



# HTTP Log Deux

## Developer Documentation v1.0.0b02

©1998-2002 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 A. Crate  
Robert T. McGoye  
Steven G. Willis

### **Manual**

James A. Crate  
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 include 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 custom or commercial databases or applications created by the original licensee. No additional product license or royalty is required.

**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, eTrans, TCP Server Deux, HTTP Server Deux, and HTTP Log Deux 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.

Macintosh and MacOS are trademarks of Apple Computer, Inc.

Windows is a trademark of Microsoft Corporation.

# Preface

The HTTP Log Deux component is designed to work in conjunction with other components, specifically the HTTP Server Deux component from Deep Sky Technologies, Inc. The HTTP Server Deux component additionally requires other components from Deep Sky Technologies, Inc.

HTTP Log Deux provides logging for the HTTP Server Deux component. HTTP Log Deux generates a standards-based web server log file using whatever fields you have specified. This log file can be processed with any log analyzer that processes standards-based webserver log files.

# Acknowledgements

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

In particular, the tireless efforts of Steven G. Willis have contributed the most to the HTTP Log Deux component. His ability to work all night as well as the next day have greatly contributed to the speedy development of the HTTP Log Deux component.

Later additions and enhancements to the HTTP Log Deux component have resulted from the training of James A. 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.

## Features

HTTP Log Deux is a 4th Dimension component intended to work with HTTP Server Deux to provide the functionality necessary for efficient creation of web server log files. Log files are created using standard tokens, and can be processed with any log analyzer software which is capable of processing standard webserver log files.

With HTTP Log Deux, the web developer doesn't need to concern themselves with handling web server log files. They only need to specify what should go into the file and what that file's name should be. HTTP Server Deux and HTTP Log Deux will automatically do the rest!



# System Requirements

The HTTP Log 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 Log 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 Log Deux component or TCP protocols and servers 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.

# Installing and Updating HTTP Log Deux

Installing HTTP Log Deux or updating an existing version of HTTP Log Deux within a 4D database is performed using 4D Insider. The activity primarily consists of installing the HTTP Log Deux component in a database structure opened with 4D Insider (installing the HTTP Log 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 Log Deux component and the limited number of object names, conflicts should be very rare.

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

**Open the uncompiled structure that you wish to install BASH into using 4D Insider.**

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

A standard open file dialog box will appear.

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

4D Insider parses the HTTP Log 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 Log Deux component.

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

In the event of an update, 4D Insider compares the version numbers of both the currently installing HTTP Log Deux component and the already installed HTTP Log 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 Log Deux component and adds new objects from the new HTTP Log Deux component. 4D Insider takes into account "public" objects having been modified by you (e.g. "\_ERROR" meth-

ods) and will prompt you to either save or replace them. If any other conflicts arise from the installation or update of the HTTP Log Deux component, 4D Insider will prompt you with an appropriate dialog box.

### **Save the database in 4D Insider.**

### **Place a copy of the Affix HTTPId document in the 4DX folder.**

The Affix HTTPId document contains essential data and resources for many of the methods within the HTTP Log Deux component. For many of the methods within the HTTP Log Deux component to function properly, the Affix HTTPId document must be in the 4DX folder for the current structure.

On Macintosh, the Affix HTTPId document is entitled **Affix\_HTTPId\_Deux\_MacOS9.4DX** and is located in the Mac4DX directory in the Logger Deux component's archive. The document should be copied into the Mac4DX folder of your current structure. If the Logger Deux component is going to be used in all of your 4D projects, the Affix HTTPId document can instead be placed within the Mac4DX folder within the 4D folder of your system.

On Windows, the Affix HTTPId document is actually two documents: **Affix\_HTTPId\_Deux.4DX** and **Affix\_HTTPId\_Deux.RSR**. These two documents correspond to the data fork and the resource fork of the Affix HTTPId document used on Macintosh. These documents are located in the Win4DX directory in the Logger Deux component's archive. These documents should be copied into the Win4DX folder of your current structure. If the Logger Deux component is going to be used in all of your 4D projects, the Affix HTTPId 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 HTTPId document should be installed.

### **Call the method `INIT_HTTPL` early in the On Startup database method.**

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

Details about the ***INIT\_HTTPPL*** method can be found in the module and method documentation, below.

The HTTP Log 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 Log 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 Log 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 Log 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 Log 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 Log 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 Log Deux component.

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

### **Affix HTTPId Document**

The Affix HTTPId document (entitled **Affix\_HTTPId\_Deux\_MacOS9.4DX** on Macintosh; entitled **Affix\_HTTPId\_Deux.4DX** and **Affix\_HTTPId\_Deux.RSR** on Windows) contains essential data and resources for many of the methods within the HTTP Log Deux component. For many of the methods within the HTTP Log Deux component to function properly, the Affix HTTPId document must be in the 4DX folder for the current structure. For distributed and installed versions of your 4D projects, the Affix HTTPId document must be available in the 4DX folder for the HTTP Log Deux methods to continue to function properly.

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

# Uninstalling HTTP Log Deux

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

To uninstall HTTP Log Deux from your 4D database:

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

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

**Consider again how great the HTTP Log 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 HTTPId document from your 4DX folder.**

**Remove the call to the method *INIT\_HTTPPL* from your On Startup database method.**

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



# HTTP Server Deux Conventions

Throughout this manual, and all other documentation and supporting materials, included with the HTTP Log Deux component package, there is certain core knowledge which is essential to know and understand. With this knowledge, you will be able to more easily and efficiently utilize the functionality available within this software package.

If there are other terms or conventions which you find difficult to understand in relation to the HTTP Log 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.

# HTTP Log Tokens

A web server log file consists of one line for each request handled by the web server. Each line has the same pieces of information about that request. These pieces of information are called Tokens.

Web server logs can be in different formats. Some formats require certain tokens, and other formats allow any or all of the tokens.

**Common Log Format** stores certain fields in a certain order.

**Combined Log Format** same as Common Log Format, with a few additional fields.

**Extended Log Format** provides a mechanism to store any information in the log file. This mechanism is using token names in the first line to indicate what information is being stored in the log file. HTTP Log Deux supports this format, and most web log analyzer programs will also support this format.

Here is a listing of all the tokens available in the Extended Log Format:

Token Name	Description
Agent	
Bytes	
Bytes_Sent	
C-DNS	
C-IP	
Connection_ID	
CS(Cookie)	
CS-SIP	
CS-URI	
CS-URI-Query	
CS-URI-STEM	
Date	
HostField	
HostName	

Token Name	Description
Method	
Path_Args	
Referer	
Result	
SC-Status	
Search_Args	
Status	
Time	
Time_Taken	
Transfer_Time	
URL	
User	

## HTTP Log Deux Conventions

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

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

# HTTPL Tokens

The HTTPL Tokens constants group contains one constant for each of the HTTP log tokens supported in the HTTP Log Deux component. The following is a listing of the constants, and their values, within the HTTPL Tokens constant group:

Constant	Value
HTTPL_Agent	1
HTTPL_Bytes	2
HTTPL_Bytes_Sent	3
HTTPL_C_DNS	4
HTTPL_C_IP	5
HTTPL_Connection_ID	6
HTTPL_CS_Cookie	7
HTTPL_CS_SIP	8
HTTPL_CS_URI	9
HTTPL_CS_URI_Query	10
HTTPL_CS_URI_STEM	11
HTTPL_Date	12
HTTPL_HostField	13
HTTPL_HostName	14
HTTPL_Method	15
HTTPL_Path_Args	16
HTTPL_Referer	17
HTTPL_Result	18
HTTPL_SC_Status	19
HTTPL_Search_Args	20
HTTPL_Status	21
HTTPL_Time	22
HTTPL_Time_Taken	23
HTTPL_Transfer_Time	24
HTTPL_URL	25
HTTPL_User	26

# Modules

All of the code within the HTTP Log Deux component is organized into modules. Each module is designated by a three (3) to five (5) character module prefix. All of the module prefixes are used within the name of every object within the module (method names, variable names, semaphore names, etc.). This allows for the easy identification of any object within the HTTP Log Deux component.

Each module contains a set of methods which can be used throughout your database once the HTTP Log Deux component is installed. Method names all begin with the module prefix followed by an underscore ("\_") characters. The remainder of the method name then describes the function of the method.

# HTTPL Module

The HTTPL Module is for building and saving the web server log files for HTTP Server Deux.

---

## HTTPL\_ERROR

*HTTPL\_ERROR* ( *HTTPL Error Number*; *Special Error Text*; *Calling Method Name* )

*HTTPL\_ERROR*

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

	Parameter	Type	Description
	<i>HTTPL Error Number</i>	Longint	Internal HTTPL 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 **HTTPL\_ERROR** acts as a callback method from within the HTTPL module for errors that may occur. Any time an error condition is detected within the HTTPL module, a call to the method **HTTPL\_ERROR** is made.

The internal *HTTPL 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 HTTPL method which call the **HTTPL\_ERROR** method.

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

---

## HTTPL\_Get\_TokenID\_f\_Name

*HTTPL\_Get\_TokenID\_f\_Name ( Token Name ) => Token ID*

*HTTPL\_Get\_TokenID\_f\_Name*  
(  
    -> *Token Name* : Text  
)  
    => *Token ID* : Longint

	Parameter	Type	Description
	<i>Token Name</i>	Text	web server log token name
	<i>Token ID</i>	Longint	HTTP Log Deux token ID

The method ***HTTPL\_Get\_TokenID\_f\_Name*** will look up the specified log token and return the HTTP Log Deux token ID for it. See the [HTTPLTokens](#) constants group for valid tokens and their IDs.

*Token Name* is the name of the web server log token to find the HTTP Log Deux Token ID for.

*Token ID* is the HTTP Log Deux Token ID returned for *Token Name*.

---

## HTTPL\_Get\_TokenName\_f\_ID

*HTTPL\_Get\_TokenName\_f\_ID ( Token ID ) => Token Name*

*HTTPL\_Get\_TokenName\_f\_ID*  
(  
    -> *Token ID* : Longint  
)  
    => *Token Name* : Text

	Parameter	Type	Description
	<i>Token ID</i>	Longint	HTTP Log Deux token ID



	Parameter	Type	Description
	<i>Token Name</i>	Text	web server log token name

The method ***HTTPL\_Get\_TokenName\_f\_ID*** will look up the specified HTTP Log Deux token ID and return the standard log token name. See the HTTPL Tokens constants group for valid tokens and their IDs.

*Token ID* is the HTTP Log Deux Token ID to look up the name for.

*Token Name* is the standard name of the web server log token specified by *Token ID*.

---

## **HTTPL\_Log\_CurrentProcess**

*HTTPL\_Log\_CurrentProcess*

*HTTPL\_Log\_CurrentProcess*

	Parameter	Type	Description
	<i>(none)</i>	n/a	n/a

The method ***HTTPL\_Log\_CurrentProcess*** is used internally by the HTTP Server Deux component.

**NOTE:** this method should never be called by the end developer.

---

## **HTTPL\_qi\_Log\_Running**

*HTTPL\_qi\_Log\_Running* => *qi Log Running*

*HTTPL\_qi\_Log\_Running*  
=> *qi Log Running* : Longint

	Parameter	Type	Description
	<i>qi Log Running</i>	Longint	qi for whether HTTP Log Deux log writing process is currently running

The method **HTTPL\_qi\_Log\_Running** returns an indicator for whether the HTTP Log Deux log writing process is currently running.

*qi Log Running* is the indicator for whether the HTTPL Log Deux log process is currently running. *qiLogRunning* will be set to one (1) if the process is currently running, or will be set to zero (0) if it is not running.

---

## HTTPL\_Set\_Preferences

**HTTPL\_Set\_Preferences** ( *Log Document Full Path* ; *Referenced Log Tokens* ) => *Success Code*

**HTTPL\_Set\_Preferences**  
(  
    -> *Log Document Full Path* : Text  
    -> *Referenced Log Tokens* : Pointer  
)  
=> *Success Code* : Longint

	Parameter	Type	Description
	<i>Log Document Full Path</i>	Text	Full path of log file document
	<i>Referenced Log Tokens</i>	Pointer	Referenced array of token IDs to save in log file
	<i>Success Code</i>	Longint	qi for whether preferences are set successfully

The method **HTTPL\_Set\_Preferences** will set the preferences for the HTTP Log Deux component.

**Note:** TCP Server Deux cannot be currently running for this method to work.

*Log Document Full Path* is the full path for the log document.

*Referenced Log Tokens* is a referenced longint array containing the HTTP Log Deux Token IDs specifying what information should be logged. The tokens will be written into the log in the exact order of the tokens in *Referenced Log Tokens*. See the [HTTPL Tokens](#) constants group for valid tokens and their IDs.

*Success Code* is the indicator for whether the HTTP Log Deux preferences were set successfully. *Success Code* will be set to one (1) if the preferences were set successfully, or set to zero (0) if the preferences were not set successfully.

---

## **HTTPL\_Start\_Logging**

*HTTPL\_Start\_Logging* => *Success Code*

*HTTPL\_Start\_Logging*  
=> *Success Code* : Longint

	Parameter	Type	Description
	<i>Success Code</i>	Longint	qi for whether HTTP Log Deux process started successfully

The method ***HTTPL\_Start\_Logging*** starts the HTTP Log Deux logging process and returns an indicator for whether it started successfully.

*Success Code* is the indicator for whether the HTTP Log Deux log process was started successfully. *Success Code* will be set to one (1) if the process was started, or will be set to zero (0) if it was not started.

---

## **HTTPL\_Stop\_Logging**

*HTTPL\_Stop\_Logging*

*HTTPL\_Stop\_Logging*

	Parameter	Type	Description
	<i>(none)</i>	n/a	n/a

The method ***HTTPL\_Stop\_Logging*** stops the HTTP Log Deux logging process.

---

## INIT\_HTTPPL

*INIT\_HTTPPL*

*INIT\_HTTPPL*

	Parameter	Type	Description
	(none)	n/a	n/a

The method ***INIT\_HTTPPL*** initializes the HTTP Log Deux process. This method should be called early in the **On Startup** database method. However, it should go after the calls to initialize BASH, TCP Deux, TCP Server Deux and HTTP Server Deux.

# Version History

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

# HTTP Log Deux v1.0.0b02

*released 20020115*

## **Changes:**

First public beta release of the component.

# HTTP Log Deux Error Codes

Coming soon...