HTTPO$ - Open Communications to HTTP Server

The HTTPO$ routine is used to open a communications link to an HTTP server.

1. Invocation

To open a connection to an HTTP server code:

```
CALL HTTPO$ USING ho
```

where *ho* is a control block of the following format:

```
01    HO
  02  HOVERS      PIC 9(4) COMP     * BLOCK VERSION NUMBER
                                    * MUST BE 1
  02  HOMTHD      PIC X(20)         * HTTP method
  02  HOPURL      PIC PTR           * Pointer to destination URL
  02  HOASYN      PIC 9(2) COMP     * Async mode & data type flag (see below)
  02  HOUSER      PIC X(50)         * User name
  02  HOPASS      PIC X(50)         * Password
  02  HOHDL       PIC 9(4) COMP     * HTTP session handle
```

2. STOP Codes and Exception Conditions

The following STOP codes may be generated by HTTPO$:

STOP code	Description
13609	HTTPO$ has been called by an application that is not running on GX.
25022	An attempt has been made to call HTTPO$ on an incompatible version of GX. The version of GX must be V3.5x, or later
13610	Invalid Control Block version
13611	The total length of the GX command block has exceeded an internal limit.

The following EXIT codes may be returned by HTTPO$:

EXIT code	$$COND	Description
13608	08	Unable to allocate memory for temporary work buffer.
13609	09	An error condition was returned by the open communications operation. The error code has been returned in $$CRES.

3. Programming Notes

HTTPO$ is only available when running on GX. Any attempt to use HTTPO$ on a non-GX terminal will result in a STOP code. The version of GX must be V3.5x or later. The version of GSM must be GSM SP-17, or later.

The HOPURL field must point to a zero-terminated string that specifies the destination URL.

HOMTHD, HOUSER and HOPASS are all normal character strings (i.e. fixed length with trailing SPACE characters). However, they are converted to zero terminated strings by the sub-routine that passes the strings to GX.

The HOASYN field is a collection of bit-flags:

Bit-0	0 = synchronous operation
	1 = asynchronous operation
Bit-1	0 = XML data
	1 = Non XML data
Bit-2	Reserved for future use
Bit-3	Reserved for future use
Bit-4	Reserved for future use
Bit-5	Reserved for future use
Bit-6	Reserved for future use
Bit-7	Reserved for future use

A successful call to HTTPO$ will return an HTTP session handle in HOHDL. This session handle must be used by subsequent HTTPH$, HTTPS$ and HTTPC$ calls.

4. Examples

```
MOVE "POST" TO HOMTHD
MOVE "https://secure.dev.gateway.gov.uk/submission" TO X-STRING
* The code also needs to zero-terminate this string
POINT HOPURL AT X-STRING
MOVE 0 TO HOASYN
MOVE SPACES TO HOUSER
MOVE SPACES TO HOPASS
*
CALL HTTPO$ USING HO
ON EXCEPTION
    * Error reported from HTTP open
  EXIT
END
MOVE HOHDL TO HTTPHDL                          * Store HTTP handle
```

5. Copy-Books

None.

6. See Also

HTTPC$	Close HTTP server session
HTTPH$	Set Request Header
HTTPS$	Send message and return status information and response text