING_CODE	Accounting Code of Fee	string/char	8
FEE_PAYABLE	Fee payable	Currency	
COMP_UNIQUE_ID	Unique Number of Competition	Long Integer	

PROCEDURE: SetCompFeesAllRefunded_1

Parameters:

None

Version compatibility

Available from Version 7.1.10 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Marks all Competition Fees shown as awaiting refund as refunded.

Output

None.

PROCEDURE: SetCompFeesRefunded_1

Parameters:

ParamCompetitionID INT

IN The COMP_UNIQUE_ID returned from a call to
'GetCompFeesPayable_1'

Version compatibility

Available from Version 7.2.32 of HandicapMaster

Databases supported

Jet 4.0 (Access)

SQL Server Express (2005 or 2008)

Expected Use:

Marks all Competition Fees shown as awaiting refund as refunded for a specified competition.

Output

None.

SQL Database Security Requirements

To be able to operate the Stored Procedures with an SQL Server Database (HandicapMaster Network edition) it is necessary to set up a suitable login to SQL Server with access to the database.

Example configuration of Login in SQL Server

Creates a Login using SQL Server Authentication:

1. In SQL Server Management Studio, select NEW QUERY from the Toolbar.
2. Enter the following:

```
CREATE LOGIN <UserName> WITH PASSWORD = '<Password>';  
USE <DatabaseName>;  
CREATE USER <UserName> FOR LOGIN <UserName>;  
GRANT SELECT,UPDATE,EXECUTE TO <UserName>;
```

Where <UserName>, <Password> and <DatabaseName> are choices for your system.

3. Click “! Execute” to apply the User to the SQL Server environment.

Example

To add a user called “MyUser” with password “MyPassword” to a database called “HandicapMaster”, enter

```
CREATE LOGIN MyUser WITH PASSWORD = 'MyPassword';  
USE HandicapMaster;  
CREATE USER MyUser FOR LOGIN MyUser;  
GRANT SELECT,UPDATE,EXECUTE TO MyUser;
```

This Login will now have sufficient access to run the HandicapMaster stored procedures.

APPENDIX 1

Deprecated Message formats

This section details examples of messages using the deprecated XML tags DataSource and Catalog

JET 4.0 Database Message format examples

These are using default locations which may be different in your configuration.

INSERT request (Jet 4.0 Database)

```
<HMRemoteRequest>
  <DataSource>
    {ProgDir}\HandicapMaster7\HandicapMaster.mdw
  </DataSource>
  <Catalog>
    {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
  </Catalog>
  <RequestType>
    INSERT
  </RequestType>
  <Object>
    MEMBER
  </Object>
  <ACCOUNTID>
    4523
  </ACCOUNTID>
  <FIRSTNAME>
    Joe
  </FIRSTNAME>
  <SURNAME>
    Bloggs
  </SURNAME>
```

And so on etc

```
</HMRemoteRequest>
```

DELETE request (Jet 4.0 Database)

Only the ACCOUNTID is required to delete a member

```
<HMRemoteRequest>
  <DataSource>
    {ProgDir}\HandicapMaster7\HandicapMaster.mdw
  </DataSource>
  <Catalog>
    {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
  </Catalog>
  <RequestType>
```

```
        DELETE
    </RequestType>
    <Object>
        MEMBER
    </Object>
    <ACCOUNTID>
        4523
    </ACCOUNTID>
</HMRemoteRequest>
```

STATUS request (Jet 4.0 Database)

```
<HMRemoteRequest>
    <DataSource>
        {ProgDir}\HandicapMaster7\HandicapMaster.mdw
    </DataSource>
    <Catalog>
        {UserDataDir}\HandicapMaster7\Database1\handicap.mdb
    </Catalog>
    <RequestType>
        STATUS
    </RequestType>
    <REQUESTID>
        23
    </ REQUESTID >
</HMRemoteRequest>
```

SQL SERVER Database Message format examples

INSERT request (SQL Server Database)

```
<HMRemoteRequest>
    <DataSource>
        handicap
    </DataSource>
    <Catalog>
        192.168.1.2\SQLEXPRESS
    </Catalog>
    <RequestType>
        INSERT
    </RequestType>
    <Object>
        MEMBER
    </Object>
    <ACCOUNTID>
        4523
    </ACCOUNTID>
    <FIRSTNAME>
        Joe
    </FIRSTNAME>
    <SURNAME>
        Bloggs
    </SURNAME>
```

And so on etc

```
</HMRemoteRequest>
```

DELETE request (SQL Server Database)

Only the ACCOUNTID is required to delete a member

```
<HMRemoteRequest>
  <DataSource>
    handicap
  </DataSource>
  <Catalog>
    192.168.1.2\SQLEXPRESS
  </Catalog>
  <RequestType>
    DELETE
  </RequestType>
  <Object>
    MEMBER
  </Object>
  <ACCOUNTID>
    4523
  </ACCOUNTID>
</HMRemoteRequest>
```

STATUS request (SQL Server Database)

```
<HMRemoteRequest>
  <DataSource>
    handicap
  </DataSource>
  <Catalog>
    MYSERVER\SQLEXPRESS
  </Catalog>
  <RequestType>
    STATUS
  </RequestType>
  <REQUESTID>
    23
  </ REQUESTID >
</HMRemoteRequest>
```