

# The HTTP Server User Exit

The single HTTP server user exit is given control at strategic points in the processing of each HTTP request to enable an installation to control and manipulate the processing of HTTP requests.

This chapter covers the following topics:

- Installation
- General Interface
- Exit Points

## Installation

The HTTP server user exit must be a load module called HAANUXIT and must be available for loading by the HTTP server.

It is loaded when the HTTP server is initialized and becomes part of the HTTP server nucleus. It must be reentrant and may reside above the 16-megabyte line.

The member HAANUXIT on the HTPvrs.SRCE dataset is a sample user exit that implements each of the user exit points currently taken but simply returns normally to the user exit point. Thus, the user exit as delivered may be compiled and will not affect the normal operation of the HTTP server.

A message is issued during the server initialization process when the module is found and loaded. If this message is not issued, the exit is not operational.

## General Interface

The exit is called with the following registers:

Reg.	Contents
0	No relevant data
1	Pointer to a parameter list as described in this section
2-12	No relevant data
13	Pointer to an 18F savearea, which may be used by the exit
14	Address to which the exit returns. The exit must return using the following instruction to insure that the caller gets control back in the appropriate mode: BSM R0,R14
15	Pointer to the entry point of the module

## Exit Parameter List

On entry to the user exit, register 1 points to a parameter list that varies depending on the call.

The parameters passed to each exit point are documented in the appropriate section; however, the first parameter is always a pointer to the UXIT control block, which contains information about the HTTP server itself and the exit point for which the user exit has been called.

A DSECT to map this control block may be generated by the following statement in the Assembler program:

```
MYUXIT    HMANUXIT DSECT=YES
```

The parameter list may be addressed using the following statements assuming R1 contains what it contained when the module was entered:

```

L        Rx,0(,R1)           Rx -> UXIT control block
USING   UXIT,Rx             Address UXIT area

```

The following table describes the field names and their contents, which are relevant for all exit points. The UXIT control block may also contain data relevant only to specific exit calls. Any fields of additional significance are documented in the description of the exit point itself.

Field Name	Description
UXITFUNC	Contains a function code indicating the exit point for which the exit has been called.
UXITSSNM	Names the HTTP server subsystem for which the exit is being called. Each HTTP server has a subsystem name assigned when the server is started. This field allows the exit to uniquely identify an HTTP server when more than one is running in the same SMARTS address space or partition.
UXITUSER	A field available to the user. This will be NULL on entry to the exit the first time and will be passed intact to the user for each subsequent call. This field is intended for use as an anchor point for the exit where storage areas are maintained from call to call. In particular, the exit must insure that any storage areas acquired during the exit processing and returned to any given exit point are freed during the termination call.

The HAANUXIT source member on the HTPVrs.SRCE dataset shows how the UXIT parameter list may be addressed and processed. It should be used as the basis for any exit written at an installation.

## Entry/Exit Processing

The HTTP server has been written using a standard entry/exit mechanism to provide consistent standards within the server and to centralize much of the common coding used during entry/exit processing.

Two macros are used:

- HMANENT for entry processing; and
- HMANEXIT for exit processing.

## HMANENT Macro for Entry Processing

The HMANENT macro takes a number of parameters; however, the important ones from the exit's point of view are the WRKL and WRKD parameters.

- WRKL specifies the length of the savearea and work area that should be allocated for the exit.
- WRKD specifies the name of the working storage DSECT that will be used to address the area.

When control is passed to the code following the HMANENT macro expansion, the following registers are set:

Reg.	Contents
0	As passed by the calling program (for HAANUXIT, contains no relevant data)
1	As passed by the calling program (for HAANUXIT, contains a pointer to the HAANUXIT parameter list)
2-11	As passed by the calling program (for HAANUXIT, contains no relevant data)
12	Set up as the program base register
13	Pointer to a newly allocated savearea and work area allocated based on the length specified on the WRKL parameter. The DSECT name provided on the WRKD parameter must have the following format: WORK DSECT WORKSAVE DS 18F Exit work fields here WORKL EQU *-WORK Note that the field names are not important in this case once they are provided correctly on the HMANENT macro. The main point is that the area must start with an 18F savearea.
14	Address to which the exit returns. While the exit returns using the HMANEXIT macro, this may be useful for determining the mode of the calling program. In the case of the HAANUXIT, it is always called from programs running AMODE=31.
15	Pointer to the entry point of the module

## HMANEXIT Macro for Exit Processing

The HMANEXIT macro returns control to the calling program, optionally returning values in registers 15, 0, and 1. For HAANUXIT, the only relevant register in this case is 15, which is used as a return code.

Note that the HMANEXIT macro must get control with registers 12 and 13 containing the same values as were set up by the HMANENT macro.

## SMARTS API

The SMARTS Software Developer Kit (SDK) is required to make the full SMARTS API available to the user exit. These functions may be invoked as documented in the SMARTS SDK Programmer's Guide.

## Exit Points

For each exit point, the following subsections will be documented:

Subsection	Documents . . .
Purpose	the purpose for which the exit point is intended.
Parameters	the parameters, if any, supplied at a given exit point in addition to the UXIT control block.
Return Codes / Return Values	the return codes that may be returned in register 15 and the processing that any given return code causes. In all cases when register 15 contains an invalid return code on return from the exit, processing continues as described for return code 0.

### Initialization

#### Purpose

The initialization exit point is designed to provide an opportunity for the exit to initialize its own environment prior to any subsequent calls.

#### Parameters

Parm	Description
1	Pointer to the UXIT control block

#### Return Codes / Return Values

Return	Description
0	Continue processing normally

### Termination

#### Purpose

The termination entry point has been designed to provide the exit with a point at which it should clean up after any given request.

#### Parameters

Parm	Description
1	Pointer to the UXIT control block

## Return Codes / Return Values

Return	Description
0	Continue processing normally

## URL Processing

### Purpose

The URL processing exit point is taken immediately before the URL as provided by the user is processed. The installation can then

- modify or replace the URL all together and cause the HTTP server to use a different URL to service the request. This may be useful when URL names change and a cut-over period is required to handle old and new requests.
- reject access to the URL, which causes the HTTP server to return a 'permission denied' response to the request.

### Parameters

Parm	Description
1	Pointer to the UXIT control block
2	Pointer to the URL. This is a null1-terminated field. If this area is modified as it stands, the URL processed is the one returned in this area. If the pointer is replaced, this is not be used unless an appropriate return code is set on return from the exit.
3	Pointer to a fullword containing the length of the URL excluding the null-termination byte at the end. If the address in parameter 2 is replaced, this field must be updated to reflect the length of the URL in the area to which parameter 2 now points.

## Return Codes / Return Values

Return	Description
0	Continue processing normally. The URL is located and returned as provided to the exit or as modified in place by the exit.
4	The second parameter contains the address of a new storage location containing the URL to be used. The fullword pointed to by the third parameter has been updated to reflect the length of this new URL.
8	Access to the URL is denied. The request is rejected with a 'permission denied' response.

## Output Processing

## Purpose

The output processing exit point that is taken one or more times as output is built to be sent in response to the request. How often the exit is taken depends totally on what the URL represents and how large the output object is.

This exit point may be used to

- control translation of the output data if the basic facilities provided by the HTTP server are not adequate. This exit can effectively replace the HTTP server translation processing and do the translation itself.
- insert data into the output data stream. This is useful for inserting HTTP headers for any given request; however, users must be careful that the resulting output is still a valid HTTP request. The HTTP server does not control what is being sent in this case and therefore the user is responsible for ensuring that it is correct.
- prevent certain data from being sent.

## Parameters

Parm	Description
1	Pointer to the UXIT control block Two additional flags are used in this control block as follows: UXITMHHS => HTTP header sent. UXITMCHS => Content-type header sent. UXITMCON => Now sending HTTP content.
2	Pointer to output data. All data sent in response to a request is passed to this exit starting with the HTTP protocol headers and finishing with the content for the request, if any. If the data area pointed to by this pointer is modified in any way, the modified value is used for 'normal' processing. If this pointer is replaced, the data is only sent if an appropriate return code is returned from the exit.
3	Pointer to a fullword containing the length of the data pointed to by parameter 2. This must be modified if the length of data in the area pointed to by parameter 2 changes or the address is changed to point to a new area.
4	Pointer to the content type being sent. This reflects the content type as the HTTP server believes it to be. If the exit changes the content type in any way, it should be reflected here by the exit. When a CGI or Natural CGI request is being processing, this contains the value 'CGI' or 'NATCGI' until the content type being sent by the CGI program is established by the program itself sending a 'CONTENT-TYPE' header or a default header being sent by the HTTP server itself.

## Return Codes / Return Values

Return	Description
0	Continue processing normally. Normal processing means that the data in the area pointed to by parameter 2 for a length based on the fullword pointed to by parameter 3 is sent in response to the request. If headers are still being processed (that is, UXITMCNS is not on), these are always translated from EBCDIC to ASCII. If content is being processed (that is, UXITMCNS is on), the data is only translated from EBCDIC to ASCII if the CONTENT-TYPE header is of the form TEXT/* (where '*' is any value).
4	Do not translate the data. The data is sent exactly as is.
8	Do not send the data. The attempt to send the data is ignored.
12	Insert the data pointed to by a parameter 2 for a length specified by the fullword pointed to by parameter 3. The data is translated based on the rules for return code 0. The next call to the exit represents the same data as was presented on this call, thus enabling the exit to insert HTTP headers and/or content data if desired.
16	Same as return code 12; however, no HTTP server translation is performed on the data.

## Input Processing

### Purpose

Called when input is being processed for an incoming HTTP request, the input processing exit point may be used to

- translate incoming content in a way different from the basic translation services provided by the HTTP server.
- track or report identified HTTP headers or requests identified by certain requests.

### Parameters

Parm	Description
1	Pointer to the UXIT control block
2	Pointer to the data relating to the header identified by parameter 4
3	Length of the data pointed to by parameter 2
4	Pointer to the name of the HTTP header in EBCDIC for which the data pointed to by parameter 2 has been provided. Two 'logical headers' are defined for this interface ('logical' because they do not exist in the HTTP protocol specification and are used only to identify the data): QUERY_STRING the parameter string on the URL CONTENT the actual request content
5	Pointer to a fullword containing the length of the HTTP header pointed to by parameter 4.
6	Points to the CONTENT-TYPE for the request when it is identified.
7	Points to a fullword containing the length of the content type as pointed to by parameter 6.

**Return Codes / Return Values**

Return	Description
0	Continue processing normally. 'Normal processing' translates any HTTP header data from ASCII to EBCDIC unless it contains encoded data. Content is translated if the incoming content type is TEXT/* (where '*' represents any value).
4	Do not translate: use the data exactly as is in the buffer.

**Accept Processing****Purpose**

The accept processing exit point is taken immediately after a client has connected to the HTTP server. It may be used to control which IP addresses may connect with the HTTP server

**Parameters**

Parm	Description
1	Pointer to the UXIT control block
2	Pointer to a fullword containing the address of the peer application which has connected to the server. This is in the form of an Internet address as follows: H Address family: always '2' to indicate an Internet address H Port number of the peer program F IP address of the peer program This structure is set to nulls when the address of the peer application cannot be established. Note: Where a proxy is in use, this is the IP address of the proxy and not of the originating client.

**Return Codes / Return Values**

Return	Description
0	Continue processing normally; that is, accept and process the request.
4	No current meaning.
8	Reject the request. The HTTP server immediately terminates the connection with the client.