

Programming CGI Requests

This chapter covers the following topics:

- HAANUPR: The HTTP Server User Program Request Module
- HAANCGIG Interface Module
- HAANCGIL Interface Module
- HAANCGIP and HAANCGIT Interface Modules
- CGI Extension Interface Module Status

HAANUPR: The HTTP Server User Program Request Module

All current and future user program requests are serviced using the HAANUPR module. The basic call to the module is documented here.

Each call has the format

```
HAANUPR status function parm1 parm2 <..> parmn
```

-where

status	returns the status of the request in the form of a two-byte return code and two-byte reason code in contiguous storage.
function	the name of the function to be invoked. The name must be in character format as described in the following subsections, left aligned in the 16-byte field and padded to the right with blanks.
parm1, parm2, ...	a set of parameters specific to the function named and described in the subsection devoted to that function.

Each available function is described in the following section along with its parameters and a list of return and reason codes and their meanings.

Standard Return and Reason Codes

The following reason and associated return codes may be returned on any request to HAANUPR:

Reason Code	Return Code	Meaning
4	8	Request failed due to insufficient storage. HAANUPR attempts to acquire a save and work area of about 90 bytes from local storage. If it fails, this status is returned.
40	12	Not an HTTP request. The HAANUPR request was issued from a program that was not running as an HTTP server request.
52	12	Unrecognized request. The 16-character function area provided as the second parameter to HAANUPR contained a function request that HAANUPR did not recognize. This may occur for the following reasons: <ul style="list-style-type: none"> • the character string in the field is either misspelled or not a valid function name. • the string contains lowercase characters. All function identifiers must be in uppercase. • the field length was not 16 characters. The function name must be left-aligned and padded to the right with blanks (not nulls).

The CONVERSE Function

The CONVERSE function is used by conversational CGI programs that wish to maintain a connection with the client browser. Refer to the section in this manual that discusses conversational CGI programs for more information. This function may only be issued after an ENABLE-CONVERSE has been successfully issued and some data has been written to stdout. If the CONVERSE is successful, when the program is next dispatched, the user response to the data sent by the CONVERSE will be available to the application program.

This function is invoked as follows:

```
HAANUPR status 'CONVERSE'
```

CONVERSE Parameters

The CONVERSE function has no additional parameters.

CONVERSE Return and Reason Codes

Reason Code	Return Code	Meaning
56	8	Conversational sequence error. The CONVERSE request may only be issued after some data has been written in response to an HTTP request. It is not possible to converse if the user has not received a mechanism with which to respond; i.e., an HTML form.
60	8	Conversation error. A request was received to converse, however, no ENABLE-CONVERSE was previously issued. The CGI program must first indicate that it wishes to establish a conversation by issuing an ENABLE-CONVERSE function call.

The DISABLE-CONVERSE Function

The DISABLE-CONVERSE function indicates that the user program no longer wishes to converse with the client browser. This function may only be issued after an ENABLE- CONVERSE has been successfully issued at some time previously. It must also be issued prior to any output data being sent to the client browser for a given conversation. In other words, after a program has been redispached after a CONVERSE call, if the conversation is to be terminated, the DISABLE-CONVERSE must be issued before any final output is written to the client browser.

This function is invoked as follows:

```
HAANUPR status 'DISABLE-CONVERSE'
```

DISABLE-CONVERSE Parameters

The DISABLE-CONVERSE function has no additional parameters.

DISABLE-CONVERSE Return and Reason Codes

Reason Code	Return Code	Meaning
56	8	Conversational sequence error. The DISABLE-CONVERSE request may only be issued before any data has been written in response to the current HTTP request. The HTTP server must know before the response is written that this is the last output for the conversation.
64	8	Conversations not supported. This indicates that the HTTP server CONV configuration parameter for the server is set to NO indicating that the server is not supporting conversations.

The ENABLE-CONVERSE Function

The ENABLE-CONVERSE function indicates that a CGI program wishes to start a conversation with the client browser. It must be issued before any output whatsoever is issued in response to a request otherwise, the request will fail. Once this request has been issued, the session will remain in conversation with the client browser until a DISABLE-CONVERSE is issued or the program terminates.

This function is invoked as follows:

```
HAANUPR status 'ENABLE-CONVERSE'
```

ENABLE-CONVERSE Parameters

The ENABLE-CONVERSE function has no additional parameters.

ENABLE-CONVERSE Return and Reason Codes

Reason Code	Return Code	Meaning
56	8	Conversational sequence error. The ENABLE-CONVERSE request may only be issued before any data has been written in response to the current HTTP request. The HTTP server must know before the response is written that the CGI program wishes to converse.
64	8	Conversations not supported. This indicates that the HTTP server CONV configuration parameter for the server is set to NO indicating that the server is not supporting conversations.

The GET-DATA Function

The GET-DATA function may be used to get the value of a field name submitted as part of a CGI request. It may also be used to test for the existence of certain fields on the screen which may be used when lists are presented in HTML format to a user. These lists result in a field name with no value being submitted as part of a CGI request when they are selected.

The interface module will search either the input parameter area as provided when using the GET HTTP method, or the content area as provided when using the POST HTTP method. If the variable requested is not found in the input from the HTML page, or if it has not been specified that only the HTML page must be searched, the interface module will check for a server defined variable (i.e. a defined environment variable) of this name. The caller of this interface module does not have to worry about the type of HTTP method that generated the CGI request as this is handled by the interface module.

This function is invoked as follows:

```
HAANUPR status 'GET-DATA' field value length type start
```

GET-DATA Parameters

field	is the name of the field from the HTML page for which this request is being issued. This field name must be terminated with a blank in order for the interface routine to correctly determine the length of the field name for which it is searching. Note: The longest variable name that can currently be handled by this interface is 255 bytes excluding the blank.
value	is a field with a minimum length of the binary value specified in the length field. If this area is smaller than the length specified in the 'length' parameter, overwrites will occur and the results will be unpredictable. When the field name specified in the 'field' parameter is found in the CGI input and has a value associated with it, the value submitted for the field is copied to this area for a maximum length of the length specified in the length parameter. The value is truncated if longer than this and a return and reason code set to indicate this event. If the value is shorter than the length set in the 'length' parameter, the actual length of the value will be set in the 'length' parameter.
length	is a 2-byte binary value containing the length of the area provided for the value to be returned in the 'value' parameter. This is set to the length of the returned value if the field name is found and has a value associated with it.
type	is a one-byte alpha indicating the type of variable which is to be returned and may be used to restrict the search as follows: 'S' Request server defined variable. 'P' Request variable defined on the HTML page. ' ' First found will satisfy request. When the request is completed and the variable requested is found, this field is modified to contain an 'S' or a 'P' depending on where the variable was found.
start	is a two-byte binary field that may be specified to indicate the offset from which the requested variable is to be returned. Its purpose is to enable a program to obtain the contents of a long text field in chunks. When this field is not specified, the default is to start at position 0 of the input field which is the start of the field.

Note:

Where a field name appears twice on a HTML page, only the first is accessible using this mechanism. For this reason, HTML pages designed to work with this mechanism should use unique names although it is perfectly legal in HTML terms to have the same field names specified many times. To process multiple fields with the same name, the LIST-DATA function must be used.

GET-DATA Return and Reason Codes

Reason Code	Return Code	Meaning
8	12	Invalid length supplied in a length field to the interface function.
20	4	Variable length error. It was not possible to return the full length of a variable due to the fact that insufficient space was provided in the users' parameters to hold the value to be returned.
24	12	The format of the parameters provided was invalid.
28	8	Variable name requested was not found in the CGI data.
32	16	A logic error occurred due to the format of the content data provided with the CGI request.
36	8	Insufficient space to return data.
40	12	Returned when these modules are called from a program which is not running as a result of an HTTP request and therefore does not have the data available to satisfy the request.
48	12	Invalid parameter list. This indicates that one or more parameters for a given request have not been provided or contain invalid data.

The LIST-DATA Function

The LIST-DATA function may be used to get a list of both the variable names from the HTML page and their values, and the server defined or environment variables defined at the server for the request. The interface allows that one or more of these variables may be returned at the same time and multiple requests may be made to return all defined variables to the application program.

The server defined or environment variables are returned first, while the variables found in the HTML page are returned once all environment variables have been returned. When issuing multiple requests, the same TOKEN parameter must be provided to the interface each and every time until the list is exhausted.

This function is invoked as follows:

```
HAANUPR status 'LIST' token entries name-length
  name1 value-length1 value1 type1
  name2 value-length2 value2 type2
  <..>
  namen value-lengthn valuen typen
```

LIST-DATA Parameters

token	is a 4-byte binary token used by the interface for multiple requests. When this is null, the listing of variables starts from the first one found. When all variables available cannot be found, this is set to an internal token value to allow the interface to continue the list at the next variable to be returned. The token is reset to null when all available values have been returned to the caller.
entries	is a 4-byte binary field containing the number of variable name and value entries which the application program may accept from the interface in one call. For each entry, a set of return variables (name, value-length, value and type) must be provided, otherwise the results will be unpredictable. When the call completes, this field contains the number of variable sets returned. This must be used in association with the return and reason codes to determine if all data has been returned or if any data has been returned (the last call may have exhausted the list but may not be obvious from the return, reason codes and entries value returned).
name-length	is a 4-byte binary field containing the maximum length of variable name that can be returned. This must be set based on the length of the name fields passed to the interface.

The following constitute a set which is required to return the information about a given variable to the application program. For each entry specified, a set of fields must be provided to contain the data to be returned. Failure to do this will result in unpredictable results.

name	is an alpha field in which the name of a server or HTML page variable is returned. Its maximum length is determined by the name-length variable. If any name to be returned to the application exceeds this length, the value is truncated.
value-length	is a 4-byte binary field containing the maximum length of the value for the variable that can be accepted for this variable instance. The following value field must have at least this amount of space allocated for it, otherwise, overwrites will occur. When a variable value is returned, this field is changed to reflect the true length of the value as determined from the data. If the value is longer than the value specified here, the value is truncated and this field remains unchanged.
value	is an alpha field in which the value for the variable name associated with this value field is returned. It must be at least as long as the value specified in its associated value-length field, otherwise storage overwrites may occur. When a variable is found, its name is returned in the associated 'name' parameter and the value is returned in this field. The actual length of the value is set in the associated value-length field when the variable name is found. If the value is longer than the value-length specification, the value is truncated.
type	is a 1-byte alpha field that indicates where the variable with which it is associated was found. When the variable is an environment variable set in the server environment, this field contains a 'S'. When the variable was found on the HTML page returned from the client, this field contains 'P'.

LIST-DATA Return and Reason Codes

Reason Code	Return Code	Meaning
8	12	Invalid length supplied in a length field to the interface function.
20	4	Variable length error. It was not possible to return the full length of a variable due to the fact that insufficient space was provided in the users' parameters to hold the value to be returned.
24	12	The format of the parameters provided was invalid.
32	16	A logic error occurred due to the format of the content data provided with the CGI request.
36	8	Insufficient space to return data.
40	12	Returned when these modules are called from a program which is not running as a result of an HTTP request and therefore does not have the data available to satisfy the request.
44	4	End of data reached. This will be set for the LIST-DATA function when all data has been returned. The application program should check the 'entries' field as the number of entries returned may be '0' depending on the sequence of LIST-DATA function requests.
48	12	Invalid parameter list. One or more parameters for a given request have not been provided or contain invalid data.

The PUT-BINARY Function

The PUT-BINARY function enables a CGI application program to send output in response to the CGI request. This output is provided to the HTTP server in the same way as 'standard' CGI output is processed.

The PUT-BINARY function differs from the PUT-TEXT function only in terms of the way it processes the parameters passed to it. PUT-BINARY simply takes the data and length provided at face value and passes them directly to the HTTP output processing module. Refer to the section on PUT-TEXT for information about how it processes output.

This function is invoked as follows:

```
HAANUPR status 'PUT-BINARY' data length
```

PUT-BINARY Parameters

data	is the data or the field containing the data to be output in response to the CGI request.
length	is a 2-byte binary value containing the length of the data area provided. The contents of the data area are output for exactly the length specified in this field.

PUT-BINARY Return and Reason Codes

Reason Code	Return Code	Meaning
8	12	Invalid length supplied in a length field to the interface function.
12	4	Warning returned from the HTTP server output processing module.
16	8	Error returned from the HTTP server output processing module.
24	12	The format of the parameters provided was invalid.
32	16	A logic error occurred due to the format of the content data provided with the CGI request.
40	12	Returned when these modules are called from a program that is not running as a result of an HTTP request and therefore does not have the data available to satisfy the request.
48	12	Invalid parameter list. One or more parameters for a given request have not been provided or contain invalid data.

The PUT-TEXT Function

The PUT-TEXT function also enables a CGI application program to send output in response to the CGI request. This output is provided to the HTTP server in the same way as 'standard' CGI output is processed.

The PUT-TEXT function differs from the PUT-BINARY function only in terms of the way it processes the parameters passed to it. PUT-TEXT assumes text output and strips all trailing non-printable characters from the end of the provided data (as determined from the provided data and length) up to the first character that has a value greater than blank. The only exception to this is where a carriage return (X'0D') or a line feed (X'0A') is encountered. In either case, this will also be treated as valid data and treated as the last character in the output data. This is useful where a standard area and length are to be used as output text data as the program generating the output must include the CR or LF characters to format text correctly. Using the PUT-BINARY request, anything following the CR or LF is also treated as data and may cause output to 'skew'.

This function is invoked as follows:

```
HAANUPR status 'PUT-TEXT' data length
```

Note:

Use of this function has no bearing on the way data is translated; only on the way the actual length of the data is calculated prior to output. After it has been output, translation occurs based on the criteria described earlier.

PUT-TEXT Parameters

data	is the data or the field containing the data to be output in response to the CGI request.
length	is a 2-byte binary value containing the length of the data area provided. This must contain the maximum length of the data area. As stated previously, PUT-TEXT strips off all trailing non-printable characters in the data area.

PUT-TEXT Return and Reason Codes

Reason Code	Return Code	Meaning
8	12	Invalid length supplied in a length field to the interface function.
12	4	Warning returned from the HTTP server output processing module.
16	8	Error returned from the HTTP server output processing module.
24	12	The format of the parameters provided was invalid.
32	16	A logic error occurred due to the format of the content data provided with the CGI request.
40	12	Returned when these modules are called from a program which is not running as a result of an HTTP request and therefore does not have the data available to satisfy the request.
48	12	Invalid parameter list. This indicates that one or more parameters for a given request have not been provided or contain invalid data.

HAANCGIG Interface Module**Note:**

This documentation is only provided for compatibility. All applications should use the HAANUPR GET-DATA function to achieve this functionality.

The HAANCGIG module obtains the value of a field name submitted as part of a CGI request.

It also determines the existence of fields selected from lists presented to the user in HTML format which result in a field name with no value being submitted as part of a CGI request.

The interface module ascertains the type of HTTP method used to generate the CGI request and searches

- the input parameter area as provided when using the GET HTTP method; or
- the content area as provided when using the POST HTTP method.

If the variable requested is not found in the input from the HTML page and if the search has not been restricted to the HTML page only, the interface module checks for a server-defined variable (that is, a defined environment variable) of this name.

It is not necessary to be concerned about the HTTP method that generated the CGI request when calling the interface module as this is handled by the interface module itself.

The HAANCGIG interface module must be invoked with the following parameter list:

```
HAANCGIG status field value length type start
```

-where

status	returns the status of the request
field	names the field from the HTML page for which the request is being issued. The field name must be terminated with a blank in order for the interface routine to correctly determine the length of the field name for which it is searching. The longest variable name that can currently be handled by this interface is 255 bytes excluding the blank.
value	is a field with a minimum length of the binary value specified in the length field. If this area is smaller than the length specified in the 'length' parameter, overwrites occur and the results are unpredictable. When the field name specified in the 'field' parameter is found in the CGI input and has a value associated with it, the value submitted for the field is copied to this area for a maximum length specified in the length parameter. If the value is longer, it is truncated and a return and reason code are set. If the value is shorter, the actual length of the value is set in the 'length' parameter.
length	is a 2-byte binary value containing the length of the area provided for the value to be returned in the 'value' parameter. This is set to the length of the returned value if the field name is found and has a value associated with it.
type	is a one-byte alpha field indicating the type of variable to be returned. It may be used to restrict the search as follows: 'S' request server-defined variable. 'P' request variable defined on the HTML page ' ' first found satisfies request. When the request is completed and the requested variable is found, this field is modified to contain an 'S' or 'P' depending on where the variable was found.
start	is a two-byte binary field used to indicate the offset from which the requested variable is to be returned. This information makes it possible for a program to obtain the contents of a long text field in chunks. By default, the program starts at the beginning of the field, which is position 0 of the input field.

Note:

If a field name appears twice on an HTML page, only the first occurrence is accessible using this mechanism. Thus HTML pages designed to work with this mechanism should use unique field names, although HTML itself allows the same field names to be specified multiple times.

HAANCGIL Interface Module

Note:

This documentation is provided only for compatibility. All applications should use the HAANUPR LIST-DATA function to achieve this functionality.

The HAANCGIL module is used to obtain a list of

- the variable names from the HTML page and their values; and
- the server-defined or environment variables defined at the server for the request.

The interface allows one or more of these variables to be returned at the same time and multiple requests may be made to return all defined variables to the application program.

The server-defined variables or environment variables are returned first, while the variables found in the HTML page are returned after all environment variables have been returned.

The HAANCGIL interface module must be invoked with the following parameter list:

```
HAANCGIL status token entries name-length
          name1 value-length1 value1 type1
          name2 value-length2 value2 type2
          ...
          namen value-lengthn valuen typen
```

-where

status	is used to return the status of the request as documented in the section Interface Module Status.
token	is a 4-byte binary value used by the interface for multiple requests. When the token value is null, the listing of variables starts from the first one found. When all variables available cannot be found, "token" is set to an internal value to allow the interface to continue the list at the next variable to be returned. The token value is reset to null when all available values have been returned to the caller.
entries	is a 4-byte binary field containing the number of variable name and value entries which the application program may accept from the interface. For each entry, a set of return variables (name, value-length, value, and type) must be provided; otherwise, the results are unpredictable. When the call has completed, this field contains the number of variable sets returned. This number must be used in association with the return and reason codes to determine if all data has been returned or if any data has been returned (the last call may have exhausted the list but may not be obvious from the return/reason codes and entries value returned).
name-length	is a 4-byte binary field that specifies the maximum length of the variable 'name' that can be returned. This length must be set based on the length of the 'namen' fields passed to the interface.
<p>For each entry specified, a set of fields must be provided to contain the data to be returned. Failure to provide these fields produces unpredictable results. The following fields comprise the set required to return the information about a given variable to the application program:</p>	
name	is an alpha field in which the name of a server or HTML page variable is returned. The maximum length of the name is determined by the variable 'name-length'. Any name to be returned to the application that exceeds this length is truncated.

value-length	is a 4-byte binary field containing the maximum length of the variable 'value' that can be accepted for this variable instance. The following 'value' field must have at least this amount of space allocated for it; otherwise, storage overwrites occur. When a variable 'value' is returned, this field is changed to reflect the true length of the 'value' as determined from the data. If the 'value' is longer than the 'value-length' specified here, the 'value' is truncated and the 'value-length' field remains unchanged.
value	is an alpha field in which the value for the variable 'name' associated with this value field is returned. This field must be at least as long as the value specified in the associated 'value-length' field; otherwise, storage overwrites occur. When a variable is found, its name is returned in the associated 'name' parameter and the value is returned in this field. The actual length of the value is set in the associated 'value-length' field when the variable name is found. If the value is longer than the value-length specification, the value is truncated.
type	is a 1-byte alpha field that indicates where the associated variable was found. When the variable is an environment variable set in the server environment, this field contains 'S'. When the variable was found on the HTML page returned from the client, this field contains 'P'.

HAANCGIP and HAANCGIT Interface Modules

Note:

This documentation is provided only for compatibility. All applications should use the HAANUPR PUT-BINARY and PUT-TEXT functions to achieve this functionality.

The HAANCGIP and HAANCGIT modules enable a CGI application program to send output in response to the CGI request. This output is provided to the HTTP server in the same way as 'standard' CGI output is processed. See the section Standard CGI Operation for more information about the processing of this output.

The HAANCGIT interface module differs from the HAANCGIP interface module only in the way it processes the parameters passed to it:

- HAANCGIP accepts the data and length provided at face value and passes them directly to the HTTP output processing module.
- HAANCGIT assumes text output and strips all trailing, nonprintable characters from the end of the provided data (as determined from the provided data and length) up to the first character which has a value greater than blank.

HAANCGIT treats a carriage return (X'0D') or a line feed (X'0A') as valid data and as the last character in the output data. This is useful where a standard area and length are to be used when outputting text data as the program generating the output must include the CR or LF characters to format text correctly. Using the HAANCGIP interface, anything following the CR or LF is also treated as data and may cause output to 'skew'.

The HAANCGIP/T interface modules must be invoked with the following parameter list:

HAANCGIP/T status data length

-where

status	is used to return the status of the request as documented in the section Interface Module Status.
data	is the data or the field containing the data to be output in response to the CGI request.
length	is a 2-byte binary value containing the length of the data area provided. For the HAANCGIP interface, this is the exact length of the data to be sent. For the HAANCGIT interface, this is the maximum length of the data area. As stated previously, HAANCGIT strips off all trailing nonprintable characters in the data area.

CGI Extension Interface Module Status

The status parameter passed to the CGI extension interface modules is a four-byte contiguous area comprising a two-byte return code followed by a two-byte reason code. Appropriate definitions are provided in the language-specific sections; however, the return and reason codes are documented in this section.

The language-related call to the interface modules must be successful before any return or reason codes are entered. The call fails if the interface module

- is not linked with the module produced by the language compiler; or
- could not be loaded at run time if this facility is available to the language.

Interface Module Return Codes

The following codes may be returned by the CGI extension interface modules.

Return Code	The CGI extension interface request . . .
0	was processed successfully
4	was processed successfully but there is additional information related to the calling of the function in the reason code field
8	failed due to some environmental error. The reason code indicates what happened
12	was invalidated based on information the user provided or failed to provide. The reason code indicates what happened.
16	failed due to an internal processing error. The reason code indicates what happened.

Interface Module Reason Codes

Note:

This documentation is provided only for compatibility. All applications should use the HAANUPR interface functions to achieve this functionality.

The following reason codes may be returned by the CGI extension interface modules. Each reason code has an associated return code as documented in this table.

Reason Code	Associated Return Code	Meaning
4	8	Request failed due to insufficient storage
8	12	Invalid length supplied to the interface function in a length field
12	4	Warning returned from the HTTP server output processing module
16	8	Error returned from the HTTP server output processing module
20	4	Variable length error The full length of a variable could not be returned due because insufficient space was provided in parameters supplied by the user to hold the value to be returned.
24	12	The format of the parameters provided was invalid
28	8	The variable name requested was not found in the CGI data
32	16	A logic error occurred due to the format of the content data provided with the CGI request
36	8	Insufficient space to return data
40	12	The module is called from a program that is not running as a result of a HTTP request and therefore does not have the data available to satisfy the request
44	4	End of data reached. This will be set for the HAANCGIL function when all data has been returned. The application program should check the 'entries' field as the number of entries returned may be '0' depending on the sequence of HAANCGIL interface requests.
48	12	Invalid parameter list. One or more parameters for a given request have not been provided or contain invalid data.