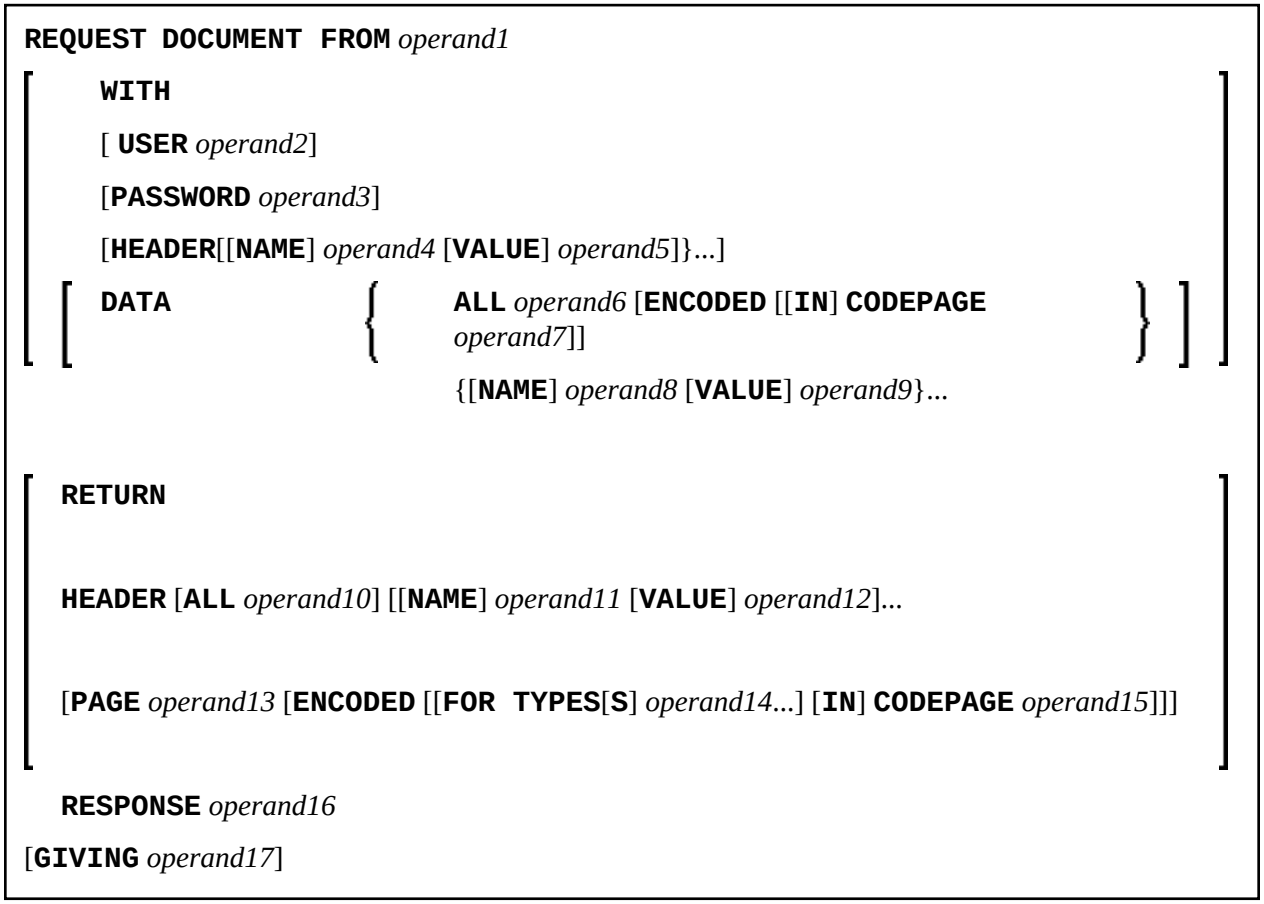
REQUEST DOCUMENT



This chapter covers the following topics:

- Function
- Syntax Description
- Encoding of Incoming/Outgoing Data
- Examples

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Belongs to Function Group: *Internet and XML*

Function

The `REQUEST DOCUMENT` statement gives you the means to access an external system, see *Statements for Internet and XML Access* in the *Programming Guide*.

For information on Unicode support, see *Statements* in the *Unicode and Code Page Support* documentation.

Restrictions for Cookies

Under the HTTP Protocol, a server uses cookies to maintain state information on the client workstation.

`REQUEST DOCUMENT` is implemented using internet option settings. This means that, depending on the security settings, cookies will be used.

If the internet option setting `Disabled` is set, no cookies will be sent, even if a cookie header (*`operand4/operand5`*) is sent.

For server environments, do not use the internet option setting `Prompt`. This setting leads to a "hanging" server, because no client will be able to answer the prompt.

In mainframe environments, cookies are not supported and are ignored.

Cookies are handled automatically by the Windows API. This means that, if cookies are enabled in the browser, all incoming cookies will be saved and sent automatically with the next request.

Syntax Description

Operand Definition Table:

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	U	N	P	I	F	B	D	T	L			no	yes
operand7	C	S				A												no	yes
operand8	C	S				A												no	yes
operand9	C	S				A		N	P	I	F		D	T	L			no	yes
operand10		S				A												no	yes
operand11	C	S				A												no	yes
operand12		S				A		N	P	I	F	B	D	T	L			no	yes
operand13		S				A	U					B						no	yes
operand14	C	S				A												no	yes
operand15	C	S				A												no	yes
operand16		S								I4								no	yes
operand17		S								I4								no	no

Syntax Element Description:

Syntax Element	Description
`DOCUMENT FROM` *operand1*	**Location of Document:** *operand1* is the URL to access a document. **Note:** The information below is only valid if *operand1* begins with `http://` or `https://`.
`WITH`	**WITH Clause:** This clause may be used to specify optional user/password, header and data details for the request.
`USER` *operand2*	**User Name:** *operand2* is the name of the user that will be used for the request.

Syntax Element	Description
PASSWORD *operand3*	**User Password:** *operand3* is the password of the user that will be used for the request.
HEADER {[[NAME] *operand4* [VALUE] *operand5*]}...	**Header Clause:** *operand4* and *operand5* can only be used in conjunction with each other: • *operand4* is the name of a HEADER variable sent with this request. • *operand5* is the value of a HEADER variable sent with this request. **Note:** **Header Name for *operand4*:** Header names must not contain a carriage return (CR), a line feed (LF) or a colon (:). This will not be checked by the REQUEST DOCUMENT statement. For valid header names, please see the HTTP specifications. For compatibility with the web interface, header names can be written with underscore (_) instead of a dash (-). (Internally, the underscore is replaced by a dash). **Header Value for *operand5*:** Header values are not allowed to contain CR/LF. This will not be checked by the REQUEST DOCUMENT statement. For valid header values and formats, please see the HTTP specifications. **General Information on Headers:** For a HTTP request, some headers are required, for example: Request-Method or Content-Type. These headers will be automatically generated depending on the parameters given with the REQUEST DOCUMENT statement. See also *Automatically Generated Headers*.
DATA	**DATA Clause:** You may specify either a specific DATA variable name and value (see *operand8* and *operand9* below) or the complete document (see *DATA ALL Clause* below).

Syntax Element	Description
ALL *operand6*	*operand6* is a complete document that is to be sent. This value is needed for the HTTP request method PUT (see *Automatically Generated Headers*). See *Encoding of Incoming/Outgoing Data*, *DATA ALL Clause*.
[ENCODED [[IN] CODEPAGE *operand7*]	*operand6* will be encoded from the default code page (value of the system variable *CODEPAGE) to the code page given in *operand7*. See *Encoding of Incoming/Outgoing Data*, *DATA ALL Clause*.

<table>
<tr><th>Syntax Element</th><th>Description</th></tr>
<tr><td>{[NAME] operand8 [VALUE] operand9}...</td><td>
DATA Variable Name and Value:

operand8 and operand9 can only be used in conjunction with each other:

• operand8 is the name of a DATA variable to be sent with this request. This value is needed for the HTTP request method POST (URL-encoding necessary, especially ampersand (&), equal sign (=), percent sign (%) characters.

• operand8 is the name of a DATA variable to be sent with this request. This value is needed for the HTTP request method POST (URL-encoding necessary, especially ampersand (&), equal sign (=), percent sign (%) characters.

Restriction:

If operand8/operand9 is given, and the communication is http:// or https:// by default, the request method POST (see Automatically Generated Headers) with content type application/x-www-form-urlencoded is used. During the request, operand8/operand9 will be separated by equal sign (=) and ampersand (&) characters. Therefore the operands are not allowed to contain equal sign (=), ampersand (&) and, because of URL-encoding, percent sign (%) characters. These characters are considered "unsafe" and need to be encoded as:
<table>
<tr><th>Character</th><th>URL-Encoding Syntax</th></tr>
<tr><td>%</td><td>%25</td></tr>
<tr><td>&</td><td>%26</td></tr>
<tr><td>=</td><td>%3D</td></tr>
</table>
See also General Note for URL-Encoding.
</td></tr>
<tr><td>RETURN</td><td>
RETURN Clause:

This clause can be used to specify the HEADER and/or PAGE return information.
</td></tr>
</table>

Syntax Element	Description
HEADER [ALL *operand10*]	**RETURN HEADER ALL Clause:** When this clause is specified, *operand10* contains all header values delivered with the HTTP response. The first line contains the status information and all following lines contain the headers as pairs of name and value. The names always end in a colon (:) and the values end in a linefeed (LF). Internally, all carriage returns/line feeds (CR/LF) are transformed into line feeds (LF).
HEADER [[NAME] *operand11*] [VALUE] *operand12*]...	**RETURN HEADER NAME/VALUE Clause:** When this clause is specified, only specific header information is returned. *operand11* and *operand12* can only be used in conjunction with each other: ● *operand11* is the name of a HEADER received with this request. The HEADER is needed for HTTP. ● *operand12* is the value of a HEADER received with this request. The HEADER is needed for HTTP. Return Header Name for *operand11*: For compatibility with the web interface, header names can be written with underscore (_) instead of dash (-) characters. Internally, the underscore is replaced by a dash. If *operand11* is a blank string, the status information is returned. HTTP/1.0 200 OK
RETURN PAGE	**RETURN PAGE Clause:** You can use the PAGE clause if you want to have the incoming data encoded in a specific code page. See *Encoding of Incoming/Outgoing Data, RETURN PAGE Clause* below.

Syntax Element	Description
PAGE *operand13*	*operand13* is the document returned for this request. See *Encoding of Incoming/Outgoing Data*, *RETURN PAGE Clause* below.
[ENCODED [[FOR TYPE[S] *operand14...*] [IN] CODEPAGE *operand15*]]	*operand14* is the list of mime-types for which an encoding of the returned document in *operand13* will be performed. See *Encoding of Incoming/Outgoing Data*, *RETURN PAGE Clause* below.
	operand15 is the code page which, if necessary, will be used for the encoding of *operand13*. If the value of *operator15* is blank, no conversion occurs. *operand13* is then encoded in the default code page (profile parameter CP in the Configuration Utility). See *Encoding of Incoming/Outgoing Data*, *RETURN PAGE Clause* below.
RESPONSE *operand16*	**RESPONSE Clause:** The RESPONSE clause is used to display the response code number of the request. *operand16* is the response code number of the request, for example: 200 (Request Completed). See also *Overview of Response Numbers for HTTP Requests*.
GIVING *operand17*	**GIVING Clause:** *operand17* contains the Natural error if the request could not be performed.

Automatically Generated Headers (*operand4/5*)

Request-Method

The following values are supported for *operand5*: HEAD, POST, GET, and PUT.

The following table shows the automatic calculation of Request-Method depending on the given operands:

Operand	Request Method			
	HEAD	**POST**	**GET**	**PUT**
WITH HEADER (***operand4/operand5***)	optional	optional	optional	optional
WITH DATA (***operand7/operand8***)	not specified	specified	not specified	only with option `ALL` (*`operand6`*)
RETURN HEADER (***operand10*** **to** ***operand12***)	specified	optional	optional	optional
RETURN PAGE (***operand13***)	not specified	specified	specified	optional

Content-Type

If the request method is `POST`, a content-type header has to be delivered with the HTTP request. If no content-type is set explicitly, the automatically generated value of *`operand5`* is:

```
application/x-www-form-urlencoded
```

Note:
It is possible to overwrite the automatically generated headers. Natural will not check them for errors. Unexpected errors may occur.

General Note for URL-Encoding

When sending `POST` data with the content type `application/x-www-form-urlencoded`, certain characters must be represented by means of URL-encoding, which means substituting the character with %*hexadecimal-character-code*. The full details of when and why URL-encoding is necessary are discussed in RFC 1630, RFC 1738 and RFC 1808. Some basic details are given here. All non-ASCII characters (that is, valid ISO 8859/1 characters that are not also ASCII characters) must be URL-encoded, for example, the file `köln.html` would appear in an URL as `k%F6ln.html`.

Some characters are considered to be "unsafe" when web pages are requested by e-mail.

These characters are:

Character	URL-Encoding Syntax
the tab character	%09
the space character	%20
[	%5B
\	%5C
]	%5D
^	%5E
‘	%60
{	%7B
|	%7C
}	%7D
~	%7E

When writing URLs, you should URL-encode these characters.

Some characters have special meanings in URLs, such as the colon (:) that separates the URL scheme from the rest of the URL, the double slash (//) that indicates that the URL conforms to the Common Internet Scheme syntax and the percent sign (%). Generally, when these characters appear as parts of file names, they must be URL-encoded to distinguish them from their special meaning in URLs (this is a simplification, read the RFCs for full details).

These characters are:

Character	URL-Encoding Syntax
"	%22
#	%23
%	%25
&	%26
+	%2B
,	%2C
/	%2F
:	%3A
<	%3C
=	%3D
>	%3E
?	%3F
@	%40

Overview of Response Numbers for HTTP Requests

Status	Value	Response
STATUS_CONTINUE	100	OK to continue with request
STATUS_SWITCH_PROTOCOLS	101	Server has switched protocols in upgrade header
STATUS_OK	200	Request completed
STATUS_CREATED	201	Object created, reason = new URL
STATUS_ACCEPTED	202	Async completion (TBS)
STATUS_PARTIAL	203	Partial completion
STATUS_NO_CONTENT	204	No info to return
STATUS_RESET_CONTENT	205	Request completed, but clear form
STATUS_PARTIAL_CONTENT	206	Partial GET fulfilled
STATUS_AMBIGUOUS	300	Server could not decide what to return
STATUS_MOVED	301	Object permanently moved
STATUS_REDIRECT	302	Object temporarily moved
STATUS_REDIRECT_METHOD	303	Redirection w/o new access method
STATUS_NOT_MODIFIED	304	If-modified-since was not modified
STATUS_USE_PROXY	305	Redirection to proxy, location header specifies proxy to use
STATUS_REDIRECT_KEEP_VERB	307	HTTP/1.1: keep same verb
STATUS_BAD_REQUEST	400	Invalid syntax
STATUS_DENIED	401	Access denied
STATUS_PAYMENT_REQ	402	Payment required
STATUS_FORBIDDEN	403	Request forbidden
STATUS_NOT_FOUND	404	Object not found
STATUS_BAD_METHOD	405	Method is not allowed
STATUS_NONE_ACCEPTABLE	406	No response acceptable to client found
STATUS_PROXY_AUTH_REQ	407	Proxy authentication required
STATUS_REQUEST_TIMEOUT	408	Server timed out waiting for request
STATUS_CONFLICT	409	User should resubmit with more info
STATUS_GONE	410	The resource is no longer available
STATUS_LENGTH_REQUIRED	411	The server refused to accept request w/o a length
STATUS_PRECOND_FAILED	412	Precondition given in request failed
STATUS_REQUEST_TOO_LARGE	413	Request entity was too large

Status	Value	Response
STATUS URL_TOO_LONG	414	Request URL too long
STATUS UNSUPPORTED_MEDIA	415	Unsupported media type
STATUS SERVER_ERROR	500	Internal server error
STATUS NOT_SUPPORTED	501	"Required" not supported
STATUS BAD_GATEWAY	502	Error response received from gateway
STATUS SERVICE_UNAVAIL	503	Temporarily overloaded
STATUS GATEWAY_TIMEOUT	504	Timed out waiting for gateway
STATUS VERSION_NOT_SUP	505	HTTP version not supported

Response 301 - 303 (Redirection)

Redirection means that the requested URL has moved. As a response, the Return Header with the name LOCATION will be displayed. This header contains the URL where the requested page has moved to. A new REQUEST DOCUMENT request can be used to retrieve the page moved.

HTTP browsers redirect automatically to the new URL, but the REQUEST DOCUMENT statement does not handle redirection automatically.

Response 401 (Denied)

The response Access Denied means that the requested page can only be accessed if a valid user ID and password are provided with the request. As a response, the Return Header with the name WWW-AUTHENTICATE will be delivered with the realm needed for this request.

HTTP browsers normally display a dialog with user ID and password, but with the REQUEST DOCUMENT statement, no dialog is displayed.

Encoding of Incoming/Outgoing Data

Data transfer with the REQUEST DOCUMENT statement normally does not involve any code page conversion. If you want to have the outgoing and/or incoming data encoded in a specific code page, you can use the DATA ALL clause and/or the RETURN PAGE clause to specify this.

- DATA ALL Clause
- RETURN PAGE Clause

DATA ALL Clause

For the encoding of outgoing data, the DATA ALL clause is used:

ALL *operand6* [**ENCODED** [[**IN**] **CODEPAGE** *operand7*]]

Syntax Element Description:

Syntax Element	Description
`ALL` *`operand6`*	*`operand6`* is a complete document that is to be sent. This value is normally needed for the automatically HTTP request method `PUT` (see *Automatically Generated Headers*).
`[ENCODED [[IN] CODEPAGE` *`operand7`*`]]`	*`operand6`* will be encoded from the default code page (value of system variable `*CODEPAGE`) to the code page given in *operand7*.

RETURN PAGE Clause

For the encoding of incoming data, the `RETURN PAGE` clause is used:

[**PAGE** *operand13* [**ENCODED** [[**FOR TYPE**[**S**] *operand14*...] [**IN**] **CODEPAGE** *operand15*]]]

As a response of an HTTP/HTTPS request, incoming data may contain binary data (for example, image/gif) or character data (for example, text/html). Together with the response, the `REQUEST DOCUMENT` statement receives a parameter which specifies the type of content of the requested document (mime-type). This parameter may contain information about the code page in which the document is encoded.

This clause provides an automatic conversion to the default code page (value of system variable `*CODEPAGE`) of the Natural session.

Syntax Element Description:

Syntax Element	Description
`RETURN PAGE` *`operand13`*	No encoding at all of the returned page will be done; that is, the page will remain encoded as delivered from the http server.
`RETURN PAGE` *`operand13`* `ENCODED`	If the returned mime-type contains an encoding, *`operand13`* will be encoded from this code page to the default code page (A/B) or (U). See note below.
`RETURN PAGE` *`operand13`* `ENCODED [IN] CODEPAGE` *`operand15`*	If the returned mime-type does not contain an encoding, then *`operand13`* will be encoded from the code page defined with *`operand15`* to the default code page (value of system variable `*CODEPAGE`) (A/B) or (U).
`RETURN PAGE` *`operand13`* `[ENCODED [[FOR TYPE[S]` *`operand14...`*`] [IN] CODEPAGE` *`operand15`*`]]`	If the returned mime-type does not contain an encoding, then an additional check is performed if the returned mime-type matches one of the types given with *`operand14`*. If a match occurs, *`operand13`* will be encoded from the code page defined with *`operand15`* to the default code page (A/B) or (U).

Note:
"Returned mime-type contains an encoding" means that the http server returns a content-type header with a `charset=` clause, for example: `charset=ISO-8859-1`.

Examples

- Example 1 - General Request
- Example 2 - Simple Get Request (no data)
- Example 3 - Simple Head Request (no return page)
- Example 4 - Simple Post Request (default)
- Example 5 - Simple Put Request (with data all)

Note:
There is an example dialog `V5-RDOC` for this statement in the example library `SYSEXV`.

Example 1 - General Request

```
REQUEST DOCUMENT FROM "http://bolsap1:5555/invoke/sap.demo/handle_RFC_XML_POST"
  WITH
    USER #User PASSWORD #Password
    DATA
    NAME 'XMLData'       VALUE #Queryxml
    NAME 'repServerName' VALUE 'NT2'
  RETURN
    PAGE #Resultxml
RESPONSE #rc
```

Example 2 - Simple Get Request (no data)

```
REQUEST DOCUMENT FROM "http://pcnatweb:8080"
  RETURN
    PAGE #Resultxml
RESPONSE #rc
```

Example 3 - Simple Head Request (no return page)

```
REQUEST DOCUMENT FROM "http://pcnatweb"
RESPONSE #rc
```

Example 4 - Simple Post Request (default)

```
REQUEST DOCUMENT FROM "http://pcnatweb/cgi-bin/nwwcgi.exe/sysweb/nat-env"
  WITH
    DATA
    NAME 'XMLData'       VALUE #Queryxml
    NAME 'repServerName' VALUE 'NT2'
  RETURN
    PAGE #Resultxml
RESPONSE #rc
```

Example 5 - Simple Put Request (with data all)

```
REQUEST DOCUMENT FROM "http://pcnatweb/test.txt"
  WITH
    DATA ALL     #document
  RETURN
    PAGE #Resultxml
RESPONSE #rc
```