



HTTP Client Deux

Developer Documentation v1.0.0b02

©1998-2001 Deep Sky Technologies, Inc. All Rights Reserved.
Published and Distributed Worldwide by Deep Sky Technologies, Inc.

Deep Sky Technologies, Inc.
P.O. Box 6897
Vero Beach, FL 32961-6897
(561) 794-9494



<http://www.deepskytech.com/>

Software Engineers

James T. Crate
Robert T. McGoye
Steven G. Willis

Manual

Steven G. Willis

Software License and Limited Warranty

Please read this license carefully before using the software. By using the software, you agree to become bound by the terms of this agreement, which includes the software license and warranty disclaimer (collectively referred to herein as the "agreement"). This agreement constitutes the complete agreement between you and Deep Sky Technologies, Inc. If you do not agree to the terms of this agreement, do not use the software and promptly destroy all copies in your possession, physical and electronically.

1. Ownership of Software: The enclosed manual and computer programs ("Software") were developed and are copyrighted by Deep Sky Technologies, Inc. ("DSTi") and are licensed, not sold, to you by DSTi for use under the following terms, and DSTi reserves any rights not expressly granted to you. DSTi retains ownership of all copies of the Software itself. Neither the manual nor the Software may be copied in whole or in part except as explicitly stated below.

2. License: DSTi, as Licensor, grants to you, the Licensee, a non-exclusive, non-transferable right to use this Software subject to the terms of the license as described below:

- a. You may make backup copies of the Software for your use provided they bear the DSTi copyright notice.
- b. You may use this Software in an unlimited number of distributed copies of a single application or database. Use in additional applications requires separate licenses for the use of this Software.
- c. Distribution or dissemination of the software license serial is strictly prohibited. This license grants the original licensee sole use of the license serial for enabling this Software.

3. Restrictions: You may not distribute copies of the Software to others (except as an integral part of a database or application within the terms of this License) or electronically transfer the Software from one computer to another over a network. You may distribute copies of the Software as an integral part of a development shell or non-compiled commercial database as long as

the DSTi copyright notices and documentation remain intact with the distribution. The Software contains trade secrets and to protect them you may not decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human perceivable form. You may not modify, adapt, translate, rent, lease, loan or resell for profit the software or any part thereof.

4. Termination: This license is effective until terminated. This license will terminate immediately without notice from DSTi if you fail to comply with any of its provisions. Upon termination you must destroy the Software and all copies thereof, and you may terminate this license at any time by doing so.

5. Update Policy: DSTi may create, from time to time, updated versions of the Software. At its option, DSTi will make such updates available to the Licensee.

6. Warranty Disclaimer: The software is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. DSTi does not warrant, guarantee, or make any representations regarding the use, or the results of the use, of the software or written materials in the terms of correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of the software is assumed by the Licensee. If the software or written materials are defective you, and not DSTi or its dealers, distributors, agents, or employees, assume the entire cost of all necessary servicing, repair or correction. No oral or written information or advice given by DSTi, its dealers, distributors, agents, or employees shall create a warranty or in any way increase the scope of this warranty, and you may not rely on such information or advice. This warranty gives you specific legal rights. You may have other rights, which vary from state to state.

7. Governing Law: This agreement shall be governed by the laws of the State of Florida.

Copyrights and Trademarks

All trade names referenced in this document are the trademark or registered trademark of their respective holder.

BASh, TCP Deux, SMTP Deux, POP3 Deux, FTP Client Deux, HTTP Client Deux, and eTrans are copyright Deep Sky Technologies, Inc.

4th Dimension, ACI, ACI US, 4D Compiler, 4D, 4D Server, 4D Client, and 4D Insider are trademarks of 4D, Inc.

4D Internet Commands plugin provided courtesy, and with permission, of 4D, Inc.

Internet Commands plugin provided courtesy, and with permission, of Christian Quest.

Macintosh and MacOS are trademarks of Apple Computer, Inc.

Windows is a trademark of Microsoft Corporation.

Table of Contents

Items in **Bold** were just added in the latest release of the HTTP Client Deux component.

Items in *Italic* have not had their content filled in completely as of yet.

Items in Underline have had important updates since the last release of the HTTP Client Deux component.

Software License and Limited Warranty

Copyrights and Trademarks

Table of Contents

Preface

About this Manual

Acknowledgements

Features

System Requirements

Support

Components

Compatibility Matrix

Installing and Updating HTTP Client Deux

Managing Installation Conflicts

Affix HTTP Client Deux Document

Uninstalling HTTP Client Deux

Basics of HTTP

Overview

GET Requests

HEAD Requests

POST Requests

Responses

Response Codes

Constants

HTTPH Selectors

Methods

HTTPcd_ERROR

HTTPcd_Extract_ContentType

HTTPcd_Extract_ResponseCode

HTTPcd_GET_Simple

HTTPcd_HEAD_Simple

HTTPcd_Parse_Header_f_Response

HTTPcd_POST_Simple

HTTPcd_Retrieve_Object

INIT_HTTPcd

Version History

HTTP Client Deux v1.0.0b02
HTTP Client Deux v1.0.0b01
Using HTTP Client Deux
HTTP Client Deux Error Codes
Index

Preface

The HTTP Client Deux component is designed to work in conjunction with many other components. Specifically, the HTTP Client Deux component requires that the BASH component, available for free from DSTi, and TCP Deux component both be installed already in your database structure file.

Make certain that you view the compatibility matrix for components available to make certain you are using compatible versions of the different components required. There is a compatibility matrix available in this manual; the most recent compatibility matrix is available on the DSTi web site.

About this Manual

asd

Acknowledgements

The creation of the HTTP Client Deux component is not directly attributable to any single person. Particular pieces of functionality within the HTTP Client Deux component may be from the direct knowledge and experience of certain developers, but the overall concept and construction of the HTTP Client Deux component has come from all of the developers at Deep Sky Technologies, Inc.

In particular, the tireless efforts of Robert T. McGoye have contributed the most to the SMTP HTTP Client component. His ability, and patience, to be able to tolerate the swings in the atmosphere at DSTi, have proven to be invaluable in the development of the HTTP Client Deux component.

Later tweaks and additions to the HTTP Client Deux component have resulted from the training of James T. Crate. Mr. Crate's experience in many different programming environments has provided refreshing insights into the overall structure and organization of the core routines at DSTi, the same core routines which are available in the BASH and HTTP Client Deux components.

Finally, I, Steven G. Willis, might have had something to do with the creation of the HTTP Client Deux component...

Features

asd

System Requirements

The HTTP Client Deux component is compatible with both Macintosh and Windows installations of 4th Dimension.

Since it is a component, it does require at least version 6.7 of 4th Dimension or above, including 4D Insider v6.7 or above for installation.

Other than the normal hardware and software requirements for your version of 4th Dimension, there are no other minimum requirements for proper use of this software.

Support

Support is provided for HTTP Client Deux component free of charge for all currently licensed users. Included support services provided for all currently licensed users encompasses all of the online support services available through the DSTi web site (email, FAQ, messaging, etc.). Check the DSTi web site for current direct support options available; we are always working to offer more resources for your needs.

Contact information, including email address(es), phone number(s), and a Contact Us request form, for Deep Sky Technologies, Inc., can be found on the DSTi web site located at:

<http://www.deepskytech.com/>

If there are terms or conventions which you find difficult to understand in relation to the HTTP Client Deux component or TCP networks in general, feel free to contact Deep Sky Technologies, Inc., support. We will be more than happy to help you in any way we reasonably can. And, only through your questions do we know what subjects to include in future versions of this manual.

Components

A component groups various 4D objects (tables, project methods, forms, menu bars, variables, etc.) representing one or more additional functions. Developing a 4D component providing electronic mail functionality is one such example. A component is autonomous and must be able to be installed in any 4D structure.

Components are defined, generated, and installed with the help of 4D Insider. The component definition is based on the cross referencing analysis performed by 4D Insider (target objects and source objects).

Unlike libraries and groups, components embed the idea of security of objects that they compose. During the development phase of the component, each object is attributed an access type, "Public", "Protected" or "Private". This attribute determines whether each object will be visible or modifiable in 4th Dimension and in 4D Insider once the component is installed within a 4D database.

Compatibility Matrix

asd

Installing and Updating HTTP Client Deux

Installing HTTP Client Deux or updating an existing version of HTTP Client Deux within a 4D database is performed using 4D Insider. The activity primarily consists of installing the HTTP Client Deux component in a database structure opened with 4D Insider (installing the TCP Deux component in a library is not supported at this time).

4D Insider will manage possible conflict issues within the installation and will inform you as they are detected. Though, with the naming conventions used within the HTTP Client Deux component and the limited number of object names, conflicts should be very rare.

To install or update the HTTP Client Deux component, follow these very simple steps:

Open the uncompiled structure that you wish to install HTTP Client Deux into using 4D Insider.

Choose the "Install/Update..." command in the "Components" menu.

A standard open file dialog box will appear.

Select the HTTP Client Deux component file and click on the "Open" button.

4D Insider parses the HTTP Client Deux component and prepares to integrate it with your open database. 4D Insider will detect if the operation is an installation or an update of the HTTP Client Deux component.

In the event of a new installation, all HTTP Client Deux objects are installed.

In the event of an update, 4D Insider compares the version numbers of both the currently installing HTTP Client Deux component and the already installed HTTP Client Deux component. If the date of the "new" component is older than the already installed component,

a dialog box will alert you, allowing you to then "Continue" or "Cancel" the update.

4D Insider replaces old objects with newer objects within the HTTP Client Deux component and adds new objects from the new HTTP Client Deux component. 4D Insider takes into account "public" objects having been modified by you (e.g. "_ERROR" methods) and will prompt you to either save or replace them. If any other conflicts arise from the installation or update of the HTTP Client Deux component, 4D Insider will prompt you with an appropriate dialog box.

Save the database in 4D Insider.

Place a copy of the Affix HTTP Client Deux document in the 4DX folder.

The Affix HTTP Client Deux document (entitled **Affix_HTTPC_Deux** on Macintosh) contains essential data, resources and constants for many of the methods within the HTTP Client Deux component. For many of the methods within the HTTP Client Deux component to function properly, the Affix HTTP Client Deux document must be in the 4DX folder for the current structure.

On Macintosh, the Affix HTTP Client Deux document is entitled **Affix_HTTPC_Deux** and is located in the Mac4DX directory in the HTTP Client Deux component's archive. The document should be copied into the Mac4DX folder of your current structure. If the HTTP Client Deux component is going to be used in all of your 4D projects, the Affix HTTP Client Deux document can instead be placed within the Mac4DX folder within the 4D folder of your system.

On Windows, the Affix HTTP Client Deux document is actually two documents: **Afx_HTTPcd.4DX** and **Afx_HTTPcd.RSR**. These two documents correspond to the data fork and the resource fork of the Affix HTTP Client Deux document used on Macintosh. These documents are located in the Win4DX directory in the HTTP Client Deux component's archive. These documents

should be copied into the Win4DX folder of your current structure. If the HTTP Client Deux component is going to be used in all of your 4D projects, the Affix HTTP Client Deux document can instead be placed within the Win4DX folder within the 4D folder of your system.

For client/server installations in cross-platform environments, both the Macintosh and Windows versions of the Affix HTTP Client Deux document should be installed.

Call the method *INIT_HTTPcd* early in the On Startup database method.

To initialize the HTTP Client Deux component in your code, place a call to the method *INIT_HTTPcd* early in your **On Startup** database method.

Details about the *INIT_HTTPcd* method can be found in the method documentation section of this manual, below.

The HTTP Client Deux component is now installed/updated in your database and is listed on the "Components" page of the 4D Explorer.

Managing Installation Conflicts

On very rare occasions, when the HTTP Client Deux component is installed or updated in your 4D database, several questions and conflicts may arise. In the event of an update, 4D Insider will detect that you have modified one of more "Public" objects in HTTP Client Deux after the initial installation. Or, one or more objects of the same type and of the same name may already exist in your database and in the HTTP Client Deux component.

4D Insider detects and solves these conflicts during installation:

Modified public objects (updates only)

In this case, 4D Insider alerts you by a dialog box, allowing you to choose an update mode:

Replace the object

Replace all objects

Do not replace the object

Stop installation

Name conflicts

In this case, 4D Insider stops the HTTP Client Deux installation process, alerts you through a dialog box and saves the list of objects in conflict. This list is stored as a text file in the 4D database folder.

Naming conflicts between logical objects, such as variables, are managed by 4D Insider, in a manner that allows database compilation and avoids conflicts between HTTP Client Deux and other 4D components.

It may be necessary to rename certain objects in your database or in other components in order to be able to install the HTTP Client Deux .

If any naming conflicts do occur between HTTP Client Deux and other 4D components, please notify Deep Sky Technologies, Inc., immediately.

Affix HTTP Client Deux Document

The Affix HTTP Client Deux document (entitled **Affix_HTTPc_Deux** on Macintosh) contains essential data and resources for many of the methods within the HTTP Client Deux component. For many of the methods within the HTTP Client Deux component to function properly, the Affix HTTP Client Deux document must be in the 4DX folder for the current structure. For distributed and installed

versions of your 4D projects, the Affix HTTP Client Deux document must be available in the 4DX folder for the HTTP Client Deux methods to continue to function properly.

Note: it is best to consider the Affix HTTP Client Deux document another plug-in within your 4D project. Though there no actual plug-in calls available within the Affix HTTP Client Deux document, it does contain data and resources essential to the operation of the HTTP Client Deux methods. The HTTP Client Deux component has been designed to find the Affix HTTP Client Deux document correctly in all environments (any platform, any 4DX folder, single user or clients/server, etc.). Since the Affix HTTP Client Deux document is configured similarly to plug-ins, 4D and the HTTP Client Deux component will automatically manage the document for you in all of the possible installations of a 4D project.

Uninstalling HTTP Client Deux

4D Insider allows you to uninstall the HTTP Client Deux component from your 4D database.

To uninstall HTTP Client Deux from your 4D database:

Using 4D Insider, open your database containing the copy of HTTP Client Deux to be uninstalled.

In the "Main" listing window, select the HTTP Client Deux component.

Consider again how great the HTTP Client Deux component is and make certain that you will *really* no longer need it in your 4D database.

Select the "Uninstall..." command in the "Components" menu.

This command is only active when a component is installed in the database. A dialog box appears allowing you to confirm or cancel the operation. If you uncertain about the previous step then the cancel option is probably your best choice at this time.

Click "OK" to validate the operation.

Remove the Affix HTTP Client Deux document and internet connectivity plugins from your 4DX folder.

Remove the call to the method *INIT_HTTPcd* from your On Startup database method.

All objects from the HTTP Client Deux component are deleted from your 4D database. Obviously, you are now very sad to no longer have the HTTP Client Deux component in your 4D database. Crying is allowed...

Basics of HTTP

asd

Overview

asd

GET Requests

asd

HEAD Requests

asd

POST Requests

asd

Responses

asd

Response Codes

asd

Constants

There are a minimal number of custom constants included with the HTTP Client Deux component package. These constants are grouped into a single, convenient constant groups for easier referencing and organization.

Where appropriate, it is highly recommended that the custom constants included with the HTTP Client Deux component be utilized within your code; this will simplify considerably future feature enhancements to the core code within HTTP Client Deux.

HTTPH Selectors

Details on this constant group will be provided in a future beta release. For the current beta, these constants are not yet in use.

Methods

The following is a detailed listing of every single method call available within the HTTP Client Deux component. Each method is listed, with all parameters for calling each method, and detailed completely.

The title of most methods begin with "HTTPcd_".



HTTPcd_ERROR

HTTPcd_ERROR (*HTTPcd Error Number* ; *Special Error Text* ; *Calling Method Name*)

HTTPcd_ERROR

(
 -> *HTTPcd Error Number*: Longint
 -> *Special Error Text*: Text
 -> *Calling Method Name*: Text
)

Parameter	Type	Description
<i>HTTPcd Error Number</i>	Longint	Internal HTTPcd error number
<i>Special Error Text</i>	Text	Special text to describe the exact error instance
<i>Calling Method Name</i>	Text	Name of the method that the error condition occurred in

The method **HTTPcd_ERROR** acts as a callback method from within the HTTP Client Deux component for errors that may occur. Any time an error condition is detected within the HTTP Client Deux, a call to the method **HTTPcd_ERROR** is made.

The internal *HTTPcd Error Number* is passed to this method as the first parameter. The *Special Error Text* parameter will contain any relevant error text which is specific to the error which occurred. It is not uncommon for the *Special Error Text* value to be empty. The *Calling Method Name* will always contain the name of the HTTP

Client Deux method which called the *HTTPcd_ERROR* method.

The *HTTPcd_ERROR* method has been implemented as a source for a consistent interface and/or error tracking mechanism to be available while using the HTTP Client Deux component. This method can be modified to suit the needs of the database in which the HTTP Client Deux component has been installed.



HTTPcd_Extract_ContentType

HTTPcd_Extract_ContentType (*Referenced HTTP Response*) => *Content Type*

HTTPcd_Extract_ContentType
(
 -> *Referenced HTTP Response* : **Pointer**
)
 => *Content Type* : **Text**

Parameter	Type	Description
<i>Referenced HTTP Response</i>	Pointer	Referenced BLOB containing HTTP response
<i>Content Type</i>	Text	HTTP response code extract from referenced response

The method *HTTPcd_Extract_ContentType* will extract the content type from a referenced HTTP response.

Referenced HTTP Response is a pointer to a BLOB containing either a full HTTP response or the header of an HTTP response.

Content Type is the content type contained in *Referenced HTTP Response* . The content type of an HTTP response is the contents of the first line in the response header which begins with the string "Content-type:".



HTTPcd_Extract_ResponseCode

HTTPcd_Extract_ResponseCode (*Referenced HTTP Response*) => HTTP
Response Code

HTTPcd_Extract_ResponseCode

```
(  
    -> Referenced HTTP Response : Pointer  
)  
=> HTTP Response Code : Longint
```

Parameter	Type	Description
<i>Referenced HTTP Response</i>	Pointer	Referenced BLOB containing HTTP response
<i>HTTP Response Code</i>	Longint	HTTP response code extract from referenced response

The method **HTTPcd_Extract_ResponseCode** will extract the HTTP response code from a referenced HTTP response.

Referenced HTTP Response is a pointer to a BLOB containing either a full HTTP response or the header of an HTTP response.

HTTP Response Code is the response code contained in *Referenced HTTP Response* . If the first line of *Referenced HTTP Response* is not a well formed HTTP response or HTTP response header line then *HTTP Response Code* will be set to NULL.



HTTPcd_GET_Simple

HTTPcd_GET_Simple (*URL ; Referenced HTTP Response*) => HTTP
Response Code

HTTPcd_GET_Simple

```
(  
    -> URL : Text  
    -> Referenced HTTP Response : Pointer
```

)

=> *HTTP Response Code* : Longint

Parameter	Type	Description
<i>URL</i>	Text	Fully formatted URL to retrieve
<i>Referenced HTTP Response</i>	Pointer	Pointer to a BLOB to contain the complete HTTP response
<i>HTTP Response Code</i>	Longint	HTTP response code returned

The method *HTTPcd_GET_Simple* will run a simple HTTP GET request for a specified URL and return the response from the remote server.

The header used for the request is contained in the Affix HTTP Client Deux document, resource of type 'TEXT', resource ID # 29000. The header is a PTEXT value which has placeholders for the host name and path with parameters.

URL is the fully formatted URL to retrieve. This method does not support usernames and passwords in the HTTP request.

Referenced HTTP Response is a pointer to a BLOB which will be set to the complete response from the remote server.

HTTP Response Code is the response code contained in *Referenced HTTP Response*. If the first line of *Referenced HTTP Response* is not a well formed HTTP response or HTTP response header line then *HTTP Response Code* will be set to NULL. If there is an error in this method before retrieving the remote *URL*, then *HTTP Response Code* will be set to negative one (-1).



HTTPcd_HEAD_Simple

HTTPcd_HEAD_Simple (*URL* ; *Referenced HTTP Response*) => *HTTP Response Code*

HTTPcd_HEAD_Simple

```
(  
    -> URL : Text  
    -> Referenced HTTP Response : Pointer  
)  
=> HTTP Response Code : Longint
```

Parameter	Type	Description
<i>URL</i>	Text	Fully formatted URL to retrieve
<i>Referenced HTTP Response</i>	Pointer	Pointer to a BLOB to contain the complete HTTP response
<i>HTTP Response Code</i>	Longint	HTTP response code returned

The method *HTTPcd_HEAD_Simple* will run a simple HTTP HEAD request for a specified URL and return the response from the remote server.

The header used for the request is contained in the Affix HTTP Client Deux document, resource of type 'TEXT', resource ID # 29002. The header is a PTEXT value which has placeholders for the host name and path with parameters.

URL is the fully formatted URL to retrieve. This method does not support usernames and passwords in the HTTP request.

Referenced HTTP Response is a pointer to a BLOB which will be set to the complete response from the remote server.

HTTP Response Code is the response code contained in *Referenced HTTP Response*. If the first line of *Referenced HTTP Response* is not a well formed HTTP response or HTTP response header line then *HTTP Response Code* will be set to NULL. If there is an error in this method before retrieving the remote *URL*, then *HTTP Response Code* will be set to negative one (-1).



HTTPcd_Parse_Header_f_Response

HTTPcd_Parse_Header_f_Response (Referenced Full HTTP Response ;
Referenced HTTP Header)

HTTPcd_Parse_Header_f_Response

(
 -> Referenced Full HTTP Response : **Pointer**
 -> Referenced HTTP Header : **Pointer**
)

Parameter	Type	Description
<i>Referenced Full HTTP Response</i>	Pointer	Referenced BLOB containing complete HTTP response
<i>Referenced HTTP Header</i>	Pointer	Referenced BLOB to contain HTTP header within <i>Referenced Full HTTP Response</i>

The method *HTTPcd_Parse_Header_f_Response* will parse a complete HTTP response into discrete header and content areas.

Referenced Full HTTP Response is a pointer to a BLOB containing a complete HTTP response. After this method runs, this value will be only the HTTP response content.

Referenced HTTP Header is a pointer to a BLOB which will be set to the HTTP header parsed from *Referenced Full HTTP Response* .



HTTPcd_POST_Simple

HTTPcd_POST_Simple (URL ; Referenced Names ; Referenced Values ;
Referenced HTTP Response) => HTTP Response Code

HTTPcd_POST_Simple

(
 -> URL : **Text**
 -> Referenced Names : **Pointer**
 -> Referenced Values : **Pointer**
 -> Referenced HTTP Response : **Pointer**
)
=> HTTP Response Code : **Longint**

Parameter	Type	Description
<i>URL</i>	Text	Fully formatted URL to retrieve
<i>Referenced Names</i>	Pointer	Referenced text array containing POST names
<i>Referenced Values</i>	Pointer	Referenced text array containing POST values
<i>Referenced HTTP Response</i>	Pointer	Pointer to a BLOB to contain the complete HTTP response
<i>HTTP Response Code</i>	Longint	HTTP response code returned

The method *HTTPcd_POST_Simple* will run a simple HTTP POST request for a specified URL and return the response from the remote server.

The header used for the request is contained in the Affix HTTP Client Deux document, resource of type 'TEXT', resource ID # 29001. The header is a PTEXT value which has placeholders for the host name, path with parameters, and content length.

URL is the fully formatted URL to retrieve. This method does not support usernames and passwords in the HTTP request.

Referenced Names is a pointer to a text array containing the names of search arguments to post to the remote HTTP server. Values in the referenced array are paired with values of the same index in *Referenced Values*. All values in *Referenced Names* will be URL encoded and formatted for a proper HTTP POST request.

Referenced Values is a pointer to a text array containing the values of search arguments to post to the remote HTTP server. Values in the referenced array are paired with values of the same index in *Referenced Names*. All values in *Referenced Values* will be URL encoded and formatted for a proper HTTP POST request.

Referenced HTTP Response is a pointer to a BLOB which will be set to the complete response from the remote server.

HTTP Response Code is the response code contained in *Referenced HTTP Response* . If the first line of *Referenced HTTP Response* is not a well formed HTTP response or HTTP response header line then *HTTP Response Code* will be set to NULL. If there is an error in this method before retrieving the remote *URL* , then *HTTP Response Code* will be set to negative one (-1).



HTTPcd_Retrieve_Object

HTTPcd_Retrieve_Object (*Host Name ; Referenced Request Header ; Referenced Request Content ; Remote Port ; Local Port ; Local IP Address ; Protocol ; Timeout ; Referenced HTTP Response*)
=> *HTTP Response Code*

HTTPcd_Retrieve_Object

```
(
    -> Host Name : String [80]
    -> Referenced Request Header : Pointer
    -> Referenced Request Content : Pointer
    -> Remote Port : Longint
    -> Local Port : Longint
    -> Local IP Address : Longint
    -> Protocol : Longint
    -> Timeout : Longint
    -> Referenced HTTP Response : Pointer
)
=> HTTP Response Code : Longint
```

Parameter	Type	Description
<i>Host Name</i>	String [80]	Host name of remote HTTP server
<i>Referenced Request Header</i>	Pointer	Referenced BLOB containing complete HTTP request header
<i>Referenced Request Content</i>	Pointer	Referenced BLOB containing complete HTTP request content
<i>Remote Port</i>	Longint	Remote port to connect to
<i>Local Port</i>	Longint	Local port to connect through
<i>Local IP Address</i>	Longint	Local IP address to connect from
<i>Protocol</i>	Longint	Protocol to use for connection, HTTP or HTTPS
<i>Timeout</i>	Longint	Timeout value for HTTP session in seconds

<i>Referenced HTTP Response</i>	Pointer	Pointer to a BLOB to contain the complete HTTP response
<i>HTTP Response Code</i>	Longint	HTTP response code returned

The method *HTTPcd_Retrieve_Object* asd

Host Name is the full domain name or IP address of the remote host to connect to.

Referenced Request Header is a pointer to a BLOB containing the complete HTTP request header to send to *Host Name* .

Referenced Request Content is a pointer to a BLOB containing the complete HTTP request content to send to *Host Name* . This value can be NULL if there is no HTTP request content to send (e.g. HEAD or GET requests).

Remote Port is the remote port to connect to.

Local Port is the local port to connect through. This value is not used if Internet Commands is the current TCP plugin being used by the TCP Deux component.

Local IP Address is the local IP address on the local host to connect from. If this value is NULL then the primary IP of the local host will be used. This values is not used if Internet Commands is the current TCP plugin being used by the TCP Deux component.

Timeout is the number of second to use as the timeout value for the HTTP connection being made.

Referenced HTTP Response is a pointer to a BLOB which will be set to the complete response from the remote server.

HTTP Response Code is the response code contained in *Referenced HTTP Response* . If the first line of *Referenced HTTP Response* is not a well formed HTTP response or HTTP response header line then *HTTP Response Code* will be set to NULL. If there is an error

in this method before retrieving the remote *URL* , then *HTTP Response Code* will be set to negative one (-1).



INIT_HTTPcd

INIT_HTTPcd

INIT_HTTPcd

Parameter	Type	Description
<i>none</i>	none	none

The method *INIT_HTTPcd* initialises the HTTP Client Deux component. A single call to this method should be made early in the On Startup database method in your 4D application. Make certain the call to this method follows the initialization call to the BASH and TCP Deux components but before any other calls to the HTTP Client Deux component package.

Version History

The following is a brief version history of the HTTP Client Deux component. It details release notes, bug fixes, and changes for each version publicly available.

HTTP Client Deux v1.0.0b02

released 20010628

Changes:

Added the method `HTTPcd_Extract_ContentType`.

Fixed a bug in the method `HTTPcd_Retrieve_Object` in which when sending HTTP content in the request when using IC the request sending would fail and generate an 2903902 error in `HTTPcd_ERROR` (thanks to Marc Conner for identifying this bug).

Fixed a bug in which the local port setting would cause the streams to fail to be opened if the port was currently in use on the local machine; made it so that the plugin chooses automatically the local port for all simple API commands in this component.

Fixed problem in the method `HTTPcd_Retrieve_Object` in which the server closing the TCP stream could possibly generate an error in the HTTPcd component even though the complete response was successfully received; now, the method will not return an error in this case though the initial asynchronous receive over TCP will have a catch, and notification, if an error occurs, though subsequent TCP receives will not generate an error.

HTTP Client Deux v1.0.0b01

released 20010605

Changes:

First public release of the component.

Using HTTP Client Deux

asd

HTTP Client Deux Error Codes

asd

Index

asd