
Juphoon Protocol Framework

HTTP Client

Published: Jan 2008

For more information on Juphoon Protocol Framework, see <http://www.juphoon.com>

Juphoon HTTP Client Function Definition

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87304379

Fax: +86-574-87304379

Text Part Number:

Copyright © 2008, Juphoon System Software Corporation.

All rights reserved.

Contents

1. INTRODUCTION.....	5
1.1 JUPHOON HTTP CLIENT	5
1.2 AUDIENCE	5
1.3 SCOPE.....	5
1.4 ABBREVIATIONS	5
2. SYSTEM ENVIRONMENT.....	5
2.1 BASIC DATA TYPES	6
3. USER INTERFACES	7
3.1 STRUCTURES	7
3.1.1 <i>ST_HTTP_MSG</i>	7
3.1.2 <i>ST_ZOS_DBUF</i>	7
3.1.3 <i>ST_ZOS_SSTR</i>	8
3.1.4 <i>ST_ZOS_SYS_TIME</i>	8
3.1.5 <i>ST_ZOS_SSTR</i>	8
3.1.6 <i>ST_HTTP_DATE</i>	8
3.1.7 <i>ST_ZOS_LSSTR</i>	9
3.1.8 <i>ST_ZOS_DBUF</i>	9
3.1.9 <i>ST_HTTP_DIGEST_RSP</i>	9
3.1.10 <i>ST_HTTP_REQ_URI</i>	10
3.1.11 <i>ST_HTTP_ALGO</i>	10
3.1.12 <i>ST_HTTP_DIG_RSP</i>	11
3.1.13 <i>ST_HTTP_CREDENTIALS</i>	11
3.1.14 <i>ST_HTTP_CHALLENGE</i>	12
3.1.15 <i>ST_HTTP_REQ_URI</i>	12
3.1.16 <i>ST_ZOS_INET_ADDR</i>	12
3.1.17 <i>PFN_HTTPPROCSTAT</i>	13
3.1.18 <i>PFN_HTTPPROCMSG</i>	13
3.1.19 <i>PFN_HTTPPROCHDRS</i>	13
3.1.20 <i>PFN_HTTPPROCBODY</i>	13
3.2 UTILITY INTERFACES	13
3.2.1 <i>Http_MsgCreate</i>	13
3.2.2 <i>Http_MsgDelete</i>	14
3.2.3 <i>Http_MsgClone</i>	14
3.2.4 <i>Http_MsgDecode</i>	15
3.2.5 <i>Http_MsgEncode</i>	15
3.2.6 <i>Http_MsgGetMethod</i>	15
3.2.7 <i>Http_MsgAddReqLine</i>	16
3.2.8 <i>Http_MsgAddStatLine</i>	16
3.2.9 <i>Http_MsgAddAcpt</i>	17
3.2.10 <i>Http_MsgAddAcptLang</i>	17
3.2.11 <i>Http_MsgAddAcptEncoding</i>	18

3.2.12	<i>Http_MsgAddAcptRange</i>	18
3.2.13	<i>Http_MsgAddAllow</i>	19
3.2.14	<i>Http_MsgAddCacheCtrl</i>	19
3.2.15	<i>Http_MsgAddConn</i>	20
3.2.16	<i>Http_MsgAddContentEncoding</i>	20
3.2.17	<i>Http_MsgAddContentType</i>	21
3.2.18	<i>Http_MsgAddContentTypeX</i>	21
3.2.19	<i>Http_MsgAddDate</i>	22
3.2.20	<i>Http_MsgAddEtag</i>	22
3.2.21	<i>Http_MsgAddExpire</i>	23
3.2.22	<i>Http_MsgAddHostByName</i>	23
3.2.23	<i>Http_MsgAddHostByIpv4</i>	24
3.2.24	<i>Http_MsgAddHostByIpv6</i>	24
3.2.25	<i>Http_MsgAddIfMatch</i>	25
3.2.26	<i>Http_MsgAddIfNoMatch</i>	25
3.2.27	<i>Http_MsgAddIfMdfySince</i>	26
3.2.28	<i>Http_MsgAddIfUnmdfySince</i>	26
3.2.29	<i>Http_MsgAddLastMdfy</i>	27
3.2.30	<i>Http_MsgAddLocat</i>	27
3.2.31	<i>Http_MsgAddRange</i>	28
3.2.32	<i>Http_MsgAddRefer</i>	28
3.2.33	<i>Http_MsgAddServer</i>	29
3.2.34	<i>Http_MsgAddTrsfEncoding</i>	29
3.2.35	<i>Http_MsgAddUserAgent</i>	30
3.2.36	<i>Http_MsgAddX3gppIntendId</i>	30
3.2.37	<i>Http_MsgAddBody</i>	31
3.2.38	<i>Http_MsgAddBodyX</i>	31
3.2.39	<i>Http_ParmFillDRspUserName</i>	32
3.2.40	<i>Http_ParmFillDRspRealm</i>	32
3.2.41	<i>Http_ParmFillDRspNonce</i>	33
3.2.42	<i>Http_ParmFillDRspUri</i>	33
3.2.43	<i>Http_ParmFillDRspRsp</i>	34
3.2.44	<i>Http_ParmFillDRspOpaque</i>	35
3.2.45	<i>Http_ParmFillDRspAlgo</i>	35
3.2.46	<i>Http_ParmDRspLstAdd</i>	36
3.2.47	<i>Http_ParmDRspLstRmv</i>	36
3.2.48	<i>Http_ParmFillCredents</i>	37
3.2.49	<i>Http_ParmCalcA1</i>	37
3.2.50	<i>Http_ParmCalcA2</i>	38
3.2.51	<i>Http_ParmCalcKd</i>	39
3.2.52	<i>Http_GetMethodDesc</i>	39
3.3	USERFUL INTERFACES	39
3.3.1	<i>Httpc_Open</i>	40
3.3.2	<i>Httpc_OpenX</i>	40
3.3.3	<i>Httpc_Close</i>	41

3.3.4	<i>Httpc_Conn</i>	41
3.3.5	<i>Httpc_ConnX</i>	42
3.3.6	<i>Httpc_Disc</i>	42
3.3.7	<i>Httpc_Send</i>	43
3.3.8	<i>Httpc_SendData</i>	43
3.3.9	<i>Httpc_SendDataX</i>	44
3.3.10	<i>Httpc_GetUserId</i>	44
3.3.11	<i>Httpc_SetUserId</i>	44
3.3.12	<i>Httpc_GetRmtAddr</i>	45
3.3.13	<i>Httpc_SetRmtAddr</i>	45
3.4	CONFIG INTERFACES.....	46
3.4.1	<i>Httpc_CfgGetTaskPriority</i>	46
3.4.2	<i>Httpc_CfgGetTaskStackSize</i>	46
3.4.3	<i>Httpc_CfgGetTaskQueueSize</i>	47
3.4.4	<i>Httpc_CfgGetTaskTimerNum</i>	47
3.4.5	<i>Httpc_CfgGetLogLevel</i>	47
3.4.6	<i>Httpc_CfgGetSessNum</i>	48
3.4.7	<i>Httpc_CfgSetTaskPriority</i>	48
3.4.8	<i>Httpc_CfgSetTaskStackSize</i>	48
3.4.9	<i>Httpc_CfgSetTaskQueueSize</i>	49
3.4.10	<i>Httpc_CfgSetTaskTimerNum</i>	49
3.4.11	<i>Httpc_CfgSetLogLevel</i>	50
3.4.12	<i>Httpc_CfgSetSessNum</i>	50
3.5	TASK INTERFACES.....	50
3.5.1	<i>Httpc_Start</i>	50
3.5.2	<i>Httpc_Stop</i>	51
3.6	VERSION INTERFACES.....	51
3.6.1	<i>Httpc_GetVersion</i>	51

List of Tables

TABLE 2-1: BASIC DATA TYPES	6
-----------------------------------	---

List of Figures

FIGURE 2-1 HTTP CLIENT.....	6
-----------------------------	---

1. Introduction

1.1 Juphoon HTTP Client

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

One of the most obvious applications of HTTP is the creation of Web servers and clients. This is software such as the Mozilla Web browser (HTTP client) and the Apache Web server (HTTP server). Juphoon HTTP Client is used by user applications to perform Web client functions. Juphoon HTTP Client is strictly adhere to the standard and is compatible (can interoperate) with all existing software.

1.2 Audience

The readers of this document are assumed to have a working knowledge of the HTTP Client.

1.3 Scope

This document provides the definitions of interfaces provided by Juphoon HTTP Client to users. Details on the realization of the HTTP Client are not within the scope of this document.

1.4 Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
HTTP	Hypertext Transfer Protocol
STUN	Simple Traversal of User Datagram Protocol Through Network Address Translators
URI	Uniform Resource Identifier
ZOS	Zero Operating System

2. System Environment

This section describes the environment in which Juphoon HTTP Client is designed to operate. Figure 2-1 illustrates the framework.

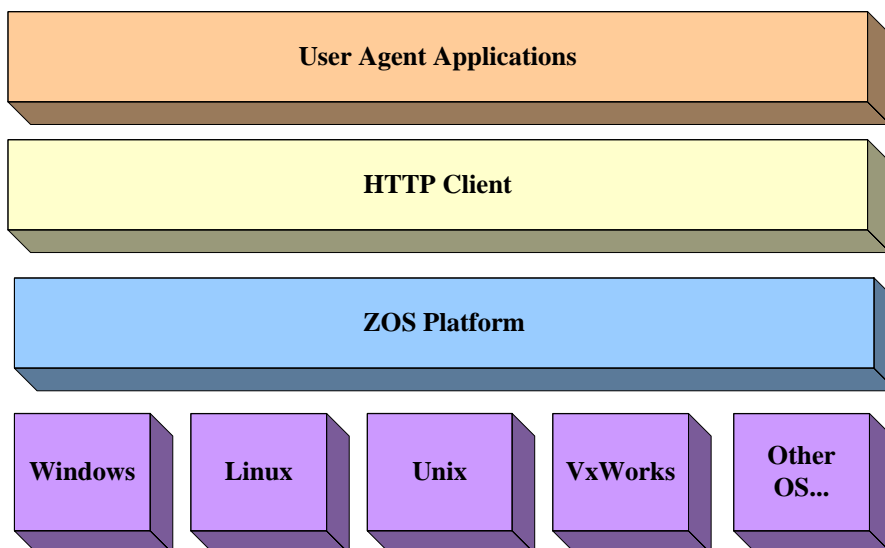


Figure 2-1 HTTP Client

2.1 Basic Data Types

There are some basic data types provided by ZOS platform. Table 2-1 lists these types used by Juphoon DNS Resolver.

Name	Type
ZDOUBLE	double
ZFLOAT	float
ZLONG	long
ZINT	int
ZSHORT	short
ZCHAR	char
ZULONG	unsigned long
ZUINT	unsigned int
ZSIZE_T	unsigned int
ZUSHORT	unsigned short
ZUCHAR	unsigned char
ZBOOL	int
ZVOID	void

Table 2-1: Basic Data Types

3. User Interfaces

3.1 Structures

3.1.1 ST_HTTP_MSG

```
typedef struct tagHTTP_MSG
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucReqPres;            /* Request present flag */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    ST_ZOS_DBUF *pstMemBuf;      /* memory buffer */
    ST_ZOS_DBUF *pstMsgBuf;      /* message buffer */
    ST_ZOS_LSSTR stMsgStr;       /* message string(decode used) */
    union
    {
        ST_HTTP_REQ_LINE stReqLine; /* Request Line */
        ST_HTTP_STAT_LINE stStatLine; /* Status Line */
    } u;
    ST_HTTP_MSG_HDR_LST stHdrLst; /* message-header list */
    ST_HTTP_BODY stBody;          /* message-body */
} ST_HTTP_MSG;
```

3.1.2 ST_ZOS_DBUF

```
typedef struct tagZOS_DBUF
{
    struct tagZOS_DBUF *pstNext; /* next data buffer */
    ZPOOLID zPoolId;            /* memory pool id for dbuf alloc */
    ZULONG dwBufLen;            /* buffer used length */
    ZULONG dwDftBlkSize;        /* default data block size in buffer */
    ZUCHAR ucBufType;           /* buffer mode ZDBUF_TYPE_BYTE... */
    ZUCHAR ucRefCnt;            /* buffer reference count */
    ZUCHAR aucSpare[2];         /* for 32 bit alignment */
#ifdef ZOS_SUPT_DUMP
    ZDUMPID zDumpId;           /* stack dump */
#endif
    ST_ZOS_DBUF_DATA *pstHead; /* the first data block in buffer */
    ST_ZOS_DBUF_DATA *pstTail; /* the last data block in buffer */
} ST_ZOS_DBUF;
```

3.1.3 ST_ZOS_SSTR

```
typedef struct tagZOS_SSTR
{
    ZCHAR *pcStr;           /* string address */
    ZUSHORT wLen;          /* string length */
    ZUCHAR aucSpare[2];    /* for 32 bit alignment */
} ST_ZOS_SSTR;
```

3.1.4 ST_ZOS_SYS_TIME

```
typedef struct tagZOS_SYS_TIME
{
    ZUSHORT wYear;         /* year */
    ZUCHAR ucMonth;       /* month 1 - 12 */
    ZUCHAR ucDate;        /* date 1 - 31 */
    ZUCHAR ucWeekDay;     /* week day 1 - 7 */
    ZUCHAR ucHour;        /* hour 0 - 23 */
    ZUCHAR ucMinute;      /* minute 0 - 59 */
    ZUCHAR ucSecond;      /* second 0 - 59 */
} ST_ZOS_SYS_TIME;
```

3.1.5 ST_ZOS_SSTR

```
typedef struct tagZOS_SSTR
{
    ZCHAR *pcStr;           /* string address */
    ZUSHORT wLen;          /* string length */
    ZUCHAR aucSpare[2];    /* for 32 bit alignment */
} ST_ZOS_SSTR;
```

3.1.6 ST_HTTP_DATE

```
typedef struct tagHTTP_DATE
{
    ZUSHORT wYear;         /* date1 year */
    ZUCHAR ucMonth;       /* date1 month EN_HTTP_MONTH */
    ZUCHAR ucDay;         /* date1 day */
    ZUCHAR ucWkDay;       /* wkday EN_HTTP_WKDAY */
    ZUCHAR ucHour;        /* time hour */
    ZUCHAR ucMinute;      /* time minute */
    ZUCHAR ucSecond;      /* second */
    ST\_ZOS\_SSTR stTimeZone; /* timer zone */
} ST_HTTP_DATE;
```

3.1.7 ST_ZOS_LSSTR

```
typedef struct tagZOS_LSSTR
{
    ZCHAR *pcStr;           /* string address */
    ZULONG dwLen;          /* string length */
} ST_ZOS_LSSTR;
```

3.1.8 ST_ZOS_DBUF

```
typedef struct tagZOS_DBUF
{
    struct tagZOS_DBUF *pstNext; /* next data buffer */
    ZPOOLID zPoolId;             /* memory pool id for dbuf alloc */
    ZULONG dwBufLen;             /* buffer used length */
    ZULONG dwDftBlkSize;         /* default data block size in buffer */
    ZUCHAR ucBufType;            /* buffer mode ZDBUF_TYPE_BYTE... */
    ZUCHAR ucRefCnt;             /* buffer reference count */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
#ifdef ZOS_SUPT_DUMP
    ZDUMPID zDumpId;             /* stack dump */
#endif
    ST_ZOS_DBUF_DATA *pstHead;   /* the first data block in buffer */
    ST_ZOS_DBUF_DATA *pstTail;   /* the last data block in buffer */
} ST_ZOS_DBUF;
```

3.1.9 ST_HTTP_DIGEST_RSP

```
typedef struct tagHTTP_DIGEST_RSP
{
    ST_HTTP_DIG_RSP_LST stDigRspLst; /* dig-resp list */
} ST_HTTP_DIGEST_RSP;
```

3.1.10 ST_HTTP_REQ_URI

```
typedef struct tagHTTP_REQ_URI
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucStrPres;             /* string present flag */
    ZUCHAR ucType;                /* Request-URI type EN_HTTP_REQ_URI */
    ZUCHAR aucSpare[1];           /* for 32 bit alignment */
    ST_ZOS_SSTR stStr;            /* request uri string(decode used) */
    union
    {
        ST_HTTP_ABSO_URI stAbsoUri; /* absoluteURI */
        ST_HTTP_ABS_PATH stAbsPath; /* abs-path */
        ST_HTTP_AUTHOR stAuthor;    /* authority */
    } u;
} ST_HTTP_REQ_URI;
```

3.1.11 ST_HTTP_ALGO

```
typedef struct tagHTTP_ALGO
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucType;                /* algorithm type EN_HTTP_ALGO */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_ZOS_SSTR stOther;          /* other algorithm */
} ST_HTTP_ALGO;
```

3.1.12 ST_HTTP_DIG_RSP

```

typedef struct tagHTTP_DIG_RSP
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucType;                /* dig-resp type EN_HTTP_DIG_RSP */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_ZOS_SSTR stUserName;   /* username */
        ST_ZOS_SSTR stRealm;      /* realm */
        ST_ZOS_SSTR stNonce;      /* nonce */
        ST_HTTP_REQ_URI stDigestUri; /* digest-uri */
        ST_ZOS_SSTR stDresp;      /* dresponse */
        ST_HTTP_ALGO stAlgo;      /* algorithm */
        ST_ZOS_SSTR stCnonce;     /* cnonce */
        ST_ZOS_SSTR stOpaque;     /* opaque */
        ST_HTTP_TYPE_VAL stMsgQop; /* message-qop:qop-value EN_HTTP_QOP_VAL */
        ST_ZOS_SSTR stNonceCount; /* nonce-count */
        ST_HTTP_GEN_VAL stAuthParm; /* auth-param */
    } u;
} ST_HTTP_DIG_RSP;

```

3.1.13 ST_HTTP_CREDENTIALS

```

typedef struct tagHTTP_CREDENTIALS
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucDigestRspPres;      /* digest-response present flag */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_HTTP_DIGEST_RSP stDigestRsp; /* digest-response */
        ST_HTTP_OTHER_RSP stOther;      /* other-response */
    } u;
} ST_HTTP_CREDENTIALS;

```

3.1.14 ST_HTTP_CHALLENGE

```

typedef struct tagHTTP_CHALLENGE
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucDigestPres;          /* digest present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    union
    {
        ST_HTTP_DIGEST stDigest; /* digest */
        ST_HTTP_OTHER_CHALLENGE stOther; /* other-challenge */
    } u;
} ST_HTTP_CHALLENGE;

```

3.1.15 ST_HTTP_REQ_URI

```

typedef struct tagHTTP_REQ_URI
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucStrPres;             /* string present flag */
    ZUCHAR ucType;                /* Request-URI type EN_HTTP_REQ_URI */
    ZUCHAR aucSpare[1];           /* for 32 bit alignment */
    ST_ZOS_SSTR stStr;            /* request uri string(decode used) */
    union
    {
        ST_HTTP_ABSO_URI stAbsoUri; /* absoluteURI */
        ST_HTTP_ABS_PATH stAbsPath; /* abs-path */
        ST_HTTP_AUTHOR stAuthor;    /* authority */
    } u;
} ST_HTTP_REQ_URI;

```

3.1.16 ST_ZOS_INET_ADDR

```

typedef struct tagZOS_INET_ADDR
{
    ZUCHAR ucType;                /* ZINET_IPV4... */
    ZUCHAR aucSpare[1];           /* for 32 bit alignment */
    ZUSHORT wPort;                /* not order[host or n/w] dependent */
    union
    {
        ZULONG dwIp;              /* not order[host or n/w] dependent */
        ZUCHAR aucIp[ZINET_IPV4_ADDR_SIZE]; /* ipv4 address */
        ZUCHAR aucIpv6[ZINET_IPV6_ADDR_SIZE]; /* ipv6 address */
    } u;
} ST_ZOS_INET_ADDR;

```

3.1.17 PFN_HTTPCPROCSTAT

The prototype of the callback function used by HTTP client when it status changed.

```

/* http event status of session */
typedef enum EN_HTTPC_SESS_STAT_TYPE
{
    EN_HTTPC_SESS_STAT_CONN_ERR,      /* transport connect error */
    EN_HTTPC_SESS_STAT_ACPTED,       /* transport accepted */
    EN_HTTPC_SESS_STAT_CONNING,      /* transport connecting */
    EN_HTTPC_SESS_STAT_CONNED,       /* transport connected */
    EN_HTTPC_SESS_STAT_DISCED,       /* transport disconnected */
    EN_HTTPC_SESS_STAT_SEND_ERR,     /* send message error */
    EN_HTTPC_SESS_STAT_RECV_ERR,     /* receive message error */
} EN_HTTPC_SESS_STAT_TYPE;

typedef ZINT (*PFN_HTTPCPROCSTAT) (ZULONG dwSessId, ZUCHAR ucStatType);

```

3.1.18 PFN_HTTPCPROCMSG

The prototype of the callback function used by HTTP client when it received the message.

```

typedef ZINT (*PFN_HTTPCPROCMSG) (ZULONG dwSessId, ST\_HTTP\_MSG *pstMsg);

```

3.1.19 PFN_HTTPCPROCHDRS

The prototype of the callback function used by HTTP client when it received the message headers.

```

typedef ZINT (*PFN_HTTPCPROCHDRS) (ZULONG dwSessId, ST\_HTTP\_MSG *pstMsg,
    ZBOOL bHasBody);

```

3.1.20 PFN_HTTPCPROCBODY

The prototype of the callback function used by HTTP client when it received the message body.

```

typedef ZINT (*PFN_HTTPCPROCBODY) (ZULONG dwSessId, ST\_ZOS\_DBUF *pstBody,
    ZBOOL bLastBody);

```

3.2 Utility Interfaces

These interfaces are included in `http_msg_util.h`.

3.2.1 Http_MsgCreate

Creates a HTTP message.

```
ZINT Http_MsgCreate(ST\_HTTP\_MSG **ppstMsg);
```

[Parameters]

Input parameters:

None.

Output parameters:

[ST_HTTP_MSG](#) **ppstMsg

The newly created HTTP message.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.2 Http_MsgDelete

Deletes a HTTP message.

```
ZINT Http_MsgDelete(ST\_HTTP\_MSG *pstMsg);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message to delete.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.3 Http_MsgClone

Clones a message.

```
ZINT Http_MsgClone(ST\_HTTP\_MSG *pstSrcMsg, ST\_HTTP\_MSG **ppstDstMsg);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message to be cloned.

Output parameters:

[ST_HTTP_MSG](#) **ppstDstMsg

The cloned HTTP message.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.4 Http_MsgDecode

Decodes the information in a data buffer into a HTTP message.

```
ZINT Http_MsgDecode(ST\_ZOS\_DBUF *pstMsgBuf, ST\_HTTP\_MSG **ppstMsg);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMsgBuf

The data buffer where the information is.

Output parameters:

[ST_HTTP_MSG](#) **ppstMsg

The decoded message.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.5 Http_MsgEncode

Encodes a HTTP message event and puts the encoded information in a data buffer.

```
ZINT Http_MsgEncode(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_DBUF **ppstMsgBuf);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

Output parameters:

[ST_ZOS_DBUF](#) **ppstMsgBuf

The data buffer where the encoded information is.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.6 Http_MsgGetMethod

Gets the method of a HTTP message.

```
ZINT Http_MsgGetMethod(ST\_HTTP\_MSG *pstMsg, ZUCHAR *pucMethod);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

Output parameters:

ZUCHAR *pucMethod

Method of the message.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.7 Http_MsgAddReqLine

Adds a request line into a HTTP message.

```
ZINT Http_MsgAddReqLine(ST\_HTTP\_MSG *pstMsg, ZUCHAR ucMethod,  
                        ST\_ZOS\_SSTR *pstUri);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZUCHAR ucMethod

Method in the request line.

[ST_ZOS_SSTR](#) *pstUri

The request URI in the request line.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.8 Http_MsgAddStatLine

Adds a status line into a HTTP message.

```
ZINT Http_MsgAddStatLine(ST\_HTTP\_MSG *pstMsg, ZULONG dwStatCode);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZULONG dwStatCode

The status code.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.9 Http_MsgAddAcpt

Adds the Accept header into a HTTP request message.

```
ZINT Http_MsgAddAcpt(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcMediaTypes);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcMediaTypes

The media range.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.10 Http_MsgAddAcptLang

Adds the Accept-Language header into a HTTP request message.

```
ZINT Http_MsgAddAcptLang(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcLangs);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcLangs

The user's language preferences.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.11 Http_MsgAddAcptEncoding

Adds the Accept-Encoding header into a HTTP request message.

```
ZINT Http_MsgAddAcptEncoding(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcCodings);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcCodings

Types of encoding the browser has the capability to decode.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.12 Http_MsgAddAcptRange

Adds the Accept-Range header into a response message.

```
ZINT Http_MsgAddAcptRange(ST\_HTTP\_MSG *pstMsg, ZUCHAR ucRange);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZUCHAR ucRange

The range type.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.13 Http_MsgAddAllow

Adds the Allow header to a HTTP message.

```
ZINT Http_MsgAddAllow(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcMethods);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcMethods

The list of request methods.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.14 Http_MsgAddCacheCtrl

Adds the Cache-Control header into a HTTP message.

```
ZINT Http_MsgAddCacheCtrl(ST\_HTTP\_MSG *pstMsg, ZUCHAR ucDirect);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZUCHAR ucDirect

The cache directive.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.15 Http_MsgAddConn

Adds the Connection header into a HTTP message.

```
ZINT Http_MsgAddConn(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcOpt);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcOpt

The connection option.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.16 Http_MsgAddContentEncoding

Adds the Content-Encoding header into a HTTP message.

```
ZINT Http_MsgAddContentEncoding(ST\_HTTP\_MSG *pstMsg,  
                                ZCHAR *pcCodings);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcCodings

Codings that have been performed on the resource.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.17 Http_MsgAddContentType

Adds the Content-Type header into a HTTP message.

```
ZINT Http_MsgAddContentType(ST\_HTTP\_MSG *pstMsg, ZUCHAR ucMtype,  
                             ZUCHAR ucMSubtype);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZUCHAR ucMtype

The m-type.

ZUCHAR ucMSubtype

The m-subtype.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.18 Http_MsgAddContentTypeX

Adds the Content-Type which includes a list of medium into a HTTP message.

```
ZINT Http_MsgAddContentTypeX(ST\_HTTP\_MSG *pstMsg,  
                              ZCHAR *pcMediaType);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcMediaType

The list of medium.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.19 Http_MsgAddDate

Adds the Date header into a HTTP message.

```
ZINT Http_MsgAddDate(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_SYS\_TIME *pstDate);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SYS_TIME](#) *postdate

ZOS system time.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.20 Http_MsgAddEtag

Adds the ETag header into a HTTP response message.

```
ZINT Http_MsgAddEtag(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_SSTR *pstTag);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstTag

The tag.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.21 Http_MsgAddExpire

Adds the Expires header into a HTTP message.

```
ZINT Http_MsgAddExpire(ST\_HTTP\_MSG *pstMsg, ST\_HTTP\_DATE *pstDate);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_HTTP_DATE](#) *postdate

The date which specifies when the receiving Web agent believes the resource to be invalid for some reason.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.22 Http_MsgAddHostByName

Adds the Host header into a HTTP request message.

```
ZINT Http_MsgAddHostByName(ST\_HTTP\_MSG *pstMsg,  
                            ST\_ZOS\_SSTR *pstName, ZULONG dwPort);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstName

The host name.

ZULONG dwPort

The host port, 0 for default.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.23 Http_MsgAddHostByIpv4

Adds the Host header into a HTTP request message.

```
ZINT Http_MsgAddHostByIpv4(ST\_HTTP\_MSG *pstMsg, ZULONG dwIpv4,  
                           ZULONG dwPort);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZULONG dwIpv4

The host's IPv4 address.

ZULONG dwPort

The host port, 0 for default.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.24 Http_MsgAddHostByIpv6

Adds the Host header into a HTTP request message.

```
ZINT Http_MsgAddHostByIpv6(ST\_HTTP\_MSG *pstMsg, ZUCHAR *pucIpv6,  
                           ZULONG dwPort);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZUCHAR *pucIpv6

The host's IPv6 address.

ZULONG dwPort

The host port, 0 for default.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.25 Http_MsgAddIfMatch

Adds the If-Match header into a HTTP request message.

```
ZINT Http_MsgAddIfMatch(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_SSTR *pstMatchTag);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstMatchTag

The match tag.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.26 Http_MsgAddIfNoMatch

Adds the If-None-Match header into a HTTP request message.

```
ZINT Http_MsgAddIfNoMatch (ST HTTP MSG *pstMsg,  
                           ST\_ZOS\_SSTR *pstNoMatchTag);
```

[Parameters]

Input parameters:

[ST HTTP MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstNoMatchTag

The entity tag.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.27 Http_MsgAddIfMdfySince

Adds the If-Modify-Since header into a HTTP request message.

```
ZINT Http_MsgAddIfMdfySince (ST HTTP MSG *pstMsg,  
                             ST HTTP DATE *pstDate);
```

[Parameters]

Input parameters:

[ST HTTP MSG](#) *pstMsg

The HTTP message.

[ST HTTP DATE](#) *postdate

The date that is used in a comparison performed by the Web server.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.28 Http_MsgAddIfUnmdfySince

Adds the If-Unmodified-Since header into a HTTP request header.

```
ZINT Http_MsgAddIfUnmdfySince (ST HTTP MSG *pstMsg,  
                               ST HTTP DATE *pstDate);
```

[Parameters]

Input parameters:

[ST HTTP MSG](#) *pstMsg

The HTTP message.

[ST HTTP DATE](#) *postdate

The data which indicates that since when the resource has been modified.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.29 Http_MsgAddLastMdfy

Adds the Last-Modified header into a HTTP message.

```
ZINT Http_MsgAddLastMdfy(ST HTTP MSG *pstMsg, ST HTTP DATE *pstDate);
```

[Parameters]

Input parameters:

[ST HTTP MSG](#) *pstMsg

The HTTP message.

[ST HTTP DATE](#) *postdate

The data which is used in many calculations to determine the age of a resource, especially by caching systems.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.30 Http_MsgAddLocat

Adds the Location header into a HTTP response message.

```
ZINT Http_MsgAddLocat(ST HTTP MSG *pstMsg, ST ZOS SSTR *pstUri);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstUri

The URI the Web client is intended to use to receive the resource being request.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.31 Http_MsgAddRange

Adds the Range header to a HTTP request message.

```
ZINT Http_MsgAddRange(ST\_HTTP\_MSG *pstMsg, ZULONG dwFirstPos,  
                      ZULONG dwLastPos);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZULONG dwFirstPos

Position of the first byte requested.

ZULONG dwLastPos

Position of the last byte requested.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.32 Http_MsgAddRefer

Adds the Refer header into a HTTP request message.

```
ZINT Http_MsgAddRefer(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_SSTR *pstUri);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstUri

The server URI.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.33 Http_MsgAddServer

Adds the Server header into a HTTP response message.

```
ZINT Http_MsgAddServer(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcProductVal);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcProductVal

Information about the Web server software.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.34 Http_MsgAddTrsfEncoding

Adds the Transfer-Encoding header into a HTTP message.

```
ZINT Http_MsgAddTrsfEncoding(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcCodings);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcCodings

The list of encodings.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.35 Http_MsgAddUserAgent

Adds the User-Agent header into a HTTP request message.

```
ZINT Http_MsgAddUserAgent(ST\_HTTP\_MSG *pstMsg, ZCHAR *pcProductVal);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

ZCHAR *pcProductVal

Information about the Web client.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.36 Http_MsgAddX3gppIntendId

Adds the X-3GPP-Intended-Identity header into a HTTP message.

```
ZINT Http_MsgAddX3gppIntendId(ST\_HTTP\_MSG *pstMsg,  
                               ST\_ZOS\_SSTR *pstIdent);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_SSTR](#) *pstIdent

The identity.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.37 Http_MsgAddBody

Adds a body string into a HTTP message.

```
ZINT Http_MsgAddBody(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_LSSTR *pstBodyStr);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_LSSTR](#) *pstBodyStr

The body string.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.38 Http_MsgAddBodyX

Adds a message body contained in a data buffer.

```
ZINT Http_MsgAddBodyX(ST\_HTTP\_MSG *pstMsg, ST\_ZOS\_DBUF *pstBodyBuf);
```

[Parameters]

Input parameters:

[ST_HTTP_MSG](#) *pstMsg

The HTTP message.

[ST_ZOS_DBUF](#) *pstBodyBuf

The data buffer which contains the message body.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.39 Http_ParmFillDRspUserName

Fills the username of a digest response.

```
ZINT Http_ParmFillDRspUserName(ST\_ZOS\_DBUF *pstMemBuf,  
                                ST\_HTTP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstUser);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

[ST_ZOS_SSTR](#) *pstUser

The username.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.40 Http_ParmFillDRspRealm

Fills the realm of a digest response.

```
ZINT Http_ParmFillDRspRealm(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_HTTP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstRealm);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

[ST_ZOS_SSTR](#) *pstRealm

The realm.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.41 Http_ParmFillDRspNonce

Fills the Nonce of a digest response.

```
ZINT Http_ParmFillDRspNonce(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_HTTP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstNonce);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

[ST_ZOS_SSTR](#) *pstNonce

The Nonce.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.42 Http_ParmFillDRspUri

Fills the digest URI of a digest response.

```
ZINT Http_ParmFillDRspUri(ST\_ZOS\_DBUF *pstMemBuf,  
                           ST\_HTTP\_DIGEST\_RSP *pstDigestRsp,  
                           ST\_HTTP\_REQ\_URI *pstDigestUri);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

[ST_HTTP_REQ_URI](#) *pstDigestUri

The digest URI.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.43 Http_ParmFillDRspRsp

Fills the response of a digest response.

```
ZINT Http_ParmFillDRspRsp(ST\_ZOS\_DBUF *pstMemBuf,  
                           ST\_HTTP\_DIGEST\_RSP *pstDigestRsp, ZCHAR *pcRspStr);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

ZCHAR *pcRspStr

The response string.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.44 Http_ParmFillDRspOpaque

Fills the Opaque of a digest response.

```
ZINT Http_ParmFillDRspOpaque(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_HTTP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstOpaque);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

[ST_ZOS_SSTR](#) *pstOpaque

The Opaque.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.45 Http_ParmFillDRspAlgo

Fills the algorithm of a digest response.

```
ZINT Http_ParmFillDRspAlgo(ST\_ZOS\_DBUF *pstMemBuf,  
                            ST\_HTTP\_DIGEST\_RSP *pstDigestRsp, ST\_HTTP\_ALGO *pstAlgo);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIGEST_RSP](#) *pstDigestRsp

The digest response.

[ST_HTTP_ALGO](#) *pstAlgo

The algorithm.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.46 Http_ParmDRspLstAdd

Creates a digest response and adds it into the digest-response list.

```
ZINT Http_ParmDRspLstAdd(ST\_ZOS\_DBUF *pstMemBuf,  
                        ST\_HTTP\_DIG\_RSP\_LST *pstRspLst, ZUCHAR ucType,  
                        ST\_HTTP\_DIG\_RSP **ppstRsp);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_DIG_RSP_LST](#) *pstRspLst

The response list.

ZUCHAR ucType

The digest response type.

Output parameters:

[ST_HTTP_DIG_RSP](#) **ppstRsp

The newly created digest response.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.47 Http_ParmDRspLstRmv

Removes a digest response from the digest-response list.

```
ZINT Http_ParmDRspLstRmv(ST\_HTTP\_DIG\_RSP\_LST *pstRspLst,  
                        ST\_HTTP\_DIG\_RSP *pstRsp);
```

[Parameters]

Input parameters:

[ST_HTTP_DIG_RSP_LST](#) *pstRspLst

The digest-response list.

[ST_HTTP_DIG_RSP](#) *pstRsp

The response to remove.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.48 Http_ParmFillCredents

Fills the credentials based on a specific challenge.

```
ZINT Http_ParmFillCredents(ST\_ZOS\_DBUF *pstMemBuf,  
                           ST\_HTTP\_CREDENTIALS *pstCredents, ZUCHAR ucMethod,  
                           ST\_HTTP\_CHALLENGE *pstChallenge, ST\_ZOS\_SSTR *pstUser,  
                           ST\_ZOS\_SSTR *pstPass, ST\_HTTP\_REQ\_URI *pstDigestUri);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The data buffer.

[ST_HTTP_CREDENTIALS](#) *pstCredents

The credentials.

ZUCHAR ucMethod

The method.

[ST_HTTP_CHALLENGE](#) *pstChallenge

The challenge.

[ST_ZOS_SSTR](#) *pstUser

The username.

[ST_ZOS_SSTR](#) *pstPass

The password.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.49 Http_ParmCalcA1

Calculates A1 on which the address the STUN must be prepared to receive Binding Requests.

```
ZINT Http_ParmCalcA1(ST\_ZOS\_SSTR *pstUsername, ST\_ZOS\_SSTR *pstPassword,
                    ST\_ZOS\_SSTR *pstRealm, ZCHAR acResultStr[HTTP_MD5_HEX_STR_LEN]);
```

[Parameters]

Input parameters:

[ST_ZOS_SSTR](#) *pstUsername

The username.

[ST_ZOS_SSTR](#) *pstPassword

The password.

[ST_ZOS_SSTR](#) *pstRealm

The address realm.

Output parameters:

ZCHAR acResultStr[HTTP_MD5_HEX_STR_LEN]

The calculation of A1.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.50 Http_ParmCalcA2

Calculates A2 on which the STUN server must be prepare to receive Binding Requests.

```
ZINT Http_ParmCalcA2(ZUCHAR ucMethod, ST\_HTTP\_REQ\_URI *pstDigestUri,
                    ZCHAR acResultStr[HTTP_MD5_HEX_STR_LEN]);
```

[Parameters]

Input parameters:

ZUCHAR ucMethod

The method.

[ST_HTTP_REQ_URI](#) *pstDigestUri

The digest URI.

Output parameters:

ZCHAR acResultStr[HTTP_MD5_HEX_STR_LEN]

The calculation of A2.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.51 Http_ParmCalcKd

Calculates the KD.

```
ZINT Http_ParmCalcKd(ZCHAR acA1Md5Str[HTTP_MD5_HEX_STR_LEN],
                    ZCHAR acA2Md5Str[HTTP_MD5_HEX_STR_LEN], ST\_ZOS\_SSTR *pstNonce,
                    ZCHAR acResultStr[HTTP_MD5_HEX_STR_LEN]);
```

[Parameters]

Input parameters:

```
ZCHAR acA1Md5Str[HTTP_MD5_HEX_STR_LEN]
```

The A1 in the form of MD5.

```
ZCHAR acA2Md5Str[HTTP_MD5_HEX_STR_LEN]
```

The A2 in the form of MD5.

```
ST\_ZOS\_SSTR *pstNonce
```

The Nonce.

Output parameters:

```
ZCHAR acResultStr[HTTP_MD5_HEX_STR_LEN]
```

The calculation of KD.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.2.52 Http_GetMethodDesc

Gets the descriptin of a specific method.

```
ZCHAR * Http_GetMethodDesc(ZUCHAR ucMethod);
```

[Parameters]

Input parameters:

```
ZUCHAR ucMethod
```

The method.

Output parameters:

None.

[Return value]

Returns the description.

3.3 Useful Interfaces

These interfaces are included in httpc_ui.h.

3.3.1 Httpc_Open

The HTTP client uses this interface to open a session.

```
ZINT Httpc_Open(ZULONG dwUserId, ST\_ZOS\_INET\_ADDR *pstLocalAddr,  
                PFN\_HTTPCPROCSTAT pfnProcStat, PFN\_HTTPCPROCMSG pfnProcMsg,  
                ZULONG *pdwSessId);
```

[Parameters]

Input parameters:

ZULONG dwUserId

The user ID.

[ST_ZOS_INET_ADDR](#) *pstLocalAddr

The local address.

[PFN_HTTPCPROCSTAT](#) pfnProcStat

Session status processing function.

[PFN_HTTPCPROCMSG](#) pfnProcMsg

Session message processing function.

Output parameters:

ZULONG *pdwSessId

The session ID.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.2 Httpc_OpenX

The HTTP client uses this interface to open a session.

```
ZINT Httpc_OpenX(ZULONG dwUserId, ST\_ZOS\_INET\_ADDR *pstLocalAddr,  
                 PFN\_HTTPCPROCSTAT pfnProcStat, PFN\_HTTPCPROCHDRS pfnProcHdrs,  
                 PFN\_HTTPCPROCBODY pfnProcBody, ZULONG *pdwSessId);
```

[Parameters]

Input parameters:

ZULONG dwUserId

The user ID.

[ST_ZOS_INET_ADDR](#) *pstLocalAddr

The local address.

[PFN_HTTPCPROCSTAT](#) pfnProcStat

Session status processing function.

[PFN_HTTPCPROCHDRS](#) pfnProcHdrs

Session message headers processing function.

[PFN_HTTPCPROCBODY](#) pfnProcBody

Session message body processing function.

Output parameters:

ZULONG *pdwSessId

The session ID.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.3 Httpc_Close

Closes a session.

```
ZINT Httpc_Close(ZULONG dwSessId);
```

[Parameters]

Input parameters:

ZULONG dwSessId

The session ID.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.4 Httpc_Conn

Connects to a remote server.

```
ZINT Httpc_Conn(ZULONG dwSessId, ST\_ZOS\_INET\_ADDR *pstRmtAddr);
```

[Parameters]

Input parameters:

ZULONG dwSessId

The session ID.

[ST_ZOS_INET_ADDR](#) *pstRmtAddr

The address of the remote server.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.5 Httpc_ConnX

Connects to a remote server by the hostname of the server.

```
ZINT Httpc_ConnX(ZULONG dwSessId, ST\_ZOS\_SSTR *pstHost, ZUSHORT wPort);
```

[Parameters]

Input parameters:

ZULONG dwSessId

The session ID.

[ST_ZOS_SSTR](#) *pstHost

The hostname of the remote server.

ZUSHORT wPort

The port of the remote server.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.6 Httpc_Disc

Disconnects a specific session with the remote server.

```
ZINT Httpc_Disc(ZULONG dwSessId);
```

[Parameters]

Input parameters:

ZULONG dwSessId
ID of the session.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.7 Httpc_Send

Sends a message through a specific session.

```
ZINT Httpc_Send(ZULONG dwSessId, ST\_HTTP\_MSG *pstMsg);
```

[Parameters]**Input parameters:**

ZULONG dwSessId
ID of the session.

[ST_HTTP_MSG](#) *pstMsg
The message to send.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.8 Httpc_SendData

Sends a message through a specific session.

```
ZINT Httpc_SendData(ZULONG dwSessId, ST\_ZOS\_DBUF *pstData);
```

[Parameters]**Input parameters:**

ZULONG dwSessId
ID of the session.

[ST_ZOS_DBUF](#) *pstData
The message data buffer to send.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.9 Httpc_SendDataX

Sends a message through a specific session.

```
ZINT Httpc_SendDataX(ZULONG dwSessId, ST\_ZOS\_LSSTR *pstData);
```

[Parameters]**Input parameters:**

ZULONG dwSessId

ID of the session.

[ST_ZOS_LSSTR](#) *pstData

The message string to send.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.10 Httpc_GetUserId

Gets a session user ID.

```
ZINT Httpc_GetUserId(ZULONG dwSessId, ZULONG *pdwUserId);
```

[Parameters]**Input parameters:**

ZULONG dwSessId

The session ID.

Output parameters:

ZULONG *pdwUserId

The user ID.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.11 Httpc_SetUserId

Sets the user ID of a session.

```
ZINT Httpc_SetUserId(ZULONG dwSessId, ZULONG dwUserId);
```

[Parameters]

Input parameters:

ZULONG dwSessId
ID of the session.

ZULONG dwUserId
The user ID.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.12 Httpc_GetRmtAddr

Gets the remote address of a session.

```
ZINT Httpc_GetRmtAddr(ZULONG dwSessId, ST\_ZOS\_INET\_ADDR *pstAddr);
```

[Parameters]

Input parameters:

ZULONG dwSessId
ID of the session.

Output parameters:

[ST_ZOS_INET_ADDR](#) *pstAddr
The remote address.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.13 Httpc_SetRmtAddr

Sets the remote address of a session.

```
ZINT Httpc_SetRmtAddr(ZULONG dwSessId, ST\_ZOS\_INET\_ADDR *pstAddr);
```

[Parameters]

Input parameters:

ZULONG dwSessId
ID of the session.

[ST_ZOS_INET_ADDR](#) *pstAddr
The remote address.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4 Config Interfaces

These interfaces are included in httpc_cfg.h.

3.4.1 Httpc_CfgGetTaskPriority

Gets the task priority. The default priority is ZTASK_PRIORITY_NORMAL.

```
ZINT Httpc_CfgGetTaskPriority(ZINT *piPriority);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZINT *piPriority
The task priority.

[Return value]

Returns ZOK.

3.4.2 Httpc_CfgGetTaskStackSize

Gets the task size. The default size is 16k.

```
ZINT Httpc_CfgGetTaskStackSize(ZULONG *pdwStackSize);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwStackSize
The task size.

[Return value]

Returns ZOK.

3.4.3 Httpc_CfgGetTaskQueueSize

Gets the task queue size. The default size is 50.

```
ZINT Httpc_CfgGetTaskQueueSize(ZULONG *pdwQueueSize);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwQueueSize

The task queue size.

[Return value]

Returns ZOK.

3.4.4 Httpc_CfgGetTaskTimerNum

Gets the number of task timers. The default number is 50.

```
ZINT Httpc_CfgGetTaskTimerNum(ZULONG *pdwTimerNum);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwTimerNum

The number of task timers.

[Return value]

Returns ZOK.

3.4.5 Httpc_CfgGetLogLevel

Gets the log level. The default level is ZLOG_LEVEL_ALL.

```
ZINT Httpc_CfgGetLogLevel(ZULONG *pdwLevel);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwLevel

The log level.

[Return value]

Returns ZOK.

3.4.6 Httpc_CfgGetSessNum

Gets the number of sessions. The default number is 20.

```
ZINT Httpc_CfgGetSessNum(ZULONG *pdwNum);
```

[Parameters]**Input parameters:**

None.

Output parameters:

ZULONG *pdwNum

The number of sessions.

[Return value]

Returns ZOK.

3.4.7 Httpc_CfgSetTaskPriority

Sets the task priority.

```
ZINT Httpc_CfgSetTaskPriority(ZINT iPriority);
```

[Parameters]**Input parameters:**

ZINT iPriority

The task priority.

Output parameters:

None.

[Return value]

Returns ZOK.

3.4.8 Httpc_CfgSetTaskStackSize

Sets the task stack size.

```
ZINT Httpc_CfgSetTaskStackSize(ZULONG dwStackSize);
```

[Parameters]

Input parameters:

ZULONG dwStackSize
The task stack size.

Output parameters:

None.

[Return value]

Returns ZOK.

3.4.9 Httpc_CfgSetTaskQueueSize

Sets the task queue size.

```
ZINT Httpc_CfgSetTaskQueueSize(ZULONG dwQueueSize);
```

[Parameters]

Input parameters:

ZULONG dwQueueSize
The task queue size.

Output parameters:

None.

[Return value]

Returns ZOK.

3.4.10 Httpc_CfgSetTaskTimerNum

Sets the number of task timers.

```
ZINT Httpc_CfgSetTaskTimerNum(ZULONG dwTimerNum);
```

[Parameters]

Input parameters:

ZULONG dwTimerNum
The number of task timers.

Output parameters:

None.

[Return value]

Returns ZOK.

3.4.11 Httpc_CfgSetLogLevel

Sets the log level.

```
ZINT Httpc_CfgSetLogLevel(ZULONG dwLevel);
```

[Parameters]

Input parameters:

ZULONG dwLevel

The log level.

Output parameters:

None.

[Return value]

Returns ZOK.

3.4.12 Httpc_CfgSetSessNum

Sets the number of sessions.

```
ZINT Httpc_CfgSetSessNum(ZULONG dwNum);
```

[Parameters]

Input parameters:

ZULONG dwNum

The number of sessions.

Output parameters:

None.

[Return value]

Returns ZOK.

3.5 Task Interfaces

These interfaces are included in httpc_task.h.

3.5.1 Httpc_Start

Starts the HTTP task.

```
ZINT Httpc_Start();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.5.2 Httpc_Stop

Stops the HTTP task.

```
ZVOID Httpc_Stop();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

None.

3.6 Version Interfaces

These interfaces are included in httpc_version.h.

3.6.1 Httpc_GetVersion

Returns the version of HTTP client.

```
ZCHAR * Httpc_GetVersion();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns the version of the HTTP client.