

REQUEST DOCUMENT

Note:

This statement is only available under Windows.

```

GET DOCUMENT FROM operand1

{
  WITH
  {
    [USER operand2]
    [PASSWORD operand2]
    [HEADER [ [NAME] operand4 [VALUE] operand5 ]...]
    [DATA { ALL operand6
            [operand7 operand8]...}]
  }
  [RETURN
  {
    [HEADER { [ [NAME] operand10 [VALUE] operand11 ]...}]
    [PAGE operand12]
  }
  RESPONSE operand13
  [GIVING operand14]

```

Operand	Possible Structure		Possible Formats												Referencing Permitted	Dynamic Definition	
Operand1	C	S	A													no	yes
Operand2	C	S	A													no	yes
Operand3	C	S	A													no	yes
Operand4	C	S	A													no	yes
Operand5	C	S	A	N	P	I	F		D	T	L					no	yes
Operand6	C	S	A	N	P	I	F	B	D	T	L					no	yes
Operand7	C	S	A													no	yes
Operand8	C	S	A	N	P	I	F		D	T	L					no	yes
Operand9		S	A	N	P	I	F	B	D	T	L					no	yes
Operand10	C	S	A													no	yes
Operand11		S	A	N	P	I	F	B	D	T	L					no	yes
Operand12		S	A					B								no	yes
Operand13		S	A													no	yes
Operand14		S						I								no	no

Function

The GET DOCUMENT statement gives you the possibility to access an external system. To be more precise, it allows you to use Natural's HTTP functionality directly.

operand1

Operand1 is the URL to access a document. The Address of the document in the first step is always "http://..."

The tables below are only valid if operand1 begins with "http://" or "https://".

The following are the values for HTTP requests:

HTTP Action	HEAD	POST	GET	PUT
WITH HEADER	?	?	?	?
WITH DATA	-	X	-	ALL
RETURN HEADER	X	?	?	?
RETURN PAGE	-	X	X	?

Key:

?	optional
-	not specified
X	has to be specified
ALL	only with option ALL

Restrictions

The tables below are only valid if operand1 begins with "ftp://"

operand2

Note:

Operand2 can only be used in conjunction with operand3.

Operand2 is the name of the user that will be used for the request.

operand3

Note:

Operand3 can only be used in conjunction with operand2.

Operand3 is the password of the user that will be used for the request.

operand4/5

Operand4 is the name of a HEADER (needed for http) sent with this request. (URL decoding necessary especially "&", "=", "%")

Operand5 is the value of a HEADER (needed for http) sent with this request (URL decoding necessary especially "&", "=", "%") structure. Additional expressions using other fields can be added.

Note:

Operand4 and operand5 can only be used as a couple.

operand6

Operand6 is a complete document that will be sent. This value is needed for http PUT.

operand7/8

Operand7 is the name of a DATA-Variable sent with this request. This value is needed for http POST. (URL decoding necessary especially "&", "=", "%")

Operand8 is the value of a DATA-Variable sent with this request. This value is needed for http POST. (URL decoding necessary especially "&", "=", "%")

Note:

Operand7 and operand8 can only be used as a couple.

operand9

Operand9 represents ALL received Header values as name value pair. The name and values are separated with ":" and "CR". (Internally all "CR-NL" will be transformed to "CR").

operand10/11

Operand10 is the name of a HEADER received with this request. The HEADER is needed for http.

Operand11 is the value of a HEADER received with this request. The HEADER is needed for http.

Note:

Operand10 and operand11 can only be used as a couple.

operand12

Operand12 is the returned document for this request.

operand13

Operand13 is the response (e.g. "HTTP 200 OK" or only "200")

operand14

Operand14 is the runtime error if the request could not be performed.



Steps in a Typical HTTP Client Application with WinInet

The following table shows the steps you might perform in a typical HTTP client application:

Goal	Action	Effects
Begin a HTTP session.	Create a CInternetSession object.	Initializes WinInet and connects to the server.
Connect to a HTTP server.	Use CInternetSession::GetHttpConnection.	Returns a CHttpConnection object.
Open a HTTP request.	Use CHttpConnection::OpenRequest.	Returns a CHttpFile object.
Send a HTTP request.	Use CHttpFile::AddRequestHeaders and CHttpFile::SendRequest.	Finds the file. Returns FALSE if the file is not found.
Read from the file.	Use CHttpFile.	Reads the specified number of bytes using a buffer you supply.
Handle exceptions.	Use the CInternetException class.	Handles all common Internet exception types.
End the HTTP session.	Dispose of the CInternetSession object.	Automatically cleans up open file handles and connections.

Example:

```
REQUEST DOCUMENT FROM "http://bolsap1:5555/invoke/sap.demo/handle_RFC_XML_POST" WITH
    USER
    PASSWORD
    HEADER
    "User-Agent"      #USER_AGENT
    "Accept-Language" #ACCEPT_LANGUAGE
    "Content-Type"    #CONTENT_TYPE
    DATA
    'XMLData'        QUERYXML
    'repServerName'  ' NT2 '
    RETURN PAGE RESULTXML
```