
Juphoon Protocol Framework

DNS Resolver

Published: Nov 2006

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

Juphoon DNS Resolver Function Definition

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87304379

Fax: +86-574-87304379

Text Part Number:

Copyright © 2006, Juphoon System Software Corporation.

All rights reserved.

Contents

1. INTRODUCTION.....	4
1.1 DNS	4
1.2 AUDIENCE	4
1.3 SCOPE.....	4
1.4 ABBREVIATIONS	4
2. SYSTEM ENVIRONMENT.....	4
2.1 BASIC DATA TYPES	5
3. ELEMENTS OF THE DNS.....	5
3.1 DOMAIN NAME SPACE AND RESOURCE RECORDS	6
3.2 NAME SERVERS.....	6
3.3 RESOLVERS	6
4. OVERVIEW OF OPERATION.....	6
5. USER INTERFACES	7
5.1 STRUCTURES	7
5.1.1 <i>ST_ZOS_SSTR</i>	7
5.1.2 <i>ST_ZOS_DBUF</i>	8
5.1.3 <i>ST_ZOS_INET_ADDR</i>	8
5.1.4 <i>ST_DNS_RSP</i>	8
5.1.5 <i>ST_DNS_RSP_RR</i>	9
5.1.6 <i>ST_DNS_RSP_A</i>	9
5.1.7 <i>ST_DNS_RSP_SRV</i>	10
5.1.8 <i>ST_DNS_RSP_NAPTR</i>	10
5.1.9 <i>PFN_DNSCALLBACK</i>	11
5.1.10 <i>PFN_DNSIPV4CALLBACK</i>	11
5.2 CONFIG INTERFACES.....	11
5.2.1 <i>Dns_CfgGetTaskPriority</i>	11
5.2.2 <i>Dns_CfgGetTaskStackSize</i>	11
5.2.3 <i>Dns_CfgGetTaskQueueSize</i>	12
5.2.4 <i>Dns_CfgGetLogLevel</i>	12
5.2.5 <i>Dns_CfgGetIsIpv6Tpt</i>	13
5.2.6 <i>Dns_CfgGetLocalIpv4</i>	13
5.2.7 <i>Dns_CfgGetLocalIpv6</i>	13
5.2.8 <i>Dns_CfgGetLocalAddr</i>	14
5.2.9 <i>Dns_CfgGetPriServIpv4</i>	14
5.2.10 <i>Dns_CfgGetPriServIpv6</i>	15
5.2.11 <i>Dns_CfgGet2ndServIpv4</i>	15
5.2.12 <i>Dns_CfgGet2ndServIpv6</i>	16
5.2.13 <i>Dns_CfgGetQryTimeLen</i>	16
5.2.14 <i>Dns_CfgGetQryNum</i>	17
5.2.15 <i>Dns_CfgSetTaskPriority</i>	17

5.2.16	<i>Dns_CfgSetTaskStackSize</i>	18
5.2.17	<i>Dns_CfgSetTaskQueueSize</i>	18
5.2.18	<i>Dns_CfgSetLogLevel</i>	18
5.2.19	<i>Dns_CfgSetLocalIpv4</i>	19
5.2.20	<i>Dns_CfgSetLocalIpv6</i>	19
5.2.21	<i>Dns_CfgSetLocalAddr</i>	20
5.2.22	<i>Dns_CfgSetPriServIpv4</i>	20
5.2.23	<i>Dns_CfgSetPriServIpv6</i>	21
5.2.24	<i>Dns_CfgSet2ndServIpv4</i>	21
5.2.25	<i>Dns_CfgSet2ndServIpv6</i>	22
5.2.26	<i>Dns_CfgSetQryTimeLen</i>	22
5.2.27	<i>Dns_CfgSetQryNum</i>	23
5.3	MODULE INTERFACES.....	23
5.3.1	<i>Dns_Start</i>	23
5.3.2	<i>Dns_Stop</i>	24
5.3.3	<i>Dns_Restart</i>	24
5.4	UPPER USER INTERFACES.....	24
5.4.1	<i>Dns_Lookup</i>	24
5.4.2	<i>Dns_LookupX</i>	28
5.4.3	<i>Dns_GetHostByName</i>	29
5.4.4	<i>Dns_GetHostByNameX</i>	30
5.5	UTILITY INTERFACES	30
5.5.1	<i>Dns_CpyRrGrp</i>	30
5.5.2	<i>Dns_CpyQRsp</i>	31
5.5.3	<i>Dns_CpyRsp</i>	32
5.5.4	<i>Dns_GetRrAIpv4</i>	33

List of Tables

TABLE 2-1: BASIC DATA TYPES	5
-----------------------------------	---

List of Figures

FIGURE 2-1 DNS CLIENT.....	5
FIGURE 4-1 COMMON CONFIGURATION	6

1. Introduction

1.1 DNS

The Domain Name System (DNS) is a mixture of functions and data types which are an official protocol and functions and data types which are still experimental.

From the user's point of view, the Domain Name System is accessed through a simple procedure or OS call to a local agent, called a resolver. Domain names are useful as arguments to a resolver which retrieves information associated with the domain name. Thus a user might ask for the host address or mail information associated with a particular domain name. To enable the user to request a particular type of information, an appropriate query type is passed to the resolver with the domain name.

1.2 Audience

The readers of this document are assumed to have a working knowledge of the Domain Name System (DNS).

1.3 Scope

This document provides the definitions of interfaces provided by Juphoon DNS resolver to users. So they can request a particular type of information with a call to DNS resolver. Details on the realization of the resolver are not within the scope of this document.

1.4 Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
DNS	Domain Name System
OS	Operating System
RR	Resource Record
ZOS	Zero Operating System

2. System Environment

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

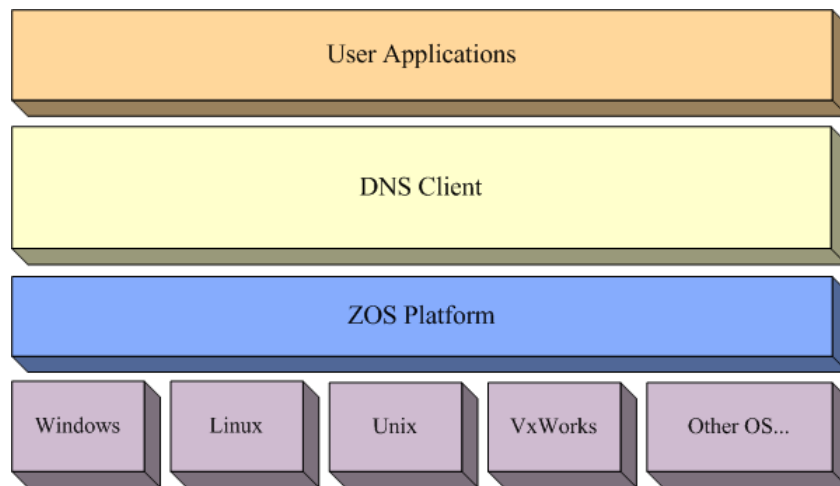


Figure 2-1 DNS Client

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. Elements of the DNS

The DNS has three major components.

3.1 Domain name space and resource records

They are specifications for a tree structured name space and data associated with the names. Conceptually, each node and leaf of the domain name space tree names a set of information, and query operations are attempts to extract specific types of information from a particular set. A query names the domain name of interest and describes the type of resource information that is desired. For example, the Internet uses some of its domain names to identify hosts; queries for address resources return Internet host addresses.

3.2 Name servers

Name servers are server programs which hold information about the domain tree's structure and set information. A name server may cache structure or set information about any part of the domain tree, but in general a particular name server has complete information about a subset of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree. Name servers know the parts of the domain tree for which they have complete information; a name server is said to be an authority for these parts of the name space. Authoritative information is organized into units called ZONES, and these zones can be automatically distributed to the name servers which provide redundant service for the data in a zone.

3.3 Resolvers

Resolvers are programs that extract information from name servers in response to client requests. Resolvers must be able to access at least one name server and use that name server's information to answer a query directly, or pursue the query using referrals to other name servers. A resolver will typically be a system routine that is directly accessible to user programs; hence no protocol is necessary between the resolver and the user program.

DNS Resolver provides four interfaces, including [Dns_Lookup](#), [Dns_LookupX](#), [Dns_GetHostByName](#) and [Dns_GetHostByNameX](#) for users to request a particular type of information through DNS resolver.

4. Overview of Operation

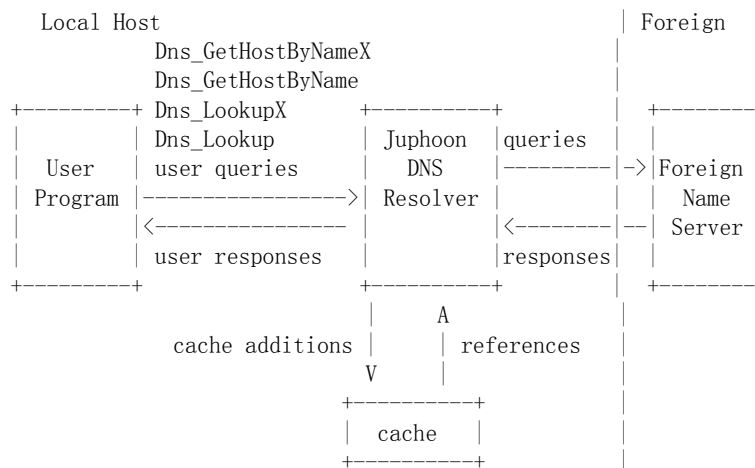


Figure 4-1 Common Configuration

A host can participate in the domain name system in a number of ways, depending on whether the host runs programs that retrieve information from the domain system, name servers that answer queries from other hosts, or various combinations of both functions. The simplest, and perhaps most typical, configuration is show in figure 4-1.

User programs interact with the domain name space through Juphoon DNS Resolver by calling [Dns_Lookup](#), [Dns_LookupX](#), [Dns_GetHostByName](#) or [Dns_GetHostByNameX](#). The different between -X functions and no X functions is that -X functions is for synchronous usage, and no X functions is for asynchronous usage. In synchronous mode, the user will wait for the query result which will transfer to user in return value and output parameters. In asynchronous mode, the user needn't wait for the query result returns, the query result will transfer to user by calling the callback function provided by user, and the query function's return value only indicates whether the query is accepted by the dns module.

DNS Resolver receives a request from the user program when the user program calls [Dns_Lookup](#), [Dns_LookupX](#), [Dns_GetHostByName](#) or [Dns_GetHostByNameX](#), then returns the desired information in a form compatible with the local host's data formats. To get the desired information, DNS Resolver first searches the cache for the desired data. If the data is in the cache, DNS Resolver returns the data. If there are no desired data in the cache, DNS Resolver looks for a name server to ask for the required data, or pursue the query using referrals to other name servers. And when it gets the required data, it returns it to the user program by sending a user response and caches the data.

There are two major timers. One is cache timer which is of type `ZTIMER_MODE_TASK_CYCLE` meaning it starts by itself every time after it fires. The cache has to update all data in it when the cache timer fires.

Another one is the resolver timer, which is of type `ZTIMER_MODE_NOCYCLE`. Unlike the cache timer, it does not start by itself. The resolver timer is started by DNS Resolver and when it fires it needs to be started by DNS Resolver again. In the case that if there are no desired data in the cache, DNS Resolver has to look for a name server to ask for the required data, or pursue the query using referrals to other name servers. In order to handle the case where a user query never generates a user response, the resolver timer must be set when the DNS Resolver looks for name servers to ask for the required data on receiving a user query. The resolver timer is started with a value of 500ms. When it fires, the resolver timer is set with a value of 2*500ms. And then DNS Resolver sends the query again. When the resolver timer fires 2*500ms later, the query is retransmitted again. This process continues so that the query is retransmitted with intervals that double and the resolver timer is reset with a value that double after each transmission unless the value exceeds 10 seconds or DNS Resolver receives a response. Note that if the time value exceeds 10 seconds, it is considered a time out. When this occurs, DNS Resolver searches desired information in the local cache again, and return `ZTRUE` if match record found, or return `ZFAILED`.

5. User Interfaces

5.1 Structures

5.1.1 ST_ZOS_SSTR

It is ZOS string structure which has a pointer to a string and a field to hold the string's length.

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

5.1.2 ST_ZOS_DBUF

It is ZOS data buffer structure.

```

/* zos data buffer block */
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;

```

5.1.3 ST_ZOS_INET_ADDR

```

/* zos internet address */
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;

```

5.1.4 ST_DNS_RSP

It is user response structure which contains the query ID (it indicates that to which user query the response pertains), the query result, the Resource Record (RR) array count and pointer *pstRrGrp* to the Resource Record (RR) response group.

```

/* dns query response */
typedef struct tagDNS_RSP
{
    ZUCHAR ucResult;           /* query result DNS_RCODE_OK... */
    ZUCHAR ucRrCount;         /* RR array count */
    ZUSHORT wRrType;          /* query type EN_DNS_RR_TYPE */
    ST\_DNS\_RSP\_RR *pstRrGrp;  /* RR response group */
} ST_DNS_RSP;

```

5.1.5 ST_DNS_RSP_RR

This is the structure of pointer *pstRrGrp* (See [ST_DNS_RSP](#)). *wType* shows the type of a resource record (the RR type code). If the value of *wType* is *EN_DNS_RR_A* which means the RR type code is A, then *pstRrGrp.u.stA* is holding an IP address. If the value is *EN_DNS_RR_SRV* which means the RR type code is SRV, then *pstRrGrp.u.stSrv* is holding the information of a target host. If the value is *EN_DNS_RR_NAPTR* which means the RR type code is NAPTR, then *pstRrGrp.u.stNaptr* is holding the information of a domain name.

```

/* dns query response of RR */
typedef struct tagDNS_RSP_RR
{
    ZUSHORT wType;            /* query response type EN_DNS_RR_TYPE */
    ZUCHAR aucSpare[2];      /* for 32 bit alignment */
    union
    {
        ST\_DNS\_RSP\_A stA;      /* A response */
        ST\_DNS\_RSP\_SRV stSrv;  /* SRV response */
        ST\_DNS\_RSP\_NAPTR stNaptr; /* NAPTR response */
        ST\_ZOS\_SSTR stOther;    /* other resource data */
    } u;
} ST_DNS_RSP_RR;

```

5.1.6 ST_DNS_RSP_A

This is the structure of the RR whose type code is A. A means host address.

The value of *dwAddr* is a host address.

```

/* dns query response of A */
typedef struct tagDNS_RSP_A
{
    ZULONG dwAddr;           /* address */
} ST_DNS_RSP_A;

```

5.1.7 ST_DNS_RSP_SRV

This is structure of the SRV RR. According to rfc 2782, the SRV RR allows administrators to use several servers for a single domain, to move services from host to host with little fuss, and to designate some hosts as primary servers for a service and others as backups.

wPriority represents the priority of the target host; *wWeight* represents Weight, a server selection mechanism, which specifies a relative weight for entries with the same priority; the value of *wPort* is the service port of the host; and *stTarget* holds the domain name of the target host.

```
/* dns query response of SRV */
typedef struct tagDNS_RSP_SRV
{
    ZUSHORT wPriority;           /* target host priority */
    ZUSHORT wWeight;           /* weight for entries */
    ZUSHORT wPort;             /* target host service port */
    ZUCHAR aucSpare[2];        /* for 32 bit alignment */
    ST_ZOS_SSTR stTarget;      /* target domain name */
} ST_DNS_RSP_SRV;
```

5.1.8 ST_DNS_RSP_NAPTR

This is the structure of the NAPTR RR. According to rfc 3958, NAPTR records are used to store application service+protocol information for a given domain.

wOrder represents the Order, which is a 16-bit unsigned integer specifying the order in which the NAPTR records must be processed to ensure the correct ordering of rules. *wPreference* represents Preference which is 16-bit unsigned integer that specifies the order in which NAPTR records with equal “order” values should be processed. *stFlags* represents Flags, a character-string containing flags to control aspects of the rewriting and interpretation of the fields in the record. *stService* is equal to Service which specifies the service(s) available down this rewrite path. *stRegexp* represents Regexp, a string containing a substitution expression that is applied to the original string held by the client in order to construct the next domain name to lookup. *stReplace* represents Replacement, the next name to query for NAPTR, SRV, or address records depending on the value of the flags field.

```
/* dns query response of NAPTR */
typedef struct tagDNS_RSP_NAPTR
{
    ZUSHORT wOrder;           /* rules order */
    ZUSHORT wPreference;      /* equal "order" rules */
    ZULONG dwFlags;           /* flags bit flag DNS_NAPTR_FLAG_S ... */
    ST_ZOS_SSTR stFlags;      /* flags */
    ST_ZOS_SSTR stService;     /* service */
    ST_ZOS_SSTR stRegexp;     /* regular expression */
    ST_ZOS_SSTR stReplace;    /* replacement domain name */
} ST_DNS_RSP_NAPTR;
```

5.1.9 PFN_DNSCALLBACK

The prototype of the callback function used by DNS Resolver when it gets the desired information or the attempt fails.

```
typedef ZVOID (*PFN_DNSCALLBACK) (ZULONG dwUserId, ST\_DNS\_RSP *pstRsp);
```

5.1.10 PFN_DNSIPV4CALLBACK

The prototype of the callback function used by Get_HostByNameX to indicates resolved IPv4 address.

```
typedef ZVOID (*PFN_DNSIPV4CALLBACK) (ZULONG dwUserId, ZULONG dwIpv4);
```

5.2 Config Interfaces

Here is a function used to configure DNS Resolver before it is operating. These interfaces are included in dns_cfg.h.

5.2.1 Dns_CfgGetTaskPriority

Gets the task priority.

```
ZINT Dns_CfgGetTaskPriority(ZINT *piPriority);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZINT *piPriority

The task priority.

[Return value]

Returns ZOK.

5.2.2 Dns_CfgGetTaskStackSize

Gets the task stack size.

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

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwStackSize

The task stack size.

[Return value]

Returns ZOK.

5.2.3 Dns_CfgGetTaskQueueSize

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

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

[Parameters]**Input parameters:**

None.

Output parameters:

ZULONG *pdwQueueSize

The task queue size.

[Return value]

Returns ZOK.

5.2.4 Dns_CfgGetLogLevel

Gets the log level. The default level is ZLOG_LEVEL_ALL.

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

[Parameters]**Input parameters:**

None.

Output parameters:

ZULONG *pdwLevel

The log level.

[Return value]

Returns ZOK.

5.2.5 Dns_CfgGetIsIpv6Tpt

Gets a Boolean value which indicates whether it is IPv6 transport.

```
ZINT Dns_CfgGetIsIpv6Tpt(ZBOOL *pbIsIpv6Tpt);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbIsIpv6Tpt
```

The Boolean value.

[Return value]

Returns ZOK.

5.2.6 Dns_CfgGetLocalIpv4

Gets the local IPv4 address and port.

```
ZINT Dns_CfgGetLocalIpv4(ZULONG *pdwIpv4, ZUSHORT *pwPort);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwIpv4
```

The IPv4 address.

```
ZUSHORT *pwPort
```

The port.

[Return value]

Returns ZOK.

5.2.7 Dns_CfgGetLocalIpv6

Gets the local IPv6 address and port.

```
ZINT Dns_CfgGetLocalIpv6(ZUCHAR **ppucIpv6, ZUSHORT *pwPort);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZUCHAR **ppucIpv6
The IPv6 address.

ZUSHORT *pwPort
The port.

[Return value]

Returns ZOK.

5.2.8 Dns_CfgGetLocalAddr

Gets the local address.

```
ZINT Dns_CfgGetLocalAddr(ST\_ZOS\_INET\_ADDR **ppstAddr);
```

[Parameters]

Input parameters:

None.

Output parameters:

[ST_ZOS_INET_ADDR](#) **ppstAddr
The local address.

[Return value]

Returns ZOK.

5.2.9 Dns_CfgGetPriServIpv4

Gets the primary server IPv4 address and port.

```
ZINT Dns_CfgGetPriServIpv4(ZULONG *pdwIpv4, ZUSHORT *pwPort);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwIpv4

The IPv4 address.

ZUSHORT *pwPort

The port.

[Return value]

Returns ZOK.

5.2.10 Dns_CfgGetPriServIpv6

Gets the primary server IPv6 address and port.

```
ZINT Dns_CfgGetPriServIpv6(ZUCHAR **ppucIpv6, ZUSHORT *pwPort);
```

[Parameters]**Input parameters:**

None.

Output parameters:

ZUCHAR **ppucIpv6

The IPv6 address.

ZUSHORT *pwPort

The port.

[Return value]

Returns ZOK.

5.2.11 Dns_CfgGet2ndServIpv4

Gets the secondary server IPv4 address and port.

```
ZINT Dns_CfgGet2ndServIpv4(ZULONG *pdwIpv4, ZUSHORT *pwPort);
```

[Parameters]**Input parameters:**

None.

Output parameters:

ZULONG *pdwIpv4
The IPv4 address.

ZUSHORT *pwPort
The port.

[Return value]

Returns ZOK.

5.2.12 Dns_CfgGet2ndServIpv6

Gets the secondary server IPv6 address and port.

```
ZINT Dns_CfgGet2ndServIpv6(ZUCHAR **ppucIpv6, ZUSHORT *pwPort);
```

[Parameters]**Input parameters:**

None.

Output parameters:

ZUCHAR **ppucIpv6
The IPv6 address.

ZUSHORT *pwPort
The port.

[Return value]

Returns ZOK.

5.2.13 Dns_CfgGetQryTimeLen

Gets the time length of waiting the response of DNS query.

```
ZINT Dns_CfgGetQryTimeLen(ZULONG *pdwTimeLen);
```

[Parameters]**Input parameters:**

None.

Output parameters:

ZULONG *pdwTimeLen

The time length of waiting the response of DNS query.

[Return value]

Returns ZOK.

5.2.14 Dns_CfgGetQryNum

Gets max number of DNS query at the same time.

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

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwNum

The max number of DNS query at the same time.

[Return value]

Returns ZOK.

5.2.15 Dns_CfgSetTaskPriority

Sets the task priority.

```
ZINT Dns_CfgSetTaskPriority(ZINT iPriority);
```

[Parameters]

Input parameters:

ZINT iPriority

The task priority.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.16 Dns_CfgSetTaskStackSize

Sets the task stack size.

```
ZINT Dns_CfgSetTaskStackSize(ZULONG dwStackSize);
```

[Parameters]

Input parameters:

ZULONG dwStackSize
The stack size.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.17 Dns_CfgSetTaskQueueSize

Sets the task queue size.

```
ZINT Dns_CfgSetTaskQueueSize(ZULONG dwQueueSize);
```

[Parameters]

Input parameters:

ZULONG dwQueueSize
The queue size.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.18 Dns_CfgSetLogLevel

Sets the log level.

```
ZINT Dns_CfgSetLogLevel(ZULONG dwLevel);
```

[Parameters]

Input parameters:

ZULONG dwLevel

The log level.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.19 Dns_CfgSetLocalIpv4

Sets the local IPv4 address and port.

```
ZINT Dns_CfgSetLocalIpv4(ZULONG dwIpv4, ZUSHORT wPort);
```

[Parameters]**Input parameters:**

ZULONG dwIpv4

The IPv4 address.

ZUSHORT wPort

The port.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.20 Dns_CfgSetLocalIpv6

Sets the local IPv6 address and port.

```
ZINT Dns_CfgSetLocalIpv6(ZUCHAR *pucIpv6, ZUSHORT wPort);
```

[Parameters]**Input parameters:**

ZUCHAR *pucIpv6

The IPv6 address.

ZUSHORT wPort

The port.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.21 Dns_CfgSetLocalAddr

Sets the local address.

```
ZINT Dns_CfgSetLocalAddr(ST\_ZOS\_INET\_ADDR *pstAddr);
```

[Parameters]

Input parameters:

[ST_ZOS_INET_ADDR](#) *pstAddr

The local address.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.22 Dns_CfgSetPriServIpv4

Sets the primary server IPv4 address and port.

```
ZINT Dns_CfgSetPriServIpv4(ZULONG dwIpv4, ZUSHORT wPort);
```

[Parameters]

Input parameters:

ZULONG dwIpv4
The IPv4 address.

ZUSHORT wPort
The port.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.23 Dns_CfgSetPriServIpv6

Sets the primary server IPv6 address and port.

```
ZINT Dns_CfgSetPriServIpv6(ZUCHAR *pucIpv6, ZUSHORT wPort);
```

[Parameters]

Input parameters:

ZUCHAR *pucIpv6
The IPv6 address.

ZUSHORT wPort
The port.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.24 Dns_CfgSet2ndServIpv4

Sets the secondary server IPv4 address and port.

```
ZINT Dns_CfgSet2ndServIpv4(ZULONG dwIpv4, ZUSHORT wPort);
```

[Parameters]

Input parameters:

ZULONG dwIpv4

The IPv4 address.

ZUSHORT wPort

The port.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.25 Dns_CfgSet2ndServIpv6

Sets the secondary server IPv6 address and port.

```
ZINT Dns_CfgSet2ndServIpv6(ZUCHAR *pucIpv6, ZUSHORT wPort);
```

[Parameters]**Input parameters:**

ZUCHAR *pucIpv6

The IPv6 address.

ZUSHORT wPort

The port.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.26 Dns_CfgSetQryTimeLen

Sets the time length of waiting the response of DNS query.

```
ZINT Dns_CfgSetQryTimeLen(ZULONG dwTimeLen);
```

[Parameters]**Input parameters:**

```
ZULONG dwTimeLen
```

The the time length of waiting the response of DNS query.

Output parameters:

None.

[Return value]

Returns ZOK.

5.2.27 Dns_CfgSetQryNum

Sets max number of DNS query at the same time.

```
ZINT Dns_CfgSetQryNum(ZULONG dwNum);
```

[Parameters]**Input parameters:**

```
ZULONG dwNum
```

The max number of DNS query at the same time.

Output parameters:

None.

[Return value]

Returns ZOK.

5.3 Module Interfaces

5.3.1 Dns_Start

This is a task interface used to start the DNS stack.

```
ZINT Dns_Start();
```

[Parameters]**Input parameters:**

None.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

5.3.2 Dns_Stop

This is a task interface used to stop the DNS stack.

```
ZVOID Dns_Stop();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

None.

5.3.3 Dns_Restart

Restarts the transport layer.

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

5.4 Upper user interfaces

Next details will be given on four user interfaces provided by DNS Resolver to request a particular type of information.

5.4.1 Dns_Lookup

A user program is able to request a particular type of information through DNS Resolver by calling *Dns_Lookup*. The different from [Dns_LookupX](#) is that the user program does not need to wait before DNS Resolver gets the desired information or the attempt fails, so it is an asynchronous function. The user program should provide a callback function as an input parameter for DNS Resolver to invoke when it gets the desired data or the attempt fails.

```
ZINT Dns_Lookup(ZULONG dwUserId, ZUSHORT wRrType, ST\_ZOS\_SSTR *pstPntDN,  
               PFN\_DNSCALLBACK pfnCallback)
```

[Parameters]

Input parameters:

ZULONG dwUserId

The user ID represents the user program who wants to request a particular type of information through DNS Resolver.

ZUSHORT wRrType

The query type -- the type of information the user program wants to request. The allowable values of the query type are:

```

EN_DNS_RR_A = 1           /* ipv4 address */
EN_DNS_RR_NS = 2         /* authoritative name server */
EN_DNS_RR_MD = 3         /* mail destination (Obsolete - use MX) */
EN_DNS_RR_MF = 4         /* mail forwarder (Obsolete - use MX) */
EN_DNS_RR_CNAME = 5      /* canonical name */
EN_DNS_RR_SOA = 6        /* start of authority zone */
EN_DNS_RR_MB = 7         /* mailbox domain name (EXPERIMENTAL) */
EN_DNS_RR_MG = 8         /* mail group member (EXPERIMENTAL) */
EN_DNS_RR_MR = 9         /* mail rename name (EXPERI.) */
EN_DNS_RR_NULL = 10      /* Null resource record (EXPERIMENTAL) */
EN_DNS_RR_WKS = 11       /* well known service */
EN_DNS_RR_PTR = 12       /* domain name pointer */
EN_DNS_RR_HINFO = 13     /* host information */
EN_DNS_RR_MINFO = 14     /* mailbox information */
EN_DNS_RR_MX = 15        /* mail exchange */
EN_DNS_RR_TXT = 16       /* text strings */
EN_DNS_RR_RP = 17        /* responsible person. */
EN_DNS_RR_AFSDB = 18     /* AFS cell database. */
EN_DNS_RR_X25 = 19       /* X_25 calling address. */
EN_DNS_RR_ISDN = 20      /* ISDN calling address. */
EN_DNS_RR_RT = 21        /* router. */
EN_DNS_RR_NSAP = 22      /* NSAP address. */
EN_DNS_RR_NSAP_PTR = 23  /* reverse NSAP lookup (deprecated). */
EN_DNS_RR_SIG = 24       /* security signature. */
EN_DNS_RR_KEY = 25       /* security key. */
EN_DNS_RR_PX = 26        /* X.400 mail mapping. */
EN_DNS_RR_GPOS = 27      /* geographical position (withdrawn). */
EN_DNS_RR_AAAA = 28      /* ipv6 Address. */
EN_DNS_RR_LOC = 29       /* location Information. */
EN_DNS_RR_NXT = 30       /* next domain (security). */
EN_DNS_RR_EID = 31       /* endpoint identifier. */
EN_DNS_RR_NIMLOC = 32    /* nimrod Locator. */
EN_DNS_RR_SRV = 33       /* server Selection. */
EN_DNS_RR_ATMA = 34      /* ATM Address */
EN_DNS_RR_NAPTR = 35     /* naming authority pointer */
EN_DNS_RR_OPT = 41       /* OPT pseudo-RR, RFC2761 */

```

```

/* query type values which do not appear in resource records. */
EN_DNS_RR_IXFR = 251          /* incremental zone transfer. */
EN_DNS_RR_AXFR = 252        /* transfer zone of authority. */
EN_DNS_RR_MAILB = 253      /* transfer mailbox records. */
EN_DNS_RR_MAILA = 254      /* transfer mail agent records. */
EN_DNS_RR_ANY = 255        /* wildcard match. */
EN_DNS_RR_MAX = 65536

```

[ST_ZOS_SSTR](#) *pstPntDN

The name of the node to which the query pertains, and the length of the name.

[PFN_STUNCALLBACK](#) pfnCallback

The callback function defined by the user program for DNS Resolver to invoke when it gets the desired data or the attempt fails.

Output parameters:

None.

[Return value]

ZOK: DNS Resolver has succeeded in sending a query message.

ZFAILED: Operation failed.

[Example]

```

ST\_ZOS\_SSTR stHostname;
ZUSHORT wdQryId;
ZINT iRet;
PFN\_DNSCALLBACK pfnCallback;

ZINT Dns_Callback(ZULONG dwUserId,
                 ST\_DNS\_RSP *pstQRsp)
{
    .....
    Return ZOK;
}
.....

stHostname.pcStr = "www.yahoo.com";
stHostname.wLen = 13;
pfnCallback = Dns_Callback;
iRet = Dns_Lookup(wdQryId, EN_DNS_RR_A, &stHostname, pfnCallback);
.....

```

5.4.2 Dns_LookupX

Like Dns_Lookup, this function is also used by the user program to request a particular type of information through DNS Resolver. The user program needs to specify the type as an input parameter. *Dns_LookupX* is a synchronous function, means that the user program has to wait until DNS Resolver gets the desired information or the attempt fails.

```
ZINT Dns_LookupX(ZULONG dwUserId, ST\_ZOS\_DBUF *pstMemBuf, ZUSHORT wRrType,
                ST\_ZOS\_SSTR *pstPntDN, ST\_DNS\_RSP *pstRsp)
```

[Parameters]

Input parameters:

ZUSHORT dwUserId

The user ID represents the user program who wants to request a particular type of information through DNS Resolver.

Currently, The DNS Resolver only supports decode RR types including ST_DNS_RSP_A, ST_DNS_RSP_SRV and ST_DNS_RSP_NAPTR. Resource Records of other types can not be decoded and put into a buffer which is pointed by ST_ZOS_SSTR.

ZUSHORT wRrType

The query type -- the type of information the user program wants to request. See [Dns_Lookup](#) for the allowable values of query type.

[ST_ZOS_SSTR](#) *pstPntDN

The name of the node to which the query pertains, and the length of the name.

[ST_DNS_RSP](#) *pstQRsp

A pointer which will point to the user response to the user query after DNS Resolver gets the desired information.

Output parameters:

[ST_DNS_RSP](#) *pstQRsp

A pointer which points to the user response to the user query after DNS Resolver gets the desired information.

[Return value]

ZOK: DNS Resolver has succeeded in getting desired information.

ZFAILED: Operation failed.

[Example]

```

ST\_ZOS\_SSTR stHostname;
ST\_DNS\_RSP stQRsp;
ZUSHORT wDQryId;
ZINT iRet;

.....

stHostname.pcStr = "www.yahoo.com";
stHostname.wLen = 13;
iRet = Dns_LookupX(wDQryId, EN_DNS_RR_A, &stHostname, &stQRsp);
.....

```

5.4.3 Dns_GetHostByName

Gets a host address by the host name in an asynchronous way.

```

ZINT Dns_GetHostByName(ZULONG dwUserId, ST\_ZOS\_SSTR *pstHostName,
PFN\_DNSIPV4CALLBACK pfnIpv4Callback)

```

[Parameters]

Input parameters:

ZULONG dwUserId

The user ID represents the user program who wants to request a particular type of information through DNS Resolver.

[ST_ZOS_SSTR](#) *pstHostName

The host name by which DNS Resolver will get the host address.

[PFN_DNSIPV4CALLBACK](#) pfnIpv4Callback

The callback function defined by the user program for DNS Resolver to invoke when it gets the desired data or the attempt fails.

Output parameters:

None.

[Return value]

ZOK: DNS Resolver has succeeded in getting the host address.

ZFAILED: Operation failed.

[Example]

5.4.4 Dns_GetHostByNameX

Gets a host address by the host name in a synchronous way

```
ZINT Dns_GetHostByNameX(ST\_ZOS\_SSTR *pstHostName, ZULONG *pdwIpAddr)
```

[Parameters]

Input parameters:

```
ST\_ZOS\_SSTR *pstHostName
```

The host name by which DNS Resolver will get the host address.

```
ZULONG *pdwIpAddr
```

A pointer which will points to the host address after DNS Resolver gets it.

Output parameters:

```
ZULONG *pdwIpAddr
```

A pointer to the host address attained by DNS Resolver.

[Return value]

ZOK: DNS Resolver has succeeded in getting the host address.

ZFAILED: Operation failed.

[Example]

```
ST\_ZOS\_SSTR stHostname;
ZULONG dwIpAddr;
ZINT iRet;

stHostname.pcStr = "www.yahoo.com";
stHostname.wLen = 13;
iRet = Dns_GetHostByName(&stHostname, &dwIpAddr);
```

5.5 Utility interfaces

5.5.1 Dns_CpyRrGrp

Copies a group of RR (resource records) responses.

```
ZINT Dns_CpyRrGrp(ST\_ZOS\_DBUF *pstMemBuf, ST\_DNS\_RSP\_RR **ppstDstRr,
ST\_DNS\_RSP\_RR *pastSrcRr, ZUCHAR ucRrCount);
```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The memory buffer.

[ST_DNS_RSP_RR](#) *pastSrcRr

The RR response group to copy.

ZUCHAR ucRrCount

Number of the RR responses to copy.

Output parameters:

[ST_DNS_RSP_RR](#) **ppstDstRr

Points to the RR responses copied on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ST\_ZOS\_DBUF *pstMemBuf;
ST\_DNS\_RSP\_RR *pastSrcRr;
ST\_DNS\_RSP\_RR *pstDstRr;
ZUCHAR ucRrCount;
ZINT iRet;
.....
/* Copy a group of RR responses. */
iRet = Dns_CpyRrGrp(pstMemBuf, &pstDstRr, pastSrcRr, ucRrCount);

```

5.5.2 Dns_CpyQRsp

Copies a query response.

```

ZINT Dns_CpyQRsp(ST\_ZOS\_DBUF *pstMemBuf, ST\_DNS\_RSP\_RR *pstDstRr,
                 ST\_DNS\_RSP\_RR *pstSrcRr);

```

[Parameters]

Input parameters:

[ST_ZOS_DBUF](#) *pstMemBuf

The memory buffer.

[ST_DNS_RSP_RR](#) *pstSrcRr

The response to copy.

Output parameters:

```
ST_DNS_RSP_RR *pstDstRr
```

The response copied.

[Return value]

Returns ZOK.

[Example]

```
ST_ZOS_DBUF *pstMemBuf;
ST_DNS_RSP_RR *pastSrcRr;
ST_DNS_RSP_RR *pstDstRr;

.....

/* Copy a query response. */
Dns_CpyQRsp(pstMemBuf, pstDstRr, pastSrcRr);
```

5.5.3 Dns_CpyRsp

Copies a RR (resource records) response group.

```
ZINT Dns_CpyRsp(ST_ZOS_DBUF *pstMemBuf, ST_DNS_RSP *pstDstRsp,
                ST_DNS_RSP *pstSrcRsp);
```

[Parameters]

Input parameters:

```
ST_ZOS_DBUF *pstMemBuf
```

The memory buffer.

```
ST_DNS_RSP *pstSrcRsp
```

The response group to copy.

Output parameters:

```
ST_DNS_RSP *pstDstRsp
```

The response group copied.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ST_ZOS_DBUF *pstMemBuf;
ST_DNS_RSP *pstSrcRsp;
ST_DNS_RSP *pstDstRsp;
ZINT iRet;
.....
/* Copy a response group. */
iRet = Dns_CpyRsp(pstMemBuf, pstDstRsp, pstSrcRsp);

```

5.5.4 Dns_GetRrAIPv4

Retrieves a RR IPv4 address from a response.

```
ZINT Dns_GetRrAIPv4(ST_DNS_RSP *pstRsp, ZULONG *pdwIpv4);
```

[Parameters]

Input parameters:

```
ST_DNS_RSP *pstRsp
```

The response.

Output parameters:

```
ZULONG *pdwIpv4
```

The RR IPv4 address.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ST_DNS_RSP *pstRsp;
ZULONG dwIpv4;
ZINT iRet;
.....
/* Retrieve a RR IPv4 address from a response. */
iRet = Dns_GetRrAIPv4(pstRsp, &dwIpv4);

```