
Juphoon Protocol Framework

JPF

Published: Dec 2006

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

Juphoon Phone Framework Function Definition

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87287820

Fax: +86-574-87304379

Text Part Number:

Copyright © 2007, Juphoon System Software Corporation.

All rights reserved.

Contents

1. INTRODUCTION.....	4
1.1 JPF	4
1.2 AUDIENCE	4
1.3 SCOPE.....	4
1.4 ABBREVIATIONS	4
2. SYSTEM ENVIRONMENT.....	4
2.1 BASIC DATA TYPES	5
3. BASIC CONCEPTS.....	6
3.1 ENDPOINT.....	6
3.2 CONNECTION	6
4. USER INTERFACES	7
4.1 STRUCTURES	7
4.1.1 <i>ST_ZOS_SSTR</i>	7
4.1.2 <i>ST_JPF_CFG</i>	8
4.1.3 <i>ST_SIP_HOST</i>	8
4.1.4 <i>ST_SIP_HOST_PORT</i>	9
4.1.5 <i>ST_SUA_ENDP_URI</i>	9
4.1.6 <i>PFN_JPFEVNTPROC</i>	9
4.1.7 <i>ST_JPF_DB_VCODEC</i>	10
4.1.8 <i>ST_ZOS_SYS_TIME</i>	11
4.2 CONFIG INTERFACES.....	11
4.2.1 <i>Jpf_CfgGetTaskPriority</i>	11
4.2.2 <i>Jpf_CfgGetTaskStackSize</i>	11
4.2.3 <i>Jpf_CfgGetTaskQueueSize</i>	12
4.2.4 <i>Jpf_CfgGetLogLevel</i>	12
4.2.5 <i>Jpf_CfgGetEndpNum</i>	13
4.2.6 <i>Jpf_CfgGetConnNum</i>	13
4.2.7 <i>Jpf_CfgGetSubsNum</i>	14
4.2.8 <i>Jpf_CfgSetTaskPriority</i>	14
4.2.9 <i>Jpf_CfgSetTaskStackSize</i>	14
4.2.10 <i>Jpf_CfgSetTaskQueueSize</i>	15
4.2.11 <i>Jpf_CfgSetLogLevel</i>	15
4.2.12 <i>Jpf_CfgSetEndpNum</i>	16
4.2.13 <i>Jpf_CfgSetConnNum</i>	16
4.2.14 <i>Jpf_CfgSetSubsNum</i>	17
4.3 TASK INTERFACES	17
4.3.1 <i>Jpf_Start</i>	17
4.3.2 <i>Jpf_Stop</i>	19
4.4 EVENT INTERFACES	19
4.4.1 <i>Jpf_EvntRegCallback</i>	19

4.5	UPPER USER INTERFACES	20
4.5.1	<i>Endpoint Interfaces</i>	20
4.5.1.1	Call Control	20
4.5.1.2	Utilities	24
4.5.2	<i>Connection Interfaces</i>	30
4.5.2.1	Call Control	31
4.5.2.2	Utilities	50
4.5.3	<i>Userful Interfaces for Data Management</i>	64
4.5.3.1	Main Data Interfaces.....	64
4.5.3.2	User Data Interfaces	69
4.5.3.3	SIP Data Interfaces	89
4.5.3.4	Media Data Interfaces.....	118
4.5.3.5	RTP Data Interfaces.....	136
4.5.3.6	STUN Data Interfaces.....	139
4.5.3.7	DNS Data Interfaces	142
4.5.4	<i>Interfaces for Data Management</i>	150
4.5.4.1	Jpf_DbAppMain	150
4.5.4.2	Jpf_DbAppUser	150
4.5.4.3	Jpf_DbAppSip	151
4.5.4.4	Jpf_DbAppMedia	151
4.5.4.5	Jpf_DbAppRtp.....	152
4.5.4.6	Jpf_DbAppStun	152
4.5.4.7	Jpf_DbAppDns	152

List of Tables

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

List of Figures

FIGURE 2-1 JPF FRAMEWORK	5
FIGURE 3-1 BASIC CONCEPTS OF SIP PHONE MULTI-SERVICE FRAMEWORK	6

1. Introduction

1.1 JPF

Juphoon Phone Framework (JPF) is a SIP Phone framework implemented on a basis of multi-service framework. It includes elements as Endpoint/Connection API, Status Report and Application Process. Endpoint/Connection carries out operations on the order of users on the inner objects and provides interfaces to users for obtaining the states of the inner objects. Status Report informs the users the changes of states of the inner objects. Application Process makes the multi-service processes between JPF, users and SUA go along correctly.

1.2 Audience

The readers of this document are assumed to have a working knowledge of SIP Phone.

1.3 Scope

This document provides the definitions of interfaces provided by JPF. Details on the realization of the JPF are not within the scope of this document.

1.4 Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
JPF	Juphoon Phone Framework
URI	Uniform Resource Identifier
DNS	Domain Name System
DTMF	DualTone Multi Frequency
SIP	Session Initiation Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
STUN	Simple Traversal of UDP through NATs
RTCP	Real Time Control Protocol
RTP	Real Time Transport Protocol
AEC	Acoustic Echo Cancellation
AGC	Automatic Gain Control
ASD	Audio Silence Detection

2. System Environment

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

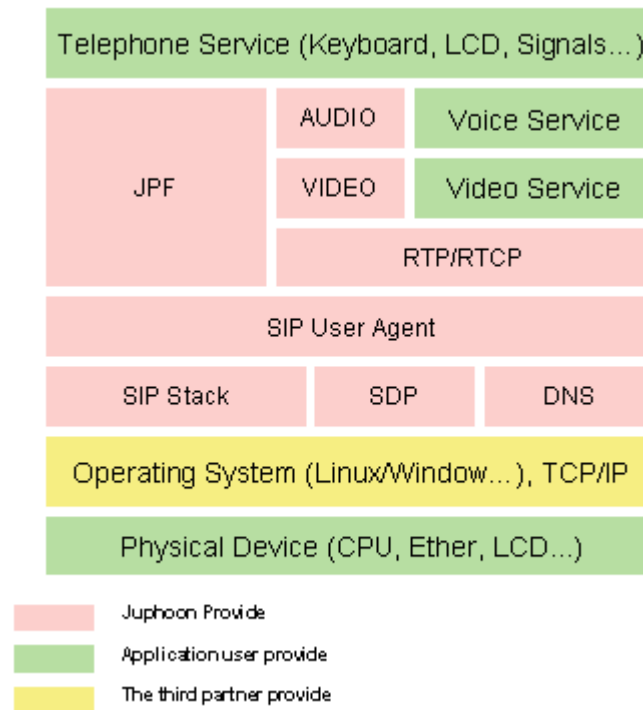


Figure 2-1 JPF Framework

2.1 Basic Data Types

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

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	boolean
ZVOID	void

Table 2-1: Basic Data Types

3. Basic Concepts

Two basic concepts about JPF are shown in Figure 3-1 below.

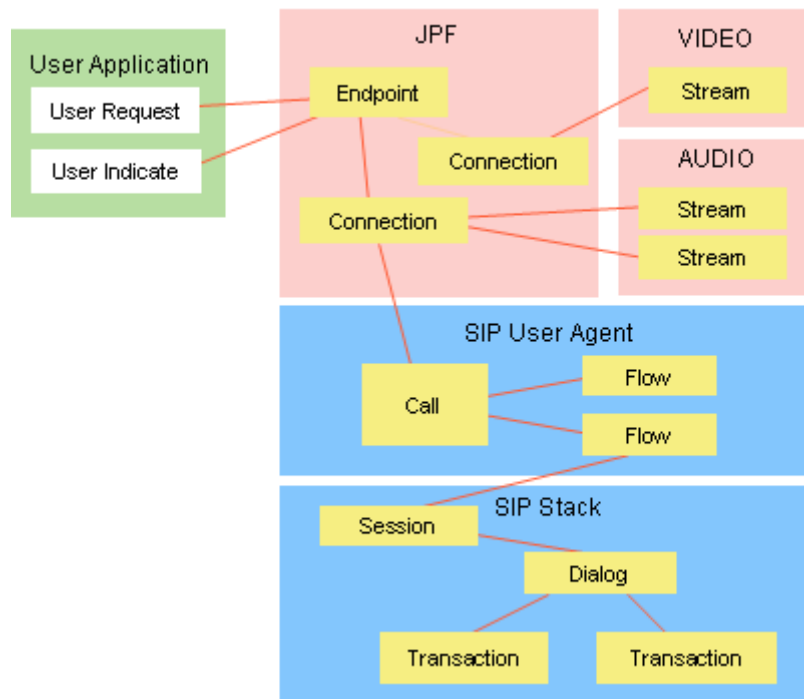


Figure 3-1 Basic Concepts of SIP Phone Multi-Service Framework

3.1 Endpoint

Endpoint is an element representing a user. It may be a registered Softphone user, a SIP phone, or a port of a SIP gateway. An Endpoint has some public attributes of equipments, such as local IP, local SIP signal port, cofig information of NAT and STUN. In addition, it has several special attributes as user Uri, account and the relating password. Each Endpoint may be responsible for different services as basic call, conference call, instant message and etc.

Users can register an Endpoint or get its information through interfaces provided by the Endpoint. Specific services depend on specific Connections.

3.2 Connection

Connection represents the relations between two Endpoints, as basic call, call transfer, and instant message. A Connection provides services that do not interfere with other services. For example a basic call can be kept or transferred. Instant messages can be sent through different connections from an Endpoint.

As for a three-part conference, an Endpoint has more than one connection. Each connection indicates a call to another Endpoint. The service for combining all of the audio streams is activated the moment a three-pary conference is established.

Each Connection keeps state information of the two Endpoints at each end of the Connection. Users use Connection interfaces to get the IP addresses of the two Endpoints and the state of the Connection. There are also other Connection interfaces for services as holding calls, call transfer and etc.

Each Connection is identified by a unique ID. Users get the ID when they are trying to establish a Connection. If users want to do some operations on a Connection, then its ID is necessary. So the ID should be well-preserved.

Users can provide a handle and make it relate to a newly created Connection. So when an event happened, users can get the handle related to the event through the Connection.

4. User Interfaces

4.1 Structures

4.1.1 ST_ZOS_SSTR

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

4.1.2 ST_JPF_CFG

```

/* jphone framework config info */
typedef struct tagJPF_CFG
{
    /* log info */
    ZCHAR *pcLogFileName;          /* log file name */
    ZULONG dwLogOpt;              /* log option */
    ZULONG dwLogLevel;           /* log level */
    ZULONG dwLogBufSize;         /* log buffer size */
    ZCHAR acLogPath[JPF_LOG_PATH_LEN]; /* log path */

    /* task config */
    ZINT iTaskPriority;           /* task priority */
    ZULONG dwTaskStackSize;      /* task stack size */
    ZULONG dwTaskQueueSize;     /* task queue size */
    ZULONG dwTaskTimerNum;      /* task timer number */

    /* endpoint resource */
    ZULONG dwEndpMaxNum;         /* endpoint max number */
    ZULONG dwEndpIncNum;        /* endpoint increase number */

    /* connection resource */
    ZULONG dwConnMaxNum;        /* connection max number */
    ZULONG dwConnIncNum;        /* connection increase number */

    /* subscription resource */
    ZULONG dwSubsMaxNum;        /* subscription max number */
    ZULONG dwSubsIncNum;        /* subscription increase number */
} ST_JPF_CFG;

```

4.1.3 ST_SIP_HOST

```

typedef struct tagSIP_HOST
{
    ZUCHAR ucPres;              /* present flag */
    ZUCHAR ucType;              /* host type EN_SIP_HOST */
    ZUCHAR aucSpare[2];         /* for 32 bit alignment */
    union
    {
        ST\_ZOS\_SSTR stName;      /* hostname */
        ZULONG dwIpv4Addr;       /* IPv4address */
        ZUCHAR aucIpv6Addr[ZINET_IPV6_ADDR_SIZE]; /* IPv6reference */
    } u;
} ST_SIP_HOST;

```

4.1.4 ST_SIP_HOST_PORT

```
typedef struct tagSIP_HOST_PORT
{
    ZUCHAR ucPres;           /* present flag */
    ZUCHAR ucPortPres;      /* Port present flag */
    ZUCHAR aucSpare[2];     /* for 32 bit alignment */
    ST\_SIP\_HOST stHost;      /* host */
    ZULONG dwPort;         /* port */
} ST_SIP_HOST_PORT;
```

4.1.5 ST_SUA_ENDP_URI

```
typedef struct tagSUA_ENDP_URI
{
    ZUCHAR ucUriType;       /* uri type EN_SUA_ENDP_URI_TYPE */
    ZUCHAR aucSpare[3];     /* for 32 bit alignment */
    ST\_ZOS\_SSTR stUserName;  /* display name */
    ST\_ZOS\_SSTR stUserInfo; /* user info */
    ST\_SIP\_HOST\_PORT stHostPort; /* host port */
} ST_SUA_ENDP_URI;
```

4.1.6 PFN_JPFEVNTPROC

```
/* event notification deliver event */
typedef ZINT (*PFN_JPFEVNTPROC)(ZUCHAR ucEvtType, ZVOID *pEvent);
```

4.1.7 ST_JPF_DB_VCODEC

```

typedef struct tagJPF_DB_VCODEC
{
    ZUCHAR ucUsed;                /* used flag */
    ZUCHAR ucEncoding;            /* codec encoding type */
    ZUCHAR aucSpare[2];           /* for 32 bits alignment */
    ZCHAR acName[JPF_NAME_LEN];   /* codec name */
    ZULONG dwPayload;             /* codec payload */
    ZULONG dwBitRate;            /* codec bitrate */
    union
    {
        ST_JPF_H261_OPTS stH261;  /* h261 options */
        ST_JPF_H263_OPTS stH263;  /* h263 options */
        ST_JPF_H264_OPTS stH264;  /* h264 options */
        ST_JPF_MPEG4_OPTS stMpeg4; /* mpeg4 options */
    } u;
} ST_JPF_DB_VCODEC;

typedef struct tagJPF_H261_OPTS
{
    ZUCHAR ucMpiCnt;              /* mpi count */
    ZUCHAR aucSpare[3];          /* for 32 bits alignment */
    ST_JPF_MPI astMpi[JPF_H261_MPI_MAX]; /* related mpi to picture size */
} ST_JPF_H261_OPTS;

typedef struct tagJPF_H263_OPTS
{
    ZUCHAR ucMpiCnt;              /* mpi count */
    ZUCHAR aucSpare[3];          /* for 32 bits alignment */
    ZULONG dwMaxBitRate;         /* max bit rate */
    ST_JPF_MPI astMpi[JPF_H263_MPI_MAX]; /* related mpi to picture size */
    ZULONG dwXRes;               /* customized x resolution */
    ZULONG dwYRes;               /* customized y resolution */
} ST_JPF_H263_OPTS;

typedef struct tagJPF_H264_OPTS
{
    ZUCHAR ucMpiCnt;              /* mpi count */
    ZUCHAR aucSpare[3];          /* for 32 bits alignment */
    ST_JPF_MPI astMpi[JPF_H264_MPI_MAX]; /* related mpi to picture size */
    ZULONG dwXRes;               /* customized x resolution */
    ZULONG dwYRes;               /* customized y resolution */
} ST_JPF_H264_OPTS;

typedef struct tagJPF_MPEG4_OPTS

```

```

{
    ZUCHAR ucCifPres;          /* CIF present flag */
    ZUCHAR aucSpare[3];       /* for 32 bit alignment */
} ST_JPF_MPEG4_OPTS;

```

4.1.8 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;

```

4.2 Config Interfaces

Here are several functions related to JPF configuration before it is operating.

4.2.1 Jpf_CfgGetTaskPriority

Gets the task priority. The default value is ZTASK_PRIORITY_NORMAL.

```
ZINT Jpf_CfgGetTaskPriority(ZINT *piPriority);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZINT *piPriority
The task priority.
```

[Return value]

Returns ZOK.

4.2.2 Jpf_CfgGetTaskStackSize

Gets the task size. The default value is 0.

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

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwStackSize

The task size.

[Return value]

Returns ZOK.

4.2.3 Jpf_CfgGetTaskQueueSize

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

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

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwQueueSize

The task queue size.

[Return value]

Returns ZOK.

4.2.4 Jpf_CfgGetLogLevel

Gets the log level. The default level is ZLOG_LEVEL_ALL.

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

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwLevel
```

The log level.

[Return value]

Returns ZOK.

4.2.5 Jpf_CfgGetEndpNum

Gets the endpoint number. The default number is 2.

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

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwNum
```

The endpoint number.

[Return value]

Returns ZOK.

4.2.6 Jpf_CfgGetConnNum

Gets the connection number. The default number is 6.

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

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwNum
```

The connection number.

[Return value]

Returns ZOK.

4.2.7 Jpf_CfgGetSubsNum

Gets the subscription number. The default number is 2.

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

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwNum
```

The subscription number.

[Return value]

Returns ZOK.

4.2.8 Jpf_CfgSetTaskPriority

Sets the task priority.

```
ZINT Jpf_CfgSetTaskPriority(ZINT iPriority);
```

[Parameters]

Input parameters:

```
ZINT iPriority
```

The task priority.

Output parameters:

None.

[Return value]

Returns ZOK.

4.2.9 Jpf_CfgSetTaskStackSize

Sets the task stack size.

```
ZINT Jpf_CfgSetTaskStackSize(ZULONG dwStackSize);
```

[Parameters]

Input parameters:

ZULONG dwStackSize
The stack size.

Output parameters:

None.

[Return value]

Returns ZOK.

4.2.10 Jpf_CfgSetTaskQueueSize

Sets the queue size.

```
ZINT Jpf_CfgSetTaskQueueSize(ZULONG dwQueueSize);
```

[Parameters]**Input parameters:**

ZULONG dwQueueSize
The queue size.

Output parameters:

None.

[Return value]

Returns ZOK.

4.2.11 Jpf_CfgSetLogLevel

Sets the log level.

```
ZINT Jpf_CfgSetLogLevel(ZULONG dwLevel);
```

[Parameters]**Input parameters:**

ZULONG dwLevel
The log level.

Output parameters:

None.

[Return value]

Returns ZOK.

4.2.12 Jpf_CfgSetEndpNum

Sets the endpoint number.

```
ZINT Jpf_CfgSetEndpNum( ZULONG dwNum );
```

[Parameters]

Input parameters:

ZULONG dwNum
The endpoint number.

Output parameters:

None.

[Return value]

Returns ZOK.

4.2.13 Jpf_CfgSetConnNum

Sets the connection number.

```
ZINT Jpf_CfgSetConnNum( ZULONG dwNum );
```

[Parameters]

Input parameters:

ZULONG dwNum
The connection number.

Output parameters:

None.

[Return value]

Returns ZOK.

4.2.14 Jpf_CfgSetSubsNum

Sets the subscription number.

```
ZINT Jpf_CfgSetSubsNum( ZULONG dwNum );
```

[Parameters]

Input parameters:

ZULONG dwNum
The subscription number.

Output parameters:

None.

[Return value]

Returns ZOK.

4.3 Task Interfaces

4.3.1 Jpf_Start

This function is responsible for starting JPF task. It will run the task according to configuration parameters, such as log option, task priority, queue size and resources number. These parameters can be configured by call Jpf_CfgInit to initialize the original information, then user can reset the global variable g_stJpfCfg member parameter.

```
ZINT Jpf_Start( )
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZINT main()
{
    /* init zos config */
    Zos_SysCfgInit();

    /* zos system init */
    if (Zos_SysInit() != ZOK)
        return -1;

    /* jpf database init */
    Jpf_DbInit(JCPE_CFG_FILE_NAME);

    /* sdp abnf init */
    if (Sdp_AbnfInit() != ZOK)
        return -1;

    /* start utal */
    if (Utal_Start() != ZOK)
        return -1;

    /* start sip stack */
    if (Sip_Start() != ZOK)
        return -1;

    /* start rtp stack */
    if (Rtp_Start() != ZOK)
        return -1;

    /* start sua */
    if (Sua_Start() != ZOK)
        return -1;

    /* start audio */
    if (Audio_Start() != ZOK)
        return -1;

    /* start dns */
    if (Dns_Start() != ZOK)
        return -1;

    /* start stun */
    if (Stun_Start() != ZOK)
        return -1;

    /* start jpf */
    if (Jpf_Start() != ZOK)
```

```

        return -1;

    /* start jcphone */
    if (Jcpe_Start() != ZOK)
        return -1;

    /* init zos shell */
    if (Zsh_Init(ZNULL) != ZOK)
        return -1;

    return 0;
}

```

4.3.2 Jpf_Stop

Stops the JPF task.

```
ZVOID Jpf_Stop();
```

[Parameters]

Input parameters

None.

Output parameters:

None.

[Return value]

None.

4.4 Event Interfaces

4.4.1 Jpf_EvntRegCallback

Users have to provide a callback function. In this callback function user should handle for registration, connection and mwi status reporting. After registration, JPF will inform user when these event happens. It can be found in jpf_evnt.h.

```
ZINT Jpf_EvntRegCallback(PFN\_JPFEVNTPROC *pstEvntProc)
```

[Parameters]

Input parameters

[PFN_JPFEVNTPROC](#) *pstEvntProc

Users provide a callback function to handle JPF event.

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
/* jcphe framework event process */
ZINT Jcpe_ActFeProcEvt(ZUCHAR ucEvtType, ZVOID *pEvent)
{
...
    switch (ucEvtType)
    {
        case EN_JPF_EVNT_REG_STAT:
            ...
            break;
        case EN_JPF_EVNT_CONN_STAT:
            ...
            break;
        default:
            break;
    }

    return ZOK;
}

/* register call back */
Jpf_EvntRegCallback(Jcpe_ActFeProcEvt);
```

4.5 Upper User Interfaces

4.5.1 Endpoint Interfaces

This section includes all user interfaces relating to [Endpoint](#). These interfaces can be found in `Jpf_endp.h` and `Jpf_endp_util.h`. Interfaces declared in the `Jpf_endp.h` are related with call control, and in the `Jpf_endp_util.h` are the utilities for endpoint.

4.5.1.1 Call Control

4.5.1.1.1 Jpf_EndpCreate

Jpf_EndpCreate creates an [Endpoint](#).

```
ZINT Jpf_EndpCreate(ZINT iIndex, ZULONG dwCookie, ZULONG *pdwEndpId);
```

[Parameters]**Input parameters:**

```
ZINT iIndex
```

An index indicating a user account which the user tends to use.

```
ZULONG dwCookie
```

User cookie.

Output parameters:

```
ZULONG *pdwEndpId
```

Pointer to the ID of the newly created Endpoint.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwCookie;
```

```
ZULONG dwEndpId;
```

```
ZINT iIndex;
```

```
ZINT iRet;
```

```
.....
```

```
/* Create an Endpoint. */
```

```
iRet = Jpf_EndpCreate(iIndex, dwCookie, &dwEndpId);
```

4.5.1.1.2 Jpf_EndpDelete

Jpf_EndpDelete deletes an [Endpoint](#).

```
ZINT Jpf_EndpDelete(ZULONG dwEndpId);
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
```

The ID of the Endpoint which is to be deleted.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZINT iRet;
.....
/* Delete an Endpoint. */
iRet = Jpf_EndpDelete(dwEndpId);
```

4.5.1.1.3 Jpf_EndpReload

Jpf_EndpReload refreshes an [Endpoint](#) with the information from JPF db module.

```
ZINT Jpf_EndpReload(ZULONG dwEndpId)
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
The Endpoint where the user account is being used.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZINT iRet;
.....
/* Refresh the information of a user account which is being used at an Endpoint.
*/
iRet = Jpf_EndpReload(dwEndpId);
```

4.5.1.1.4 Jpf_EndpReg

Jpf_EndpReg perform register on a server with a specific Endpoint information.

```
ZINT Jpf_EndpReg(ZULONG dwEndpId);
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
The ID of the Endpoint which is to be registered.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZINT iRet;
.....
/* Register an Endpoint with a server. */
iRet = Jpf_EndpReg(dwEndpId);
```

4.5.1.1.5 Jpf_EndpUnreg

Jpf_EndpUnreg deregisters an [Endpoint](#) on a server.

```
ZINT Jpf_EndpUnreg(ZULONG dwEndpId)
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
The ID of the Endpoint which is to be deregistered.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZINT iRet;
.....
/* Deregister an Endpoint. */
iRet = Jpf_EndpReg(dwEndpId);
```

4.5.1.1.6 Jpf_EndpSubsMwi

Jpf_EndpMwiSubs subscribes MWI service on a server with an endpoint ID.

```
ZINT Jpf_EndpSubsMwi(ZULONG dwEndpId)
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
ID of the Endpoint.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZINT iRet;
.....
/* Subscribe mwi service. */
iRet = Jpf_EndpSubsMwi(dwEndpId);
```

4.5.1.1.7 Jpf_EndpUnsubsMwi

Jpf_EndpMwiUnsubs unsubscribes endpoint's MWI (message wait indication) service on a server.

```
ZINT Jpf_EndpUnsubsMwi(ZULONG dwEndpId)
```

[Parameters]**Input parameters:**

```
ZULONG dwEndpId
ID of the Endpoint.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZINT iRet;
.....
/* unsubscribe MWI service. */
iRet = Jpf_EndpUnsubsMwi(dwEndpId);
```

4.5.1.2 Utilities**4.5.1.2.1 Jpf_EndpGetCookie**

Jpf_EndpGetCookie gets the user cookie of an [Endpoint](#).

```
ZINT Jpf_EndpGetCookie(ZULONG dwEndpId, ZULONG *pdwCookie)
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId  
The ID of the Endpoint.
```

Output parameters:

```
ZULONG *pdwCookie  
The user cookie of the Endpoint on success.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwCookie;  
ZULONG dwEndpId;  
ZINT iRet;  
.....  
/* Get the user cookie. */  
iRet = Jpf_EndpGetCookie(dwEndpId, &dwCookie);
```

4.5.1.2.2 Jpf_EndpGetEndpId

Gets the endpoint ID by the index.

```
ZINT Jpf_EndpGetEndpId(ZINT iIndex, ZULONG *pdwEndpId);
```

[Parameters]

Input parameters:

```
ZINT iIndex  
The index.
```

Output parameters:

```
ZULONG *pdwEndpId  
The endpoint ID.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

4.5.1.2.3 Jpf_EndpGetLocalAddr

Jpf_EndpGetLocalAddr gets the local address, including IP address and port number.

```
ZINT Jpf_EndpGetLocalAddr(ZULONG dwEndpId,
                           ST\_SIP\_HOST\_PORT **ppstLocalAddr);
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
ID of the Endpoint.
```

Output parameters:

```
ST\_SIP\_HOST\_PORT **ppstLocalAddr
The local address on success.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ST\_SIP\_HOST\_PORT *pstLocalAddr;
ZINT iRet;
.....
/* Get the local address.*/
iRet = Jpf_EndpGetLocalAddr(dwEndId, &pstLocalAddr);
```

4.5.1.2.4 Jpf_EndpGetLocalUri

Jpf_EndpGetLocalUri gets the local URI.

```
ZINT Jpf_EndpGetLocalUri(ZULONG dwEndpId,
                          ST\_SUA\_ENDP\_URI **ppstLocalUri);
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
ID of the Endpoint.
```

Output parameters:

```
ST\_SUA\_ENDP\_URI **ppstLocalUri
The local URI on success.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwEndpId;
ST\_SUA\_ENDP\_URI *pstLocalUri;
ZINT iRet;
.....
/* Get the local URI. */
iRet = Jpf_EndpGetLocalUri(dwEndpId, &pstLocalUri);

```

4.5.1.2.5 Jpf_EndpGetRegState

Jpf_EndpGetRegState gets the register state of an [Endpoint](#).

```
ZINT Jpf_EndpGetRegState(ZULONG dwEndpId, ZINT *piState)
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId
ID of the Endpoint.
```

Output parameters:

```
ZINT *piState
The register state of the Endpoint on success. There are five kinds of state:
EN_JPF_REG_STATE_IDLE,      EN_JPF_REG_STATE_REGING,      EN_JPF_REG_STATE_REGED,
EN_JPF_REG_STATE_UNREGING, EN_JPF_REG_STATE_NUM.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwEndpId;
ZINT iState;
ZINT iRet;
....
/* Get the register state of an Endpoint. */
iRet = Jpf_EndpGetRegState(dwEndpId, &iState);

```

4.5.1.2.6 Jpf_EndpGetRegDate

Gets the registered date.

```
ZINT Jpf_EndpGetRegDate(ZULONG dwEndpId, ST\_ZOS\_SYS\_TIME *pstTime);
```

[Parameters]

Input parameters:

ZULONG dwEndpId
The endpoint ID.

Output parameters:

[ST_ZOS_SYS_TIME](#) *pstTime
The registered date.

4.5.1.2.7 Jpf_EndpGetConnNum

Jpf_EndpGetConnNum gets the number of [Connections](#) an [Endpoint](#) has.

```
ZINT Jpf_EndpGetConnNum(ZULONG dwEndpId, ZULONG *pdwConnNum);
```

[Parameters]

Input parameters:

ZULONG dwEndpId
ID of the Endpoint.

Output parameters:

ZULONG *pdwConnNum
The number of [Connections](#) the Endpoint has on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZULONG dwConnNum;
ZINT iRet;
.....
/* Get the number of Connections an Endpoint has. */
iRet = Jpf_EndpGetConnNum(dwEndpId, &dwConnNum);
```

4.5.1.2.8 Jpf_EndpGetConnItem

Jpf_EndpGetConnItem gets the ID of a [Connection](#) by an index.

```
ZINT Jpf_EndpGetConnItem(ZULONG dwEndpId, ZULONG dwIndex,
                          ZULONG *pdwConnId)
```

[Parameters]

Input parameters:

ZULONG dwEndpId
ID of the Endpoint.

ZULONG dwIndex
The index by which the ID of the Connection is obtained.

Output parameters:

ZULONG *pdwConnId
ID of the Connection that has just been found on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwEndpId;
ZULONG dwIndex;
ZULONG dwConnId;
ZINT iRet;
.....
/* Get the ID of a Connection. */
iRet = Jpf_EndpGetConnItem(dwEndpId, dwIndex, &dwConnId);
```

4.5.1.2.9 Jpf_EndpGetTalking

Jpf_EndpGetTalking gets whether the specific endpoint is in talking state.

```
ZINT Jpf_EndpGetTalking(ZULONG dwEndpId, ZBOOL *pbTalking)
```

[Parameters]**Input parameters:**

ZULONG dwEndpId
ID of the Endpoint.

Output parameters:

ZBOOL *pbTalking
It will be set to ZTRUE if the endpoint is in talking state, or it will be set to ZFALSE if the endpoint is not in talking state.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwEndpId;
ZBOOL bTalking;
ZINT iRet;
.....
/* Get whether the specific endpoint is in talking state. */
iRet = Jpf_EndpGetTalking(dwEndpId, &bTalking);

```

4.5.1.2.10 Jpf_EndpGetVMsgCount

Jpf_EndpGetVMsgCount gets the number of voice messages.

```

ZINT Jpf_EndpGetVMsgCount(ZULONG dwEndpId, ZULONG *pdwNewCount,
                          ZULONG *pdwOldCount)

```

[Parameters]

Input parameters:

```

ZULONG dwEndpId
ID of the Endpoint.

```

Output parameters:

```

ZULONG *pdwNewCount
The number of voice messages which haven't been read.

ZULONG *pdwOldCount
The number of voice messages which have been read.

```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwEndpId;
ZULONG dwNewCount;
ZULONG dwOldCount;
ZINT iRet;
.....
/* Get the number of voice messages. */
iRet = Jpf_EndpGetVMsgCount(dwEndpId, &dwNewCount, &dwOldCount);

```

4.5.2 Connection Interfaces

This section includes all user interfaces relating to [Connection](#). These interfaces can be found in `Jpf_conn.h` and `Jpf_conn_util.h`. Interfaces declared in the `Jpf_conn.h` are related with call control, and in the `Jpf_conn_util.h` are utilities for connection.

4.5.2.1 Call Control

4.5.2.1.1 Jpf_ConnOpen

Jpf_ConnOpen creates an outgoing [Connection](#) at an endpoint. It first checks whether the endpoint is valid.

```
ZINT Jpf_ConnOpen(ZULONG dwEndpId, ZULONG dwCookie, ZULONG *pdwConnId)
```

[Parameters]

Input parameters:

```
ZULONG dwEndpId  
ID of the endpoint.
```

```
ZULONG dwCookie  
The user cookie.
```

Output parameters:

```
ZULONG *pdwConnId  
The newly created connection.
```

[Return value]

Returns ZOK on success, or ZFAILED if the endpoint is not valid, or the input parameter *pdwConnId* is ZNULL.

[Example]

```
ZULONG dwConnId;  
ZULONG dwEndpId;  
ZINT iRet;  
  
.....  
/* Open an outgoing connection at an endpoint. */  
iRet = Jpf_ConnOpen(dwEndpId, ZMAXULONG, &dwConnId);
```

4.5.2.1.2 Jpf_ConnCall

Jpf_ConnCall uses a [Connection](#) between the local [Endpoint](#) and the remote Endpoint that connects to the callee (another user application) to make an outgoing call.

```
ZINT Jpf_ConnCall(ZULONG dwConnId, ST\_ZOS\_SSTR *pstCalleeUri)
```

[Parameters]

Input parameters:

ZULONG dwConnId
The Connection ID.

[ST_ZOS_SSTR](#) *pstCalleeUri
The callee's URI.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ST\_ZOS\_SSTR *pstCalleeUri;  
ZULONG dwConnId;  
  
.....  
/* Make an outgoing call. */  
iRet = Jpf_ConnCall(dwConnId, pstCalleeUri);
```

4.5.2.1.3 Jpf_ConnAnswer

Jpf_ConnAnswer answers an incoming [Connection](#).

```
ZINT Jpf_ConnAnswer(ZULONG dwConnId, ZULONG dwCookie)
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the incoming Connection.

ZULONG dwCookie
The user cookie.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwConnId;
ZULONG dwCookie;
ZINT iRet;
.....
/* Answer an incoming Connection. */
iRet = Jpf_ConnAnswer(dwConnId, dwCookie);

```

4.5.2.1.4 Jpf_ConnTerminate

Jpf_ConnTerminate terminates a [Connection](#).

```
ZINT Jpf_ConnTerminate(ZULONG dwConnId, ZUCHAR ucReason);
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the incoming Connection.
```

```
ZUCHAR ucReason
```

The reason to terminate. The reasons can be one of the listed below:

```

EN_JPF_TERM_REASON_NORMAL /* normal session termination */
EN_JPF_TERM_REASON_DND /* do not disturb or not available */
EN_JPF_TERM_REASON_BUSY /* user or ui are busy */
EN_JPF_TERM_REASON_REJ /* user rejected an incoming session */
EN_JPF_TERM_REASON_TIMEOUT /* session not answered in some time */
EN_JPF_TERM_REASON_CF /* call forward */
EN_JPF_TERM_REASON_NOT_SUPTTED /* some feature is not supported */

```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwConnId;
ZUCHAR ucReason;
ZINT iRet;
.....
/* Terminates a Connection. */
iRet = Jpf_ConnTerminate(dwConnId, ucReason);

```

4.5.2.1.5 Jpf_ConnHold

Jpf_ConnHold holds a [Connection](#).

```
ZINT Jpf_ConnHold(ZULONG dwConnId)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId  
ID of the Connection which Jpf_ConnHold is to hold.
```

Output parameters:

```
None.
```

[Return value]

```
Returns ZOK on success, or ZFAILED on failure.
```

[Example]

```
ZULONG dwConnId;  
ZINT iRet;  
.....  
/* Hold a Connection. */  
iRet = Jpf_ConnHold(dwConnId);
```

4.5.2.1.6 Jpf_ConnUnhold

Jpf_ConnUnhold stops holding a [Connection](#).

```
ZINT Jpf_ConnUnhold(ZULONG dwConnId);
```

[Parameters]

Input parameters:

```
ZULONG dwConnId  
ID of the Connection.
```

Output parameters:

```
None.
```

[Return value]

```
Returns ZOK on success, or ZFAILED on failure.
```

[Example]

```

ZULONG dwConnId;
ZINT iRet;
.....
/* Stop holding a Connection. */
iRet = Jpf_ConnUnhold(dwConnId);

```

4.5.2.1.7 Jpf_ConnCHold

Jpf_ConnCHold calls another callee and hold the call to the first callee.

```
ZINT Jpf_ConnCHold(ZULONG dwConnId, ST\_ZOS\_SSTR *pstHoldUri)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection between the local Endpoint and the first remote Endpoint.

ST\_ZOS\_SSTR *pstHoldUri
The URI of the second callee.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwConnId;
ST\_ZOS\_SSTR *pstHoldUri;
ZINT iRet;
.....
/* Consulation hold */
iRet = Jpf_ConnCHold(dwConnId, pstHoldUri);

```

4.5.2.1.8 Jpf_ConnUTrsf

Jpf_ConnUTrsf establishes a call to a second callee and stops the call to the first callee.

```
ZINT Jpf_ConnUTrsf(ZULONG dwConnId, ST\_ZOS\_SSTR *pstTrsfUri);
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the Connection.

[ST_ZOS_SSTR](#) *pstTrsfUri
The URI of the second callee.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwConnId;
ST\_ZOS\_SSTR *pstTrsfUri;
ZINT iRet;
.....
/* Transfer a call */
iRet = Jpf_ConnUTrsf(dwConnId, pstTrsfUri);
```

4.5.2.1.9 Jpf_ConnATrsf

Jpf_ConnATrsf makes an attended transfer. It establishes a call to a second callee and holds the call to the first callee.

```
ZINT Jpf_ConnATrsf(ZULONG dwConnId, ST\_ZOS\_SSTR *pstTrsfUri);
```

[Parameters]**Input parameters:**

ZULONG dwConnId
ID of the Connection.

[ST_ZOS_SSTR](#) *pstTrsfUri
The URI of the second callee.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```

ZULONG dwConnId;
ST_ZOS_SSTR *pstTrsfUri;
ZINT iRet;
.....
/* Transfer a call */
iRet = Jpf_ConnATrsf(dwConnId, pstTrsfUri);

```

4.5.2.1.10 Jpf_ConnRedirect

Jpf_ConnRedirect redirects the connection to a 3rd party user.

```
ZINT Jpf_ConnRedirect(ZULONG dwConnId, ST_ZOS_SSTR *pstPUri)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ST_ZOS_SSTR *pstPUri
A pointer to URI of 3rd party user.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwConnId;
ST_ZOS_SSTR *pstPUri;
ZINT iRet;
.....
/* Redirect the connection to a 3rd party user. */
iRet = Jpf_ConnRedirect(dwConnId, pstPUri);

```

4.5.2.1.11 Jpf_ConnReferAcpt

Jpf_ConnReferAcpt accepts a Refer request.

```
ZINT Jpf_ConnReferAcpt(ZULONG dwConnId)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ZINT iRet;
.....
/* Accept a Refer request. */
iRet = Jpf_ConnReferAcpt(dwConnId);
```

4.5.2.1.12 Jpf_ConnReferRej

Jpf_ConnReferRej rejects a Refer request.

```
ZINT Jpf_ConnReferRej(ZULONG dwConnId)
```

[Parameters]**Input parameters:**

```
ZULONG dwConnId
ID of the Connection.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ZINT iRet;
.....
/* Reject a Refer request. */
iRet = Jpf_ConnReferRej(dwConnId);
```

4.5.2.1.13 Jpf_ConnHeartbeat

Jpf_ConnHeartbeat sends heart beat messages.

```
ZINT Jpf_ConnHeartbeat(ZULONG dwConnId)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ZINT iRet;
.....
/* Send heart beat messages. */
iRet = Jpf_ConnHeartbeat(dwConnId);
```

4.5.2.1.14 Jpf_ConnAConf

Jpf_ConnAConf adds a 3rd party user to a conference.

```
ZINT Jpf_ConnAConf(ZULONG dwConnId, ST_ZOS_SSTR *pstPUri)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.

ST_ZOS_SSTR *pstPUri
Pointer to the 3rd party user's uri which should added to the conference.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwConnId
ST_ZOS_SSTR *pstPUri;
ZINT iRet;
.....
/* add a 3rd party user to a conference. */
iRet = Jpf_ConnAConf(dwConnId, pstPUri);

```

4.5.2.1.15 Jpf_ConnPickUp

Jpf_ConnPickUp picks up a specific call on another terminal.

```
ZINT Jpf_ConnPickUp(ZULONG dwConnId, ST_ZOS_SSTR *pstPUri)
```

[Parameters]

Input parameters:

```

ZULONG dwConnId
ID of the Connection.

ST_ZOS_SSTR *pstPUri
Pointer to the user's uri which would be picked up.

```

Output parameters:

None.

[Return value]

```

ZOK: Operation succeeded.
ZFAILED: Operation failed.

```

[Example]

```

ZULONG dwConnId
ST_ZOS_SSTR *pstPUri;
ZINT iRet;
.....
/* pick up call on another phone. */
iRet = Jpf_ConnPickUp(dwConnId, pstPUri);

```

4.5.2.1.16 Jpf_ConnPlayTone

Jpf_ConnPlayTone starts to play audio data.

```
ZINT Jpf_ConnPlayTone(ZUCHAR ucToneType, ZULONG dwLen)
```

[Parameters]

Input parameters:

ZUCHAR ucToneType

Type of the audio data. The types of audio data are shown as the following:

```
EN_AUDIO_TONE_DIAL
    EN_AUDIO_TONE_BUSY
    EN_AUDIO_TONE_RING
    EN_AUDIO_TONE_RING_BACK
    EN_AUDIO_TONE_CONGESTION
    EN_AUDIO_TONE_SPEC_INFO
    EN_AUDIO_TONE_WARN
    EN_AUDIO_TONE_CALL_WAIT
    EN_AUDIO_TONE_CALLER_WAIT
    EN_AUDIO_TONE_ON_HOOK
    EN_AUDIO_TONE_0
    EN_AUDIO_TONE_1
    EN_AUDIO_TONE_2
    EN_AUDIO_TONE_3
    EN_AUDIO_TONE_4
    EN_AUDIO_TONE_5
    EN_AUDIO_TONE_6
    EN_AUDIO_TONE_7
    EN_AUDIO_TONE_8
    EN_AUDIO_TONE_9
    EN_AUDIO_TONE_STAR
    EN_AUDIO_TONE_POUND
    EN_AUDIO_TONE_A
    EN_AUDIO_TONE_B
    EN_AUDIO_TONE_C
EN_AUDIO_TONE_D
```

ZULONG dwLen

Indicats how many milliseconds the tone should be played each time.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZUCHAR ucToneType;  
ZINT iRet;  
ZULONG dwLen;  
.....  
ucToneType = EN_AUDIO_TONE_RING_BACK;  
/* Play audio data of a specific type. */  
iRet = Jpf_ConnPlayTone(ucToneType, dwLen);
```

4.5.2.1.17 Jpf_ConnStopTone

Jpf_ConnStopTone stops playing audio data.

```
ZINT Jpf_ConnStopTone()
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.

4.5.2.1.18 Jpf_ConnDtmf

Jpf_ConnStartDtmf starts sending DTMF signals.

```
ZINT Jpf_ConnDtmf(ZULONG dwConnId, ZUCHAR ucDtmfType)
```

[Parameters]

Input parameters:

ZULONG dwConnId

ID of the connection.

ZUCHAR ucDtmfType

Type of a DTMF signal. The valid DTMF types are:

EN_AUDIO_DTMF_0

EN_AUDIO_DTMF_1

EN_AUDIO_DTMF_2

EN_AUDIO_DTMF_3

EN_AUDIO_DTMF_4

EN_AUDIO_DTMF_5

EN_AUDIO_DTMF_6

EN_AUDIO_DTMF_7

EN_AUDIO_DTMF_8

EN_AUDIO_DTMF_9

EN_AUDIO_DTMF_STAR

EN_AUDIO_DTMF_POUND

EN_AUDIO_DTMF_A

EN_AUDIO_DTMF_B

EN_AUDIO_DTMF_C

EN_AUDIO_DTMF_D

EN_AUDIO_DTMF_FLASH

EN_AUDIO_DTMF_OFF_HOOK = 64

EN_AUDIO_DTMF_ON_HOOK = 65

EN_AUDIO_DTMF_DIAL = 66

EN_AUDIO_DTMF_PABX_DT = 67/

EN_AUDIO_DTMF_SPECIAL_DT = 68

EN_AUDIO_DTMF_SECOND_DT = 69

EN_AUDIO_DTMF_RING_BACK = 70

EN_AUDIO_DTMF_SPEC_RT = 71

EN_AUDIO_DTMF_BUSY = 72

EN_AUDIO_DTMF_CONG = 73

EN_AUDIO_DTMF_SPEC_INFO = 74

EN_AUDIO_DTMF_COMF = 75

EN_AUDIO_DTMF_HOLD = 76

EN_AUDIO_DTMF_RECORD = 77

EN_AUDIO_DTMF_CALLER_WAIT = 78

EN_AUDIO_DTMF_CALL_WAIT = 79

EN_AUDIO_DTMF_PAY = 80

EN_AUDIO_DTMF_POS_IND = 81

EN_AUDIO_DTMF_NEG_IND = 82

EN_AUDIO_DTMF_WARN = 83

EN_AUDIO_DTMF_INTRUSION = 84

EN_AUDIO_DTMF_CALL_CARD = 85

EN_AUDIO_DTMF_PAYPHONE = 86

```
EN_AUDIO_DTMF_CAS = 87
EN_AUDIO_DTMF_OFF_HOOK_WARN = 88
EN_AUDIO_DTMF_RING = 89
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ZINT iRet;
.....
/* Start sending DTMF signals. */
iRet = Jpf_ConnDtmf(dwConnId, EN_AUDIO_DTMF_RING);
```

4.5.2.1.19 Jpf_ConnGetMuted

Jpf_ConnGetMuted gets the mute state of a Connection.

```
ZINT Jpf_ConnGetMuted(ZULONG dwConnId, ZBOOL *pbMuted)
```

[Parameters]**Input parameters:**

```
ZULONG dwConnId
ID of the connection.

ZBOOL *pbMuted
A pointer which will point to the mute state.
```

Output parameters:

```
ZBOOL *pbMuted
A pointer which points to the mute state.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwConnId;
ZBOOL bMuted;
ZINT iRet;
.....
/* Get the mute state for the audio stream. */
iRet = Jpf_ConnGetMuted(dwConnId, &bMuted);

```

4.5.2.1.20 Jpf_ConnSetMuted

Jpf_ConnSetMuted sets mute state of a Connection. If the Connection is not in the holding state, then *Jpf_ConnSetMuted* will clear the mute flag.

```
ZINT Jpf_ConnSetMuted(ZULONG dwConnId, ZBOOL bMuted)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the connection.
```

```
ZBOOL bMuted
The mute state.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwConnId;
ZBOOL bMuted;
ZINT iRet;
.....
/* Set the mute state. */
iRet = Jpf_ConnSetMuted(dwConnId, bMuted);

```

4.5.2.1.21 Jpf_ConnSetConf

Jpf_ConnSetConf sets the conference state of a Connection. The voice of the connection will be mixed if the conference state is setted. There can only support one conference at the same time.

```
ZINT Jpf_ConnSetConf(ZULONG dwConnId, ZBOOL bConf)
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the connection.

ZBOOL bConf
The conference state.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ZBOOL bMuted;
ZINT iRet;
.....
/* Set the mute state. */
iRet = Jpf_ConnSetMuted(dwConnId, bMuted);
```

4.5.2.1.22 Jpf_ConnEnableCap

Enables or disables one stream.

```
ZINT Jpf_ConnEnableCap(ZULONG dwConnId, ZUCHAR ucStrmType,
                      ZBOOL bEnable)
```

[Parameters]**Input parameters:**

ZULONG dwConnId
The connection ID.

ZUCHAR ucStrmType
Type of the stream.

ZBOOL bEnable
ZTRUE indicates that the stream will be enabled; ZFALSE indicates that the stream will be disabled.

Output parameters:

None.

[Return value]

Returns ZOK on success or ZFAILED on failure.

[Example]

```
ZULONG dwConnId;
ZUCHAR ucStrmType;
ZBOOL bEnable;
ZINT iRet;
.....
/* Enable or disable one stream. */
iRet = Jpf_ConnEnableCap(dwConnId, ucStrmType, bEnable);
```

4.5.2.1.23 Jpf_ConnGetHolded

Jpf_ConnGetHolded gets the holding state of a Connection.

```
ZINT Jpf_ConnGetHolded(ZULONG dwConnId, ZBOOL *pbHolded)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the connection.
```

```
ZBOOL *pbHolded
A pointer which will point to the holding state.
```

Output parameters:

```
ZBOOL *pbHolded
A pointer which points to the holding state.
```

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ZBOOL bHolded;
ZINT iRet;
.....
/* Gets the holding state. */
iRet = Jpf_ConnGetHolded(dwConnId, &bHolded);
```

4.5.2.1.24 Jpf_ConnRecPlayStart

Jpf_ConnRecPlayStart starts recording playout of a connection.

```
ZINT Jpf_ConnRecPlayStart(ZULONG dwConnId, ZCHAR *pcFileName,  
                          ZCHAR *pcCodec)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId  
ID of the connection.
```

```
ZCHAR *pcFileName  
Indicates the file name in which recording data saved.
```

```
ZCHAR *pcCodec  
Indicates the codec type of recording file.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;  
.....  
/* Start recording playout. */  
iRet = Jpf_ConnRecPlayStart(dwConnId, "rec.wav", "gsm");
```

4.5.2.1.25 Jpf_ConnRecPlayStop

Jpf_ConnRecPlayStop stops recording playout of a connection.

```
ZINT Jpf_ConnRecPlayStop(ZULONG dwConnId)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId  
ID of the connection.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;  
  
.....  
/* Stop recording playout. */  
iRet = Jpf_ConnRecPlayStop(dwConnId);
```

4.5.2.1.26 Jpf_ConnStartVideo

Starts playing video.

```
ZINT Jpf_ConnStartVideo(ZULONG dwConnId, ZULONG dwX, ZULONG dwY,  
                        ZULONG dwWidth, ZULONG dwHeight);
```

[Parameters]

Input parameters:

```
ZULONG dwConnId  
The connection ID.
```

```
ZULONG dwX  
The Abscissa.
```

```
ZULONG dwY  
The ordinate.
```

```
ZULONG dwWidth  
Width of the video window.
```

```
ZULONG dwHeight  
Height of the video window.
```

Output parameters:

None.

[Return value]

Returns ZOK on success or ZFAILED on failure.

[Example]

```

ZULONG dwConnId;
ZULONG dwX;
ZULONG dwY;
ZULONG dwWidth;
ZULONG dwHeight;
ZINT iRet;
.....
/* Start playing video.*/
iRet = Jpf_ConnStartVideo(dwConnId, dwX, dwY, dwWidth, dwHeight);

```

4.5.2.1.27 Jpf_ConnStopVideo

Stops playing video.

```
ZINT Jpf_ConnStopVideo(ZULONG dwConnId);
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
The connection ID.
```

Output parameters:

None.

[Return value]

Returns ZOK on success or ZFAILED on failure.

[Example]

```

ZULONG dwConnId;
ZINT iRet;
.....
/* Stop play video. */
iRet = Jpf_ConnStopVideo(dwConnId);

```

4.5.2.2 Utilities

4.5.2.2.1 Jpf_ConnGetEndpId

Jpf_ConnGetEndpId gets ID of the [Endpoint](#) to which a specific [Connection](#) belongs.

```
ZINT Jpf_ConnGetEndpId(ZULONG dwConnId, ZULONG *pdwEndpId)
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the connection.

ZULONG *pdwEndpId
A pointer which will point to ID of the Endpoint to which the Connection belongs.

Output parameters:

ZULONG *pdwEndpId
A pointer which points to ID of the Endpoint to which the Connection belongs.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ZULONG dwEndpId;
ZINT iRet;
.....
/* Get ID of the Endpoint to which a specific Connection belongs. */
iRet = Jpf_ConnGetEndpId(dwConnId, &dwEndpId);
```

4.5.2.2.2 Jpf_ConnGetCookie

Jpf_ConnGetCookie gets cookie of the user relating to a specific Connection.

```
ZINT Jpf_ConnGetCookie(ZULONG dwConnId, ZULONG *pdwCookie)
```

[Parameters]**Input parameters:**

ZULONG dwConnId
ID of the connection.

ZULONG *pdwCookie
A pointer which will point to cookie of the user relating to the Connection.

Output parameters:

ZULONG *pdwCookie
A pointer which points to cookie of the user relating to the Connection.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ZULONG dwCookie;
ZINT iRet;
.....
/* Get cookie of the user relating to a specific Connection. */
iRet = Jpf_ConnGetCookie(dwConnId, &dwCookie);
```

4.5.2.2.3 Jpf_ConnGetType

Jpf_ConnGetType gets type of a Connection.

```
ZINT Jpf_ConnGetType(ZULONG dwConnId, ZUCHAR *pucType)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ZUCHAR *pucType
A pointer which will point to type of the Connection.
```

Output parameters:

```
ZUCHAR *pucType
A pointer which points to type of the Connection.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ZUCHAR ucType;
ZINT iRet;
.....
/* Get type of a Connection. */
iRet = Jpf_ConnGetType(dwConnId, &ucType);
```

4.5.2.2.4 Jpf_ConnGetState

Jpf_ConnGetState gets the state of a Connection.

```
ZINT Jpf_ConnGetState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the Connection.

ZINT *piState
A pointer which will point to the state of the Connection.

Output parameters:

ZINT *piState
A pointer which points to the state of the Connection.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;  
ZINT iState;  
ZINT iRet;  
.....  
/* Get the state of a Connection. */  
iRet = Jpf_ConnGetState(dwConnId, &iState);
```

4.5.2.2.5 Jpf_ConnGetCHoldState

Jpf_ConnGetCHoldState gets the consultation state of a Connection.

```
ZINT Jpf_ConnGetCHoldState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]**Input parameters:**

ZULONG dwConnId
ID of the Connection.

ZINT *piState
A pointer which will point to the consultation state of the Connection.

Output parameters:

ZINT *piState
A pointer which points to the consultation state of the Connection.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the consultation state of a Connection. */
iRet = Jpf_ConnGetCHoldState(dwConnId, &iState);
```

4.5.2.2.6 Jpf_ConnGetUTrsfState

Jpf_ConnGetUTrsfState gets the unattended transfer state of a Connection.

```
ZINT Jpf_ConnGetUTrsfState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]

Input parameters:

ZULONG dwConnId
 ID of the Connection.

ZINT *piState
 A pointer which will point to the unattended transfer state. The unattended transfer states are:

```
EN_JPF_UTRSF_STATE_IDLE
  EN_JPF_UTRSF_STATE_TRSFING/
  EN_JPF_UTRSF_STATE_TRSFED
  EN_JPF_UTRSF_STATE_TERM
  EN_JPF_UTRSF_STATE_NUM
```

Output parameters:

ZINT *piState
 A pointer to the unattended transfer state.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```

ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the unattended transfer state. */
iRet = Jpf_ConnGetUTrsfState(dwConnId, &iState);

```

4.5.2.2.7 Jpf_ConnGetATrsfState

Jpf_ConnGetATrsfState gets the attended transfer state of a Connection.

```
ZINT Jpf_ConnGetATrsfState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ZINT *piState
```

A pointer which will point to the attended transfer state. The valid attended transfer states are:

```

EN_JPF_ATRSF_STATE_IDLE
EN_JPF_ATRSF_STATE_HOLDING1
EN_JPF_ATRSF_STATE_CALLING
EN_JPF_ATRSF_STATE_HOLDING2
EN_JPF_ATRSF_STATE_TRSFING
EN_JPF_ATRSF_STATE_TRSFED
EN_JPF_ATRSF_STATE_TERM
EN_JPF_ATRSF_STATE_NUM

```

Output parameters:

```
ZINT *piState
A pointer to the attended transfer state.
```

[Return value]

```

ZOK: Operation succeeded.
ZFAILED: Operation failed.

```

[Example]

```

ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the attended transfer state. */
iRet = Jpf_ConnGetATrsfState(dwConnId, &iState);

```

4.5.2.2.8 Jpf_ConnGetReferState

Jpf_ConnGetReferState gets the transfer state of a Connection.

```
ZINT Jpf_ConnGetReferState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ZINT *piState
A pointer which will point to the transfer state.
```

Output parameters:

```
ZINT *piState
A pointer to the transfer state.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the transfer state. */
iRet = Jpf_ConnGetReferState(dwConnId, &iState);

```

4.5.2.2.9 Jpf_ConnGetAudioState

Jpf_ConnGetAudioState gets the audio state of a Connection.

```
ZINT Jpf_ConnGetAudioState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ZINT *piState
A pointer which will point to the audio state.
```

Output parameters:

```
ZINT *piState
A pointer to the audio state.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the audio state. */
iRet = Jpf_ConnGetAudioState(dwConnId, &iState);
```

4.5.2.2.10 Jpf_ConnGetVideoState

Jpf_ConnGetVideoStat gets the video state of a Connection.

```
ZINT Jpf_ConnGetVideoState(ZULONG dwConnId, ZINT *piState)
```

[Parameters]**Input parameters:**

```
ZULONG dwConnId
ID of the Connection.
```

Output parameters:

```
ZINT *piState
A pointer to the video state.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the audio state. */
iRet = Jpf_ConnGetVideoState(dwConnId, &iState);

```

4.5.2.2.11 Jpf_ConnGetConfStat

Gets the conference state of a Connection.

```
ZINT Jpf_ConnGetConfStat(ZULONG dwConnId, ZINT *piStatus);
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
The connection ID.
```

Output parameters:

```
ZINT *piStatus
The conference state.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```

ZULONG dwConnId;
ZINT iState;
ZINT iRet;
.....
/* Get the conference state. */
iRet = Jpf_ConnGetConfStat(dwConnId, &iState);

```

4.5.2.2.12 Jpf_ConnGetLocalAddr

Jpf_ConnGetLocalAddr gets the local IP address and port.

```
ZINT Jpf_ConnGetLocalAddr(ZULONG dwConnId,
ST\_SIP\_HOST\_PORT **ppstLocalAddr)
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the Connection.

[ST_SIP_HOST_PORT](#) **ppstLocalAddr
A pointer which will point to the pointer to the local address and port.

Output parameters:

[ST_SIP_HOST_PORT](#) **ppstLocalAddr
A pointer which points to the pointer to the local address and port.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ST\_SIP\_HOST\_PORT *pstLocalAddr;
ZINT iRet;
.....
/* Get the local IP address and port. */
iRet = Jpf_ConnGetLocalAddr(dwConnId, &pstLocalAddr);
```

4.5.2.2.13 Jpf_ConnGetPeerAddr

Jpf_ConnGetPeerAddr gets the peer IP address and port.

```
ZINT Jpf_ConnGetPeerAddr(ZULONG dwConnId,
ST\_SIP\_HOST\_PORT **ppstPeerAddr)
```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the Connection.

[ST_SIP_HOST_PORT](#) **ppstLocalAddr
A pointer which will point to the pointer to the peer address and port.

Output parameters:

[ST_SIP_HOST_PORT](#) **ppstLocalAddr
A pointer which points to the pointer to the peer address and port.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ST_SIP_HOST_PORT *pstLocalAddr;
ZINT iRet;
.....
/* Get the peer IP address and port. */
iRet = Jpf_ConnGetPeerAddr(dwConnId, &pstLocalAddr);
```

4.5.2.2.14 Jpf_ConnGetLocalUri

Jpf_ConnGetLocalUri gets the local URI.

```
ZINT Jpf_ConnGetLocalUri(ZULONG dwConnId,
                         ST_SUA_ENDP_URI **ppstLocalUri)
```

[Parameters]

Input parameters:

ZULONG dwConnId
 ID of the Connection.

[ST_SUA_ENDP_URI](#) **ppstLocalUri
 A pointer which will point to the pointer to the local URI.

Output parameters:

[ST_SUA_ENDP_URI](#) **ppstLocalUri
 A pointer which points to the pointer to the local URI.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ST_SUA_ENDP_URI *pstLocalUri;
ZINT iRet;
.....
/* Get the local URI. */
iRet = Jpf_ConnGetLocalUri(dwConnId, &pstLocalUri);
```

4.5.2.2.15 Jpf_ConnGetPeerUri

Jpf_ConnGetPeerUri gets the peer URI.

```
ZINT Jpf_ConnGetPeerUri(ZULONG dwConnId,
                        ST\_SUA\_ENDP\_URI **ppstPeerUri)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ST\_SUA\_ENDP\_URI **ppstLocalUri
A pointer which will point to the pointer to the peer URI.
```

Output parameters:

```
ST\_SUA\_ENDP\_URI **ppstLocalUri
A pointer which points to the pointer to the peer URI.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwConnId;
ST\_SUA\_ENDP\_URI *pstLocalUri;
ZINT iRet;
.....
/* Get the peer URI. */
iRet = Jpf_ConnGetPeerUri(dwConnId, &pstLocalUri);
```

4.5.2.2.16 Jpf_ConnGetCHoldUri

Jpf_ConnGetCHoldUri gets the consultation URI.

```
ZINT Jpf_ConnGetCHoldUri(ZULONG dwConnId,
                          ST\_SUA\_ENDP\_URI **ppstCholdUri)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ST\_SUA\_ENDP\_URI **ppstLocalUri
A pointer which will point to the pointer to the consultation URI.
```

Output parameters:

```
ST_SUA_ENDP_URI **ppstLocalUri
```

A pointer which points to the pointer to the consultation URI.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG dwConnId;
ST_SUA_ENDP_URI *pstLocalUri;
ZINT iRet;
.....
/* Get the consultation URI. */
iRet = Jpf_ConnGetCHoldUri(dwConnId, &pstLocalUri);
```

4.5.2.2.17 Jpf_ConnGetReferToUri

Jpf_ConnGetReferToUri gets the URI to which a call is referred.

```
ZINT Jpf_ConnGetReferToUri(ZULONG dwConnId,
ST_SUA_ENDP_URI **ppstReferToUri)
```

[Parameters]

Input parameters:

```
ZULONG dwConnId
ID of the Connection.
```

```
ST_SUA_ENDP_URI **ppstLocalUri
```

A pointer which will point to the pointer to the URI to which a call is referred.

Output parameters:

```
ST_SUA_ENDP_URI **ppstLocalUri
```

A pointer which points to the pointer to the URI to which a call is referred.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```

ZULONG dwConnId;
ST\_SUA\_ENDP\_URI *pstLocalUri;
ZINT iRet;
.....
/* Get the URI to which a call is referred. */
iRet = Jpf_ConnGetReferToUri(dwConnId, &pstLocalUri);

```

4.5.2.2.18 Jpf_ConnGetReferByUri

Jpf_ConnGetReferByUri gets the URI by which a call is referred.

```

ZINT Jpf_ConnGetReferByUri(ZULONG dwConnId,
ST\_SUA\_ENDP\_URI **ppstReferByUri)

```

[Parameters]

Input parameters:

```

ZULONG dwConnId
ID of the Connection.

```

```

ST\_SUA\_ENDP\_URI **ppstLocalUri
A pointer which will point to the pointer to the URI by which a call is referred.

```

Output parameters:

```

ST\_SUA\_ENDP\_URI **ppstLocalUri
A pointer which points to the pointer to the URI by which a call is referred.

```

[Return value]

```

ZOK: Operation succeeded.
ZFAILED: Operation failed.

```

[Example]

```

ZULONG dwConnId;
ST\_SUA\_ENDP\_URI *pstLocalUri;
ZINT iRet;
.....
/* Get the URI by which a call is referred. */
iRet = Jpf_ConnGetReferByUri(dwConnId, &pstLocalUri);

```

4.5.2.2.19 Jpf_ConnOfferHasVideo

Jpf_ConnOfferHasVideo check if the offer from peer has video parameter in SDP .

```

ZBOOL Jpf_ConnOfferHasVideo(ZULONG dwConnId)

```

[Parameters]

Input parameters:

ZULONG dwConnId
ID of the Connection.

Output parameters:

None.

[Return value]

ZTRUE: There is video parameter in offer SDP.
ZFALSE: There isn't.

[Example]

4.5.3 Useful Interfaces for Data Management

Interfaces in this section can be found in Jpf_db_ui.h.

4.5.3.1 Main Data Interfaces**4.5.3.1.1 Jpf_DbGetLocalIp**

Jpf_DbGetLocalIp gets the local IP address.

```
ZINT Jpf_DbGetLocalIp(ZULONG *pdwIp);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwIp
A pointer which points to the local IP address.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwIp;
.....
/* Get the local IP address. */
Jpf_DbGetLocalIp(&dwIp);
```

4.5.3.1.2 Jpf_DbGetLocalIpv6

Jpf_DbGetLocalIpv6 gets the local IPv6 address which is a character string.

```
ZINT Jpf_DbGetLocalIpv6(ZUCHAR **ppucIpv6)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZUCHAR **ppucIpv6
```

This pointer has the address of the string of the local IPv6 address.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZUCHAR **ppucIpv6;
.....
/* Get the local IPv6 address. */
Jpf_DbGetLocalIpv6(ppucIpv6);
```

4.5.3.1.3 Jpf_DbGetLocalIpList

Jpf_DbGetLocalIpList gets local ip address list.

```
ZINT Jpf_DbGetLocalIpList(ZULONG **ppdwIpList, ZULONG *pdwNum)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG **ppdwIpList
```

Pointer to the array of local ip addresses.

```
ZULONG *pdwNum
```

Pointer to the count of local ip addresses.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG *pdwIpList;
ZULONG dwNum;
ZINT iRet;
.....
/* Get local ip address list. */
iRet = Jpf_DbGetLocalIpList(&pdwIpList, &dwNum);
```

4.5.3.1.4 Jpf_DbGetUseIpv4

Jpf_DbGetUseIpv4 gets a variant which indicates whether the IPv4 address will be used.

```
ZINT Jpf_DbGetUseIpv4(ZBOOL *pbIsUseIpv4)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbIsUseIpv4
```

The pointer which point to the variant indicating whether the IPv4 address will be used.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZBOOL bIsUseIpv4;
.....
/* Get a variant which indicates whether the IPv4 address will be used. */
Jpf_DbGetUseIpv4(&bIsUseIpv4);
```

4.5.3.1.5 Jpf_DbGetLogLevel

Jpf_DbGetLogLevel gets the current level of the log file.

```
ZINT Jpf_DbGetLogLevel(ZULONG *pdwLogLevel)
```

[Parameters]

Input parameters:

```
ZULONG *pdwLogLevel
```

A pointer which will point to the current level of the log file. The valid log levels are:

Output parameters:

```
ZULONG *pdwLogLevel
```

Pointer to the current level of the log file, which should be one of following:

```
ZLOG_LEVEL_NULL
```

```
ZLOG_LEVEL_FATAL
```

```
ZLOG_LEVEL_ERROR
```

```
ZLOG_LEVEL_WARNING
```

```
ZLOG_LEVEL_INFO
```

```
ZLOG_LEVEL_DBG
```

```
ZLOG_LEVEL_ALL
```

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwLogLevel;
```

```
.....
```

```
/* Get the current log level. */
```

```
Jpf_DbGetLogLevel(&dwLogLevel);
```

4.5.3.1.6 Jpf_DbSetLocalIp

Jpf_DbSetLocalIp sets the local IP address.

```
ZINT Jpf_DbSetLocalIp(ZULONG dwIp);
```

[Parameters]

Input parameters:

```
ZULONG dwIp
```

The local IP address.

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwIp;
.....
/* Set the local IP address. */
Jpf_DbSetLocalIp(dwIp);
```

4.5.3.1.7 Jpf_DbSetLocalIpv6

Jpf_DbSetLocalIpv6 sets the local IPv6 address.

```
ZINT Jpf_DbSetLocalIpv6(ZUCHAR *pucIpv6)
```

[Parameters]

Input parameters:

```
ZUCHAR *pucIpv6
The pointer to a character string of a given IPv6 address.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZUCHAR *pucIpv6;
.....
/* Set the local IPv6 address. */
Jpf_DbSetLocalIpv6(*pucIpv6);
```

4.5.3.1.8 Jpf_DbSetUseIpv4

Jpf_DbSetUseIpv4 sets the level of the log file.

```
ZINT Jpf_DbSetUseIpv4(ZBOOL bIsUseIpv4)
```

[Parameters]

Input parameters:

```
ZBOOL bIsUseIpv4
ZTRUE for using IPv4, or using IPv6.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

4.5.3.1.9 Jpf_DbSetLogLevel

Jpf_DbSetLogLevel sets the level of the log file.

```
ZINT Jpf_DbSetLogLevel(ZULONG dwLogLevel);
```

[Parameters]

Input parameters:

```
ZULONG dwLogLeve  
The log level.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwLogLeve;  
.....  
/* Set the log level. */  
Jpf_DbSetLogLevel(dwLogLevel);
```

4.5.3.2 User Data Interfaces

4.5.3.2.1 Jpf_DbGetUsrName

Jpf_DbGetUsrName gets the specific local user name.

```
ZINT Jpf_DbGetUsrName(ZINT iIndex, ZCHAR **ppcName)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

Output parameters:

```
ZCHAR **ppcName  
A pointer which points to the local user name.
```

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZCHAR *pcName;
ZINT iRet;
.....
/* Get the specific local user name. */
iRet = Jpf_DbGetUsrName(iIndex, &pcName);
```

4.5.3.2.2 Jpf_DbGetDispName

Jpf_DbGetDispName gets the specific local user display name.

```
ZINT Jpf_DbGetDispName(ZINT iIndex, ZCHAR **ppcName)
```

[Parameters]

Input parameters:

ZINT iIndex
Local user index.

Output parameters:

ZCHAR **ppcName
A pointer which points to the local user display name.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZCHAR *pcUri;
.....
/* Get the URI of the local user. */
Jpf_DbGetUsrUri(&pcUri);
```

4.5.3.2.3 Jpf_DbGetAuthName

Jpf_DbGetAuthName gets the specific local user authorization name.

```
ZINT Jpf_DbGetAuthName(ZINT iIndex, ZCHAR **ppcName);
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

Output parameters:

```
ZCHAR **ppcName  
A pointer which points to the local user authorization name.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;  
ZCHAR *ppcName;  
ZINT iRet;  
.....  
/* Get the specific local user authorization name. */  
iRet = Jpf_DbGetAuthName(iIndex, &ppcName);
```

4.5.3.2.4 Jpf_DbGetAuthPasswd

Jpf_DbGetAuthPasswd gets the specific local user authorization password.

```
ZINT Jpf_DbGetAuthPasswd(ZINT iIndex, ZCHAR **ppcPasswd)
```

[Parameters]**Input parameters:**

```
ZINT iIndex  
Local user index.
```

Output parameters:

```
ZCHAR ** ppcPasswd  
A pointer which points to the local user authorization password.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```

ZINT iIndex;
ZCHAR *pcPasswd;
ZINT iRet;
.....
/* Get the specific local user authorization password. */
iRet = Jpf_DbGetAuthPasswd(iIndex, &pcPasswd);

```

4.5.3.2.5 Jpf_DbGetRegistrarIp

Jpf_DbGetRegistrarIp gets the IP address of a registrar.

```
ZINT Jpf_DbGetRegistrarIp(ZINT iIndex, ZULONG *pdwIp)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZULONG *pdwIp
A pointer which points to the IP address.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZINT iIndex;
ZULONG dwIp;
ZINT iRet;
.....
/* Get the IP address of a given registrar. */
iRet = Jpf_DbGetRegistrarIp(iIndex, &dwIp);

```

4.5.3.2.6 Jpf_DbGetRegistrarIpv6

Jpf_DbGetRegistrarIpv6 gets the IPv6 address of a registrar.

```
ZINT Jpf_DbGetRegistrarIpv6(ZINT iIndex, ZCHAR **ppucIpv6)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZCHAR **ppucIpv6
A pointer to the pointer which points to the IPv6 address.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;
ZCHAR *pucIpv6;
ZINT iRet;
.....
/* Get the IPv6 address of a given registrar. */
iRet = Jpf_DbGetRegistrarIpv6(iIndex, &pucIpv6);
```

4.5.3.2.7 Jpf_DbGetRegistrarRealm

Jpf_DbGetRegistrarRealm gets the register server realm name.

```
ZINT Jpf_DbGetRegDomain(ZINT iIndex, ZCHAR **ppcRealm)
```

[Parameters]**Input parameters:**

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZCHAR ** ppcRealm
A pointer which points to the handle of the register server realm.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZCHAR *pcDomain;
.....
/* Get the register server realm. */
Jpf_DbGetRegDomain(&pcDomain);
```

4.5.3.2.8 Jpf_DbGetRegistrarTpt

Jpf_DbGetRegistrarTpt gets the register server transport type.

```
ZINT Jpf_DbGetRegistrarTpt(ZINT iIndex, ZULONG *pdwTptType)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZULONG *pdwTptType
Pointer to the transport type of register server, which may be one of following:
    EN_UTAL_TPT_UDP,           /* udp protocol */
    EN_UTAL_TPT_TCP_SERV,     /* tcp server protocol */
    EN_UTAL_TPT_TCP_CLI,      /* tcp client protocol */
    EN_UTAL_TPT_TLS_SERV,     /* tls(over tcp) server protocol */
    EN_UTAL_TPT_TLS_CLI,      /* tls(over tcp) client protocol */
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;
ZULONG dwTptType;
ZINT iRet;
.....
/* Get the register server transport type. */
iRet = Jpf_DbGetRegistrarTpt(iIndex, &dwTptType);
```

4.5.3.2.9 Jpf_DbGetRegistrarUdpPort

Jpf_DbGetRegistrarUdpPort gets the port number of register server for UDP protocol.

```
ZINT Jpf_DbGetRegistrarUdpPort(ZINT iIndex, ZULONG *pdwPort)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZULONG *pdwPort
Points to port number of register server for UDP protocol.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;
ZULONG dwPort;
ZINT iRet;
.....
/* Get the port number of register server for UDP protocol. */
iRet = Jpf_DbGetRegistrarUdpPort(iIndex, &dwPort);
```

4.5.3.2.10 Jpf_DbGetRegistrarTcpPort

Jpf_DbGetRegistrarTcpPort gets the port number of register server for TCP protocol.

```
ZINT Jpf_DbGetRegistrarTcpPort(ZINT iIndex, ZULONG *pdwPort)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZULONG *pdwPort
Points to port number of register server for TCP protocol.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
INT iIndex;
ZULONG dwPort;
ZINT iRet;
.....
/* Get the port number of register server for TCP protocol. */
iRet = Jpf_DbGetRegistrarTcpPort(iIndex, &dwPort);
```

4.5.3.2.11 Jpf_DbGetProxyIp

Jpf_DbGetProxyIp gets the IP address of a proxy.

```
Jpf_DbGetProxyIp(ZINT iIndex, ZULONG *pdwIp)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

Output parameters:

```
ZULONG *pdwIp  
A pointer which points to the proxy IP address.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwIp;  
.....  
/* Get the proxy IP address. */  
Jpf_DbGetProxyIp(&dwIp);
```

4.5.3.2.12 Jpf_DbGetProxyIpv6

Jpf_DbGetProxyIpv6 gets the IPv6 address of a proxy.

```
ZINT Jpf_DbGetProxyIpv6(ZINT iIndex, ZCHAR **ppucIpv6)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

Output parameters:

```
ZCHAR **ppucIpv6  
A pointer to the pointer which points to the IPv6 address.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```

ZINT iIndex;
ZCHAR *pucIpv6;
ZINT iRet;
.....
/* Get the IPv6 address of a given proxy. */
iRet = Jpf_DbGetProxyIpv6 (iIndex, &pucIpv6);

```

4.5.3.2.13 Jpf_DbGetProxyRealm

Jpf_DbGetProxyRealm gets the proxy server realm name.

```
ZINT Jpf_DbGetProxyRealm(ZINT iIndex, ZCHAR **ppcRealm)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZCHAR ** ppcRealm
Pointer to name string of proxy server realm.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZCHAR *pcDomain;
.....
/* Get the register server realm. */
Jpf_DbGetRegDomain(&pcDomain);

```

4.5.3.2.14 Jpf_DbGetProxyTpt

Jpf_DbGetProxyTpt gets the proxy server transport type.

```
ZINT Jpf_DbGetProxyTpt(ZINT iIndex, ZULONG *pdwTptType)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZULONG *pdwTptType
```

Point to the transport type of proxy server. See also Jpf_DbGetRegistrarTpt.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZULONG dwTptType;
ZINT iRet;
.....
/* Get the proxy server transport type. */
iRet = Jpf_DbGetProxyTpt(iIndex, &dwTptType);
```

4.5.3.2.15 Jpf_DbGetProxyUdpPort

Jpf_DbGetProxyUdpPort gets the port number of proxy server for UDP protocol.

```
ZINT Jpf_DbGetProxyUdpPort(ZINT iIndex, ZULONG *pdwPort)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

Output parameters:

```
ZULONG *pdwPort
Points to port number of proxy server for UDP protocol.
```

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZULONG dwPort;
ZINT iRet;
.....
/* Get the port number of proxy server for UDP protocol. */
iRet = Jpf_DbGetProxyUdpPort(iIndex, &dwPort);
```

4.5.3.2.16 Jpf_DbGetProxyTcpPort

Jpf_DbGetProxyTcpPort gets the port number of register server for TCP protocol.

```
ZINT Jpf_DbGetProxyTcpPort(ZINT iIndex, ZULONG *pdwPort)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

Output parameters:

```
ZULONG *pdwPort  
Points to port number of proxy server for TCP protocol.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;  
ZULONG dwPort;  
ZINT iRet;  
.....  
/* Get the port number of register server for TCP protocol. */  
iRet = Jpf_DbGetProxyTcpPort(iIndex, &dwPort);
```

4.5.3.2.17 Jpf_DbSetUsrName

Jpf_DbSetUsrName sets the local user name.

```
ZINT Jpf_DbSetUsrName(ZINT iIndex, ZCHAR *pcName)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.  
  
ZCHAR *pcName  
A pointer to the user name.
```

Output parameters:

```
None.
```

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZCHAR *pcName;
.....
/* Set the local user name. */
Jpf_DbSetUsrName(pcName);
```

4.5.3.2.18 Jpf_DbSetDispName

Jpf_DbSetDispName sets the specific local display name.

```
ZINT Jpf_DbSetDispName(ZINT iIndex, ZCHAR *pcName)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.

ZCHAR *pcName
A pointer to the display name.
```

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZCHAR *pcName;
ZINT iRet;
.....
/* Set the specific local display name. */
iRet = Jpf_DbSetDispName(iIndex, *pcName);
```

4.5.3.2.19 Jpf_DbSetAuthName

Jpf_DbSetAuthName sets the specific local user's authorization name.

```
ZINT Jpf_DbSetAuthName(ZINT iIndex, ZCHAR *pcAccount)
```

[Parameters]

Input parameters:

ZINT iIndex
Local user index.

ZCHAR * pcAccount
A pointer to the authorization name.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZCHAR *pcName;
ZINT iRet;
.....
/* Set the specific local user's authorization name. */
iRet = Jpf_DbSetAuthName(iIndex, pcAccount);
```

4.5.3.2.20 Jpf_DbSetAuthPasswd

Jpf_DbSetAuthPasswd sets the specific local user's authorization password.

```
ZINT Jpf_DbSetDispName(ZINT iIndex, ZCHAR *pcName)
```

[Parameters]

Input parameters:

ZINT iIndex
Local user index.

ZCHAR *pcName
A pointer to the authorization password.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```

ZINT iIndex;
ZCHAR *pcName;
ZINT iRet;
.....
/* Set the specific local user's authorization password. */
iRet = Jpf_DbSetDispName(iIndex, pcName);

```

4.5.3.2.21 Jpf_DbSetRegistrarIp

Jpf_DbSetRegistrarIp sets the IP address of a register server.

```
ZINT Jpf_DbSetRegistrarIp(ZINT iIndex, ZULONG dwIp)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

```
ZULONG dwIp
The IP address.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZINT iIndex;
ZULONG dwIp;
ZINT iRet;
.....
/* Set the proxy IP address. */
iRet = ZINT Jpf_DbSetRegistrarIp(iIndex, dwIp);

```

4.5.3.2.22 Jpf_DbSetRegistrarIpv6

Jpf_DbSetRegistrarIpv6 sets the IPv6 address of a register server.

```
ZINT Jpf_DbSetRegistrarIpv6(ZINT iIndex, ZUCHAR *pucIpv6)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

```
ZUCHAR *pucIpv6  
The pointer to string of the IPv6 address.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;  
ZUCHAR *pucIpv6;  
ZINT iRet;  
.....  
/* Set the IPv6 address of a proxy. */  
iRet = Jpf_DbSetRegistrarIpv6(ZINT iIndex, ZUCHAR *pucIpv6);
```

4.5.3.2.23 Jpf_DbSetRegistrarRealm

Jpf_DbSetRegistrarRealm sets the register realm.

```
ZINT Jpf_DbSetRegistrarRealm(ZINT iIndex, ZCHAR *pcDomain)
```

[Parameters]**Input parameters:**

```
ZINT iIndex  
Local user index.
```

```
ZCHAR *pcDomain  
The register realm name.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZCHAR *pcDomain;
.....
/* Set the register domain name. */
Jpf_DbSetRegDomain(pcDomain);
```

4.5.3.2.24 Jpf_DbSetRegistrarTpt

Jpf_DbSetRegistrarTpt sets the transport type of register server.

```
ZINT Jpf_DbSetRegistrarTpt(ZINT iIndex, ZULONG dwTptType)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.
```

```
ZULONG dwTptType
Indicates the transport type of register server. See also Jpf_DbGetRegistrarTpt.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;
ZULONG dwTptType;
ZINT iRet;
.....
/* Set the transport type of register server. */
iRet = Jpf_DbSetRegistrarTpt(iIndex, dwTptType);
```

4.5.3.2.25 Jpf_DbSetRegistrarUdpPort

Jpf_DbSetRegistrarUdpPort sets the port number of register server for UDP protocol.

```
ZINT Jpf_DbSetRegistrarUdpPort(ZINT iIndex, ZULONG dwPort)
```

[Parameters]

Input parameters:

ZINT iIndex
Local user index.

ZULONG dwPort
Indicates the port number of register server for UDP protocol.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;  
ZULONG dwPort;  
ZINT iRet;  
.....  
/* Set the port number of register server for UDP protocol. */  
iRet = Jpf_DbSetRegistrarUdpPort(iIndex, dwPort);
```

4.5.3.2.26 Jpf_DbSetRegistrarTcpPort

Jpf_DbSetRegistrarTcpPort sets the port number of register server for TCP protocol.

```
ZINT Jpf_DbSetRegistrarTcpPort(ZINT iIndex, ZULONG dwPort)
```

[Parameters]

Input parameters:

ZINT iIndex
Local user index.

ZULONG dwPort
Indicates the port number of register server for TCP protocol.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

4.5.3.2.27 Jpf_DbSetProxyIp

Jpf_DbSetProxyIp sets the IP address of a proxy.

```
ZINT Jpf_DbSetProxyIp(ZINT iIndex, ZULONG dwIp)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

```
ZULONG dwIp  
The proxy IP address.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwIp;  
ZINT iIndex;  
ZINT iRet;  
.....  
/* Set the proxy IP address. */  
iRet = Jpf_DbSetProxyIp(iIndex, dwIp);
```

4.5.3.2.28 Jpf_DbSetProxyIpv6

Jpf_DbSetProxyIpv6 sets the IPv6 address of a proxy.

```
ZINT Jpf_DbSetProxyIpv6(ZINT iIndex, ZUCHAR *pucIpv6)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

```
ZUCHAR *pucIpv6  
The pointer to the IPv6 address.
```

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZUCHAR *pucIpv6;
ZINT iRet;
.....
/* Set the IPv6 address of a proxy. */
iRet = Jpf_DbSetProxyIpv6(iIndex, pucIpv6);
```

4.5.3.2.29 Jpf_DbSetProxyRealm

Jpf_DbSetProxyRealm sets the proxy realm.

```
ZINT Jpf_DbSetProxyRealm(ZINT iIndex, ZCHAR *pcDomain)
```

[Parameters]

Input parameters:

```
ZINT iIndex
Local user index.

ZCHAR *pcDomain
The proxy realm name.
```

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZCHAR *pcDomain;
ZINT iRet;
.....
/* Set the proxy realm. */
iRet = Jpf_DbSetProxyRealm(iIndex, pcDomain);
```

4.5.3.2.30 Jpf_DbSetProxyTpt

Jpf_DbSetProxyTpt sets the transport type of proxy server.

```
ZINT Jpf_DbSetProxyTpt(ZINT iIndex, ZULONG dwTptType)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

```
ZULONG dwTptType  
Indicates the transport type of proxy server. See also Jpf_DbGetRegistrarTpt.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZINT iIndex;  
ZULONG dwTptType;  
ZINT iRet;  
.....  
/* Set the transport type of proxy server. */  
iRet = Jpf_DbSetProxyTpt(iIndex, dwTptType);
```

4.5.3.2.31 Jpf_DbSetProxyUdpPort

Jpf_DbSetProxyUdpPort sets the port number of proxy server for UDP protocol.

```
ZINT Jpf_DbSetProxyUdpPort(ZINT iIndex, ZULONG dwPort)
```

[Parameters]

Input parameters:

```
ZINT iIndex  
Local user index.
```

```
ZULONG dwPort  
The proxy UDP port.
```

Output parameters:

```
None.
```

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZULONG dwPort;
ZINT iRet;
.....
/* Set the port number of proxy server for UDP protocol. */
iRet = Jpf_DbSetProxyUdpPort(iIndex, dwPort);
```

4.5.3.2.32 Jpf_DbSetProxyTcpPort

Jpf_DbSetProxyUdpPort sets the port number of proxy server for TCP protocol.

```
ZINT Jpf_DbSetProxyTcpPort(ZINT iIndex, ZULONG dwPort)
```

[Parameters]

Input parameters:

ZINT iIndex
Local user index.

ZULONG dwPort
The proxy TCP port.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZINT iIndex;
ZULONG dwPort;
ZINT iRet;
.....
/* Set the port number of proxy server for TCP protocol. */
iRet = Jpf_DbSetProxyTcpPort(iIndex, dwPort);
```

4.5.3.3 SIP Data Interfaces

4.5.3.3.1 Jpf_DbGetSipUdpListenPort

Jpf_DbGetSipUdpListenPort gets the SIP listen UDP port.

```
ZINT Jpf_DbGetSipUdpListenPort(ZULONG *pdwPort)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwPort
```

A pointer which points to the SIP listen UDP port.

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwPort;
```

```
.....
```

```
/* Get the SIP listen UDP port. */
```

```
Jpf_DbGetSipUdpListenPort(&dwPort);
```

4.5.3.3.2 Jpf_DbGetSipTcpListenPort

Jpf_DbGetSipTcpListenPort gets the SIP listen TCP port.

```
ZINT Jpf_DbGetSipTcpListenPort(ZULONG *pdwPort)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwPort
```

A pointer which points to the SIP listen TCP port.

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwPort;  
  
.....  
/* Get the SIP listen TCP port. */  
Jpf_DbGetSipTcpListenPort(&dwPort);
```

4.5.3.3 Jpf_DbGetRegEnable

Jpf_DbGetRegEnable gets the register enabling flag.

```
ZINT Jpf_DbGetRegEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable  
Pointer to the register enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Get the register enabling flag. */  
Jpf_DbGetRegEnable(&bEnable);
```

4.5.3.3.4 Jpf_DbGetProxyEnable

Jpf_DbGetProxyEnable gets the proxy enabling flag.

```
ZINT Jpf_DbGetProxyEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable  
Pointer to the proxy enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
ZINT iRet;  
.....  
/* Get the proxy enabling flag. */  
iRet = Jpf_DbGetProxyEnable(&bEnable);
```

4.5.3.3.5 Jpf_DbGetStunEnable

Jpf_DbGetStunEnable gets the STUN enabling flag.

```
ZINT Jpf_DbGetStunEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable  
Pointer to the STUN enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Get the STUN enabling flag. */  
Jpf_DbGetStunEnable(&bEnable);
```

4.5.3.3.6 Jpf_DbGetRtcpEnable

Jpf_DbGetRtcpEnable gets the RTCP enabling flag.

```
ZINT Jpf_DbGetRtcpEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZBOOL *pbEnable
 Pointer to the RTCP enabling flag.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;
.....
/* Get the RTCP enabling flag. */
Jpf_DbGetRtcpEnable(&bEnable);
```

4.5.3.3.7 Jpf_DbGetAacEnable

Jpf_DbGetAacEnable gets the auto-accept-call enabling flag.

```
ZINT Jpf_DbGetAacEnable(ZBOOL *pbEnable)
```

[Parameters]**Input parameters:**

None.

Output parameters:

ZBOOL *pbEnable
 Pointer to the auto-accept-call enabling flag.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;
.....
/* Get the auto-accept-call enabling flag. */
Jpf_DbGetAacEnable(&bEnable);
```

4.5.3.3.8 Jpf_DbGetDndEnable

Jpf_DbGetDndEnable gets the do-not-disturb enabling flag.

```
ZINT Jpf_DbGetDndEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

ZBOOL *pbEnable
It will point to the do-not-disturb enabling flag.

Output parameters:

ZBOOL *pbEnable
Pointer to the do-not-disturb enabling flag.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Get the do-not-disturb enabling flag. */  
Jpf_DbGetDndEnable(&bEnable);
```

4.5.3.3.9 Jpf_DbGetArEnable

Gets audio redial enabling flag.

```
ZINT Jpf_DbGetArEnable(ZBOOL *pbEnable);
```

[Parameters]

Input parameters:

ZBOOL *pbEnable
It will point to the redial enabling flag.

Output parameters:

ZBOOL *pbEnable
Pointer to the redial enabling flag.

[Return value]

Returns ZOK.

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Get audio redial enabling flag. */  
Jpf_DbGetArEnable(&bEnable);
```

4.5.3.3.10 Jpf_DbGetStEnable

Gets the session timer enabling flag.

```
ZINT Jpf_DbGetStEnable(ZBOOL *pbEnable);
```

[Parameters]

Input parameters:

```
ZBOOL *pbEnable  
It will point to the session timer flag.
```

Output parameters:

```
ZBOOL *pbEnable  
Pointer to the redial enabling flag.
```

[Return value]

```
Returns ZOK.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Get the session timer enabling flag. */  
Jpf_DbGetStEnable(&bEnable);
```

4.5.3.3.11 Jpf_DbGetCfuEnable

Jpf_DbGetCfuEnable gets the unconditional call forward enabling flag.

```
ZINT Jpf_DbGetCfuEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

```
None.
```

Output parameters:

```
ZBOOL *pbEnable  
Pointer to the unconditional call forward enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```

ZBOOL bEnable;
ZINT iRet;
.....
/* Get the unconditional call forward enabling flag. */
iRet = Jpf_DbGetCfuEnable(bEnable);

```

4.5.3.3.12 Jpf_DbGetCfbEnable

Jpf_DbGetCfbEnable gets call forward on busy enabling flag.

```
ZINT Jpf_DbGetCfbEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
Pointer to the call forward on busy enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZBOOL bEnable;
ZINT iRet;
.....
/* Get call forward on busy enabling flag. */
iRet = Jpf_DbGetCfbEnable(&bEnable);

```

4.5.3.3.13 Jpf_DbGetCfnaEnable

Jpf_DbGetCfnaEnable gets call forward on no answer enabling flag.

```
ZINT Jpf_DbGetCfnaEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
Pointer to the call forward on no answer enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;
ZINT iRet;
.....
/* Get call forward on no answer enabling flag. */
iRet = Jpf_DbGetCfnaEnable(bEnable);
```

4.5.3.3.14 Jpf_DbGetCfUri

Jpf_DbGetCfUri gets the call forward unconditional URI.

```
ZINT Jpf_DbGetCfUri(ZCHAR **ppcUri)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZCHAR **ppcUri
Pointer to the call forward URI.
```

[Return value]

```
ZOK: Operation succeeded.
```

[Example]

```
ZCHAR *pcUri;
ZINT iRet;
.....
/* Get the call forward uri. */
iRet = Jpf_DbGetCfUri(&pcUri);
```

4.5.3.3.15 Jpf_DbGetCfbUri

Gets call forward on busy URI.

```
ZINT Jpf_DbGetCfbUri(ZCHAR **ppcUri);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZCHAR **ppcUri
 Pointer to the URI.

[Return value]

ZOK: Operation succeeded.

[Example]

```
ZCHAR *pcUri;
.....
/* Get the call forward on busy URI. */
Jpf_DbGetCfbUri(ZCHAR &pcUri);
```

4.5.3.3.16 Jpf_DbGetCfnaUri

Gets call forward on no-answer URI.

```
ZINT Jpf_DbGetCfnaUri(ZCHAR **ppcUri);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZCHAR **ppcUri
 Pointer to the URI.

[Return value]

ZOK: Operation succeeded.

[Example]

```
ZCHAR *pcUri;
.....
Jpf_DbGetCfnaUri(&pcUri);
```

4.5.3.3.17 Jpf_DbGetUsedCompactHdr

Jpf_DbGetUsedCompactHdr gets call forward on no answer enabling flag.

```
ZINT Jpf_DbGetUsedCompactHdr(ZBOOL *pbIsUsed)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZBOOL *pbIsUsed
 Pointer to the using flag of compact header.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZBOOL bIsUsed;
ZINT iRet;
.....
/* Get call forward on no answer enabling flag. */
iRet = Jpf_DbGetUsedCompactHdr(&bIsUsed);
```

4.5.3.3.18 Jpf_DbGetSipTmr1

Jpf_DbGetSipTmr1 gets setting of SIP timer 1 length.

```
ZINT Jpf_DbGetSipTmr1(ZULONG *pdwTimer)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwTimer
 Pointer to length of SIP timer 1 in microsecond.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZULONG dwTimer;
ZINT iRet;
.....
/* Get setting of SIP timer 1 length. */
iRet = Jpf_DbGetSipTmr1(&dwTimer);
```

4.5.3.3.19 Jpf_DbGetSipTmr2

Jpf_DbGetSipTmr2 gets setting of SIP timer 2 length.

```
ZINT Jpf_DbGetSipTmr2(ZULONG *pdwTimer)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTimer
```

Pointer to length of SIP timer 2 in microsecond.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG dwTimer;  
ZINT iRet;  
.....  
/* Get setting of SIP timer 2 length. */  
iRet = Jpf_DbGetSipTmr2(&dwTimer);
```

4.5.3.3.20 Jpf_DbGetSipTmr4

Jpf_DbGetSipTmr4 gets setting of SIP timer 4 length.

```
ZINT Jpf_DbGetSipTmr4(ZULONG *pdwTimer)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTimer
```

Pointer to length of SIP timer 4 in microsecond.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```

ZULONG dwTimer;
ZINT iRet;
.....
/* Get setting of SIP timer 4 length. */
iRet = Jpf_DbGetSipTmr4(&dwTimer);

```

4.5.3.3.21 Jpf_DbGetSipTmrC

Jpf_DbGetSipTmrC gets setting of SIP timer C length.

```
ZINT Jpf_DbGetSipTmrC(ZULONG *pdwTimer)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTimer
Pointer to length of SIP timer C in microsecond.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwTimer;
ZINT iRet;
.....
/* Get setting of SIP timer C length. */
iRet = Jpf_DbGetSipTmrC(&dwTimer);

```

4.5.3.3.22 Jpf_DbGetSipTmrD

Jpf_DbGetSipTmrD gets setting of SIP timer D length.

```
ZINT Jpf_DbGetSipTmrD(ZULONG *pdwTimer)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTimer  
Pointer to the length of SIP timer D in microsecond.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwTimer;  
ZINT iRet;  
.....  
/* Get setting of SIP timer D length. */  
iRet = Jpf_DbGetSipTmrD (&dwTimer);
```

4.5.3.3.23 Jpf_DbGetTmrSess

Gets the session timer length.

```
ZINT Jpf_DbGetTmrSess(ZULONG *pdwTimer);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTimer  
Pointer to the session timer length.
```

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwTimer;  
.....  
/* Get the session timer length. */  
Jpf_DbGetTmrSess(&dwTimer);
```

4.5.3.3.24 Jpf_DbGetHoldType

Jpf_DbGetHoldType gets type of call on hold.

```
ZINT Jpf_DbGetHoldType(ZULONG *pdwType)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwType
 Pointer to the type of call on hold.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZULONG dwType;
ZINT iRet;
.....
/* Get type of call on hold.*/
iRet = Jpf_DbGetHoldType(&dwType);
```

4.5.3.3.25 Jpf_DbGetUserAgent

Jpf_DbGetUserAgent gets string of user agent.

```
ZINT Jpf_DbGetUserAgent(ZCHAR **ppcUserAgent)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZCHAR ** ppcUserAgent
 Pointer to string of user agent.

[Return value]

ZOK: Operation succeeded.
 ZFAILED: Operation failed.

[Example]

```
ZCHAR *pcUserAgent;
ZINT iRet;
.....
/* Get string of user agent.*/
iRet = Jpf_DbGetUserAgent(&pcUserAgent);
```

4.5.3.3.26 Jpf_DbSetSipUdpListenPort

Jpf_DbSetSipUdpListenPort sets the SIP listen UDP port.

```
ZINT Jpf_DbSetSipUdpListenPort(ZULONG dwPort)
```

[Parameters]

Input parameters:

```
ZULONG dwPort  
The SIP listen UDP port.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwPort  
.....  
/* Set the SIP listen UDP port. */  
Jpf_DbSetSipUdpListenPort(dwPort);
```

4.5.3.3.27 Jpf_DbSetSipTcpListenPort

Jpf_DbSetSipTcpListenPort sets the SIP listen TCP port.

```
ZINT Jpf_DbSetSipTcpListenPort(ZULONG dwPort)
```

[Parameters]

Input parameters:

```
ZULONG dwPort  
The SIP listen TCP port.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwPort;  
  
.....  
/* Set the SIP listen TCP port. */  
Jpf_DbSetSipTcpListenPort(dwPort);
```

4.5.3.3.28 Jpf_DbSetRegEnable

Jpf_DbSetRegEnable sets the register enabling flag.

```
ZINT Jpf_DbSetRegEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The register enabling flag.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Set the register enabling flag. */  
Jpf_DbSetRegEnable(bEnable);
```

4.5.3.3.29 Jpf_DbSetProxyEnable

Jpf_DbSetProxyEnable sets the proxy enabling flag.

```
ZINT Jpf_DbSetProxyEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The proxy enabling flag.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
ZINT iRet;  
.....  
/* Set the proxy enabling flag.*/  
iRet = Jpf_DbSetProxyEnable(bEnable);
```

4.5.3.3.30 Jpf_DbSetStunEnable

Jpf_DbSetStunEnable sets the STUN enabling flag.

```
ZINT Jpf_DbSetStunEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The STUN enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the STUN enabling flag. */  
Jpf_DbSetStunEnable(bEnable);
```

4.5.3.3.31 Jpf_DbSetRtcpEnable

Jpf_DbSetRtcpEnable sets the RTCP enabling flag.

```
ZINT Jpf_DbSetRtcpEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

ZBOOL bEnable
The RTCP enabling flag.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the RTCP enabling flag. */  
Jpf_DbSetRtcpEnable(bEnable);
```

4.5.3.3.32 Jpf_DbSetAacEnable

Jpf_DbSetAacEnable sets the auto-accept-call enabling flag.

```
ZINT Jpf_DbSetAacEnable(ZBOOL bEnable)
```

[Parameters]**Input parameters:**

ZBOOL bEnable
The auto-accept-call enabling flag.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the auto-accept-call enabling flag. */  
Jpf_DbSetAacEnable(bEnable);
```

4.5.3.3.33 Jpf_DbSetDndEnable

Jpf_DbSetDndEnable sets the do-not-disturb enabling flag.

```
ZINT Jpf_DbSetDndEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The do-not-disturb enabling flag value.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the do-not-disturb enabling flag. */  
Jpf_DbSetDndEnable(bEnable);
```

4.5.3.3.34 Jpf_DbSetArEnable

Sets the audio redial enabling flag.

```
ZINT Jpf_DbSetArEnable(ZBOOL bEnable);
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The flag value.
```

Output parameters:

```
None.
```

[Return value]

```
Returns ZOK.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the audio enabling flag. */  
Jpf_DbSetArEnable(bEnable);
```

4.5.3.3.35 Jpf_DbSetStEnable

Sets the session timer enabling flag.

```
ZINT Jpf_DbSetStEnable(ZBOOL bEnable);
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The flag value.
```

Output parameters:

```
None.
```

[Return value]

```
Returns ZOK.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the session timer enabling flag. */  
Jpf_DbSetStEnable(bEnable);
```

4.5.3.3.36 Jpf_DbSetCfuEnable

Jpf_DbSetCfuEnable sets the unconditional call forward enabling flag.

```
ZINT Jpf_DbSetCfuEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The unconditional call forward enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
ZINT iRet;  
.....  
/* Set the unconditional call forward enabling flag. */  
iRet = Jpf_DbSetCfuEnable(bEnable);
```

4.5.3.3.37 Jpf_DbSetCfbEnable

Jpf_DbSetCfbEnable sets the call forward on busy enabling flag.

```
ZINT Jpf_DbSetCfbEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The call forward on busy enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
ZINT iRet;  
.....  
/* Set the call forward on busy enabling flag. */  
iRet = Jpf_DbSetCfbEnable(bEnable);
```

4.5.3.3.38 Jpf_DbSetCfnaEnable

Jpf_DbSetCfnaEnable sets the call forward on no answer enabling flag.

```
ZINT Jpf_DbSetCfnaEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The call forward on no answer enabling flag.
```

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;  
ZINT iRet;  
.....  
/* Set the call forward on no answer enabling flag. */  
iRet = Jpf_DbSetCfnaEnable(bEnable);
```

4.5.3.3.39 Jpf_DbSetCfuUri

Sets the call forward unconditional URI.

```
ZINT Jpf_DbSetCfuUri(ZCHAR *pcUri);
```

[Parameters]

Input parameters:

ZCHAR *pcUri
The call forward unconditional URI.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZCHAR *pcUri;  
ZINT iRet;  
.....  
/* Set the call forward unconditional URI. */  
iRet = Jpf_DbSetCfuUri(pcUri);
```

4.5.3.3.40 Jpf_DbSetCfbUri

Sets the call forward on busy URI.

```
ZINT Jpf_DbSetCfbUri(ZCHAR *pcUri);
```

[Parameters]

Input parameters:

ZCHAR *pcUri
The call forward on busy URI.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZCHAR *pcUri;  
ZINT iRet;  
.....  
/* Set the call forward on busy URI. */  
iRet = Jpf_DbSetCfbUri(pcUri);
```

4.5.3.3.41 Jpf_DbSetCfnaUri

Sets the call forward on no-answer URI.

```
ZINT Jpf_DbSetCfnaUri(ZCHAR *pcUri);
```

[Parameters]

Input parameters:

ZCHAR *pcUri
The call forward on no-answer URI.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZCHAR *pcUri;  
ZINT iRet;  
.....  
/* Set the call forward on no-answer URI. */  
iRet = Jpf_DbSetCfnaUri(pcUri);
```

4.5.3.3.42 Jpf_DbSetUsedCompactHdr

Jpf_DbSetUsedCompactHdr sets using flag SIP compact header.

```
ZINT Jpf_DbSetUsedCompactHdr(ZBOOL bIsUsed)
```

[Parameters]

Input parameters:

```
ZBOOL bIsUsed
The using flag for SIP compact header.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bIsUsed;
ZINT iRet;
.....
/* Set using flag SIP compact header.*/
iRet = Jpf_DbSetUsedCompactHdr(bIsUsed);
```

4.5.3.3.43 Jpf_DbSetSipTmr1

Jpf_DbSetSipTmr1 sets length of timer 1.

```
ZINT Jpf_DbSetSipTmr1(ZULONG dwTimer)
```

[Parameters]

Input parameters:

```
ZULONG dwTimer
The length of timer 1 in microsecond.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwTimer;  
ZINT iRet;  
.....  
/* Set length of timer 1. */  
iRet = Jpf_DbSetSipTmr1(dwTimer);
```

4.5.3.3.44 Jpf_DbSetSipTmr2

Jpf_DbSetSipTmr2 sets length of timer 2.

```
ZINT Jpf_DbSetSipTmr2(ZULONG dwTimer)
```

[Parameters]

Input parameters:

```
ZULONG dwTimer  
The length of timer 2 in microsecond.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwTimer;  
ZINT iRet;  
.....  
/* Set length of timer 2. */  
iRet = Jpf_DbSetSipTmr2(dwTimer);
```

4.5.3.3.45 Jpf_DbSetSipTmr4

Jpf_DbSetSipTmr4 sets length of timer 4.

```
ZINT Jpf_DbSetSipTmr4(ZULONG dwTimer)
```

[Parameters]

Input parameters:

```
ZULONG dwTimer  
The length of timer 4 in microsecond.
```

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwTimer;
ZINT iRet;
.....
/* Set length of timer 4. */
iRet = Jpf_DbSetSipTmr4(dwTimer);
```

4.5.3.3.46 Jpf_DbSetSipTmrC

Jpf_DbSetSipTmrC sets length of timer C.

```
ZINT Jpf_DbSetSipTmrC(ZULONG dwTimer)
```

[Parameters]

Input parameters:

ZULONG dwTimer
The length of timer C in microsecond.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwTimer;
ZINT iRet;
.....
/* Set length of timer C. */
iRet = Jpf_DbSetSipTmrC(dwTimer);
```

4.5.3.3.47 Jpf_DbSetSipTmrD

Jpf_DbSetSipTmrD sets length of timer D.

```
ZINT Jpf_DbSetSipTmrD(ZULONG dwTimer)
```

[Parameters]

Input parameters:

ZULONG dwTimer
The length of timer D in microsecond.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwTimer;
ZINT iRet;
.....
/* Set length of timer D. */
iRet = Jpf_DbSetSipTmrD(dwTimer);
```

4.5.3.3.48 Jpf_DbSetTmrSess

Sets the session timer length.

```
ZINT Jpf_DbSetTmrSess(ZULONG dwTimer);
```

[Parameters]**Input parameters:**

ZULONG dwTimer
Length of the timer.

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwTimer;
.....
/* Set the session timer length. */
Jpf_DbSetTmrSess(dwTimer);
```

4.5.3.3.49 Jpf_DbSetHoldType

Jpf_DbSetHoldType sets type of call on hold.

```
ZINT Jpf_DbSetHoldType(ZULONG dwType)
```

[Parameters]

Input parameters:

```
ZULONG dwType  
The type of call on hold.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwType;  
ZINT iRet;  
.....  
/* Set type of call on hold.*/  
iRet = Jpf_DbSetHoldType(dwType);
```

4.5.3.3.50 Jpf_DbSetUserAgent

Jpf_DbSetUserAgent sets string of user agent.

```
ZINT Jpf_DbSetUserAgent(ZCHAR *pcUserAgent)
```

[Parameters]

Input parameters:

```
ZCHAR *pcUserAgent  
The string of user agent.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```

ZCHAR *pcUserAgent;
ZINT iRet;
.....
/* Set string of user agent.*/
iRet = Jpf_DbSetUserAgent(pcUserAgent);

```

4.5.3.4 Media Data Interfaces

4.5.3.4.1 Jpf_DbGetCodecGrp

Jpf_DbGetCodecGrp obtains the value of a codec string which represents the name of a codec, and store the value in an element of an array.

```
ZINT Jpf_DbGetCodecGrp(ZCHAR *pcCodecs, ZUCHAR aucGrp[], ZINT *piGrpSize)
```

[Parameters]

Input parameters:

```
ZCHAR *pcCodecs
```

The codec string.

```
ZUCHAR aucGrp[]
```

The array where the values will be stored.

```
ZINT *piGrpSize
```

Pointer to the max size of the array.

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZCHAR *pcCodecs;
ZUCHAR aucGrp[1024];
ZINT iGrpSize;
.....
/* Store a codec string in an array. */
Jpf_DbGetCodecGrp(pcCodecs, aucGrp, &iGrpSize);

```

4.5.3.4.2 Jpf_DbGetUsedAudioCodecCnt

Jpf_DbGetUsedAudioCodecCnt gets the count of used audio codecs.

```
ZINT Jpf_DbGetUsedAudioCodecCnt(ZULONG *pdwCodecCnt)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwCodecUsedCnt
Pointer to the used codec count.
```

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwCodecUsedCnt;
.....
/* Get the count of used audio codecs.*/
Jpf_DbGetUsedAudioCodecCnt(&dwCodecCnt);
```

4.5.3.4.3 Jpf_DbGetUsedAudioCodec

Jpf_DbGetUsedAudioCodec gets the specific audio codec.

```
ZINT Jpf_DbGetUsedAudioCodec(ZULONG dwIdx, ZCHAR **ppcCodecUsed);
```

[Parameters]

Input parameters:

```
ZULONG dwIdx
Specific codec index.
```

Output parameters:

```
ZCHAR **ppcCodecUsed
Pointer to the used codec.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwIdx;
ZCHAR *pcCodecUsed;
ZINT iRet;
.....
/* Get the specific audio codec. */
iRet = Jpf_DbGetUsedAudioCodec(dwIdx, &pcCodecUsed);

```

4.5.3.4.4 Jpf_DbGetSuptAudioCodecCnt

Jpf_DbGetSuptAudioCodecCnt gets the count of supported audio codecs.

```
ZINT Jpf_DbGetSuptAudioCodecCnt(ZULONG *pdwCodecCnt)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwCodecCnt
Pointer to the support codec count.
```

[Return value]

Returns ZOK.

[Example]

```

ZULONG dwCodecCnt;
.....
/* Get the count of supported audio codecs.*/
Jpf_DbGetSuptAudioCodecCnt(&dwCodecCnt);

```

4.5.3.4.5 Jpf_DbGetSuptAudioCodec

Jpf_DbGetSuptAudioCodec gets the specific audio codec.

```
ZINT Jpf_DbGetSuptAudioCodec(ZULONG dwIdx, ZCHAR **ppcCodecSupt);
```

[Parameters]

Input parameters:

```
ZULONG dwIdx
Specific codec index.
```

Output parameters:

```
ZCHAR **ppcCodecSupt
Pointer to the support codec.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwIdx;
ZCHAR *pcCodecSupt;
ZINT iRet;
.....
/* Get the specific audio codec. */
iRet = Jpf_DbGetSuptAudioCodec(dwIdx, &pcCodecSupt);
```

4.5.3.4.6 Jpf_DbGetVideoCodecCnt

Gets the number of used video codecs.

```
ZINT Jpf_DbGetVideoCodecCnt(ZULONG *pdwCodecCnt)
```

[Parameters]

Input parameters:

```
ZULONG *pdwCodecCnt
This will point to the number of used video codecs.
```

Output parameters:

```
ZULONG *pdwCodecCnt
The number of the used video codecs.
```

[Return value]

```
Returns ZOK.
```

[Example]

```
ZULONG dwCodecCnt;
.....
Jpf_DbGetVideoCodecCnt(&dwCodecCnt);
```

4.5.3.4.7 Jpf_DbGetVideoCodec

Gets a specific video codec by the index.

```
ZINT Jpf_DbGetVideoCodec(ZULONG dwIdx, ST\_JPF\_DB\_VCODEC **ppstCodec);
```

[Parameters]

Input parameters:

ZULONG dwIdx

Index of the video codec in the group of video codecs.

Output parameters:

[ST_JPF_DB_VCODEC](#) **ppstCodec

The video codec.

[Return value]

Returns ZOK.

4.5.3.4.8 Jpf_DbGetVideoCodecX

Gets a specific video codec by the encoding type.

```
ZINT Jpf_DbGetVideoCodecX(ZUCHAR ucEncoding, ST\_JPF\_DB\_VCODEC **ppstCodec);
```

[Parameters]

Input parameters:

ZUCHAR ucEncoding

The encoding type.

Output parameters:

[ST_JPF_DB_VCODEC](#) **ppstCodec

The video codec.

[Return value]

Returns ZOK.

4.5.3.4.9 Jpf_DbGetDtmfType

Jpf_DbGetDtmfType gets the specific audio codec.

```
ZINT Jpf_DbGetDtmfType(ZULONG *pdwDtmfType)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwDtmfType
Pointer to the DTMF type, which should be one of following:
    EN_JPF_DB_DTMF_INBAND,          /* inband dtmf */
    EN_JPF_DB_DTMF_OUTBAND,        /* rfc2833 */
    EN_JPF_DB_DTMF_INFO            /* info */
```

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwDtmfType;
Jpf_DbGetDtmfType(&dwDtmfType);
```

4.5.3.4.10 Jpf_DbGetAecEnable

Jpf_DbGetAecEnable gets the AEC enabling flag.

```
ZINT Jpf_DbGetAecEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
A pointer to the AEC enabling flag.
```

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;
.....
/* Get the AEC enabling flag. */
Jpf_DbGetAecEnable(&bEnable);
```

4.5.3.4.11 Jpf_DbGetAnrEnable

Jpf_DbGetAnrEnable gets the audio noise reduction enabling flag.

```
ZINT Jpf_DbGetAnrEnable(ZBOOL *pbEnable);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
```

A pointer to the audio noise reduction enabling flag.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;
```

```
.....
```

```
/* Get the audio noise reduction enabling flag. */
```

```
Jpf_DbGetAnrEnable(&bEnable);
```

4.5.3.4.12 Jpf_DbGetAgcEnable

Jpf_DbGetAgcEnable gets the AGC enabling flag.

```
ZINT Jpf_DbGetAgcEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
```

A pointer to the AGC enabling flag.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;
```

```
.....
```

```
/* Get the AGC enabling flag. */
```

```
ZINT Jpf_DbGetAgcEnable(&bEnable);
```

4.5.3.4.13 Jpf_DbGetAsdEnable

Jpf_DbGetAsdEnable gets the ASD enabling flag.

```
ZINT Jpf_DbGetAsdEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
```

A pointer to the ASD enabling flag.

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;
```

```
.....
```

```
/* Get the ASD enabling flag. */
```

```
Jpf_DbGetAsdEnable(&bEnable);
```

4.5.3.4.14 Jpf_DbGetDisplayRect

Gets the video display rectangle.

```
ZINT Jpf_DbGetDisplayRect(ZULONG *pdwTop, ZULONG *pdwLeft,  
                          ZULONG *pdwRight, ZULONG *pdwBottom);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwTop

The topmost location of the rectangle.

ZULONG *pdwLeft

The leftmost location of the rectangle.

ZULONG *pdwRight

The the rightmost location of the rectangle.

ZULONG *pdwBottom

The bottom location of the rectangle.

[Return value]

Returns ZOK.

4.5.3.4.15 Jpf_DbGetPreviewRect

Gets the video preview rectangle.

```
ZINT Jpf_DbGetPreviewRect(ZULONG *pdwTop, ZULONG *pdwLeft,  
                          ZULONG *pdwRight, ZULONG *pdwBottom);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwTop

The topmost location of the rectangle.

ZULONG *pdwLeft

The leftmost location of the rectangle.

ZULONG *pdwRight

The the rightmost location of the rectangle.

ZULONG *pdwBottom

The bottom location of the rectangle.

[Return value]

Returns ZOK.

4.5.3.4.16 Jpf_DbGetLiveToneEnable

Jpf_DbGetLiveToneEnable gets the live tone enabling flag.

```
ZINT Jpf_DbGetLiveToneEnable(ZBOOL *pbEnable)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZBOOL *pbEnable
```

A pointer to the live tone enabling flag.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZBOOL bEnable;  
.....  
/* Get the live tone enabling flag. */  
Jpf_DbGetLiveToneEnable(&bEnable);
```

4.5.3.4.17 Jpf_DbSetUsedAudioCodecCnt

Jpf_DbSetUsedAudioCodecCnt sets the number of used audio codecs.

```
ZINT Jpf_DbSetUsedAudioCodecCnt(ZULONG dwCodecCnt)
```

[Parameters]

Input parameters:

```
ZULONG dwCodecCnt
```

The number of used audio codec.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG dwCodecCnt;
ZINT iRet;
.....
/* Set the number of used audio codecs.*/
iRet = Jpf_DbSetUsedAudioCodecCnt(dwCodecCnt);
```

4.5.3.4.18 Jpf_DbSetUsedAudioCodec

Jpf_DbSetUsedAudioCodec sets the specific used audio codec.

```
ZINT Jpf_DbSetUsedAudioCodec(ZULONG dwIdx, ZCHAR *pcCodec);
```

[Parameters]

Input parameters:

```
ZCHAR *pcCodec
The name of used audio codec.

ZULONG dwIdx
Audio codec index.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```
ZCHAR *pcCodec;
ZULONG dwIdx;
ZINT iRet;
.....
/* Set the specific used audio codec.*/
iRet = Jpf_DbSetUsedAudioCodec(dwIdx, pcCodec);
```

4.5.3.4.19 Jpf_DbSetSuptAudioCodecCnt

Jpf_DbSetSuptAudioCodecCnt sets the number of support audio codecs.

```
ZINT Jpf_DbSetSuptAudioCodecCnt(ZULONG dwCodecCnt)
```

[Parameters]

Input parameters:

```
ZULONG dwCodecCnt
```

The number of support audio codec.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG dwCodecCnt;
```

```
ZINT iRet;
```

```
.....
```

```
/* Set the number of support audio codecs.*/
```

```
iRet = Jpf_DbSetSuptAudioCodecCnt(dwCodecCnt);
```

4.5.3.4.20 Jpf_DbSetSuptAudioCodec

Jpf_DbSetSuptAudioCodec sets the specific support audio codec.

```
ZINT Jpf_DbSetSuptAudioCodec(ZULONG dwIdx, ZCHAR *pcCodec);
```

[Parameters]**Input parameters:**

```
ZCHAR *pcCodec
```

The name of support audio codec.

```
ZULONG dwIdx
```

Audio codec index.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```

ZCHAR *pcCodec;
ULONG dwIdx;
ZINT iRet;
.....
/* Set the specific support audio codec.*/
iRet = Jpf_DbSetSuptAudioCodec(dwIdx, pcCodec);

```

4.5.3.4.21 Jpf_DbSetVideoCodecCnt

Sets the number of used video codecs.

```
ZINT Jpf_DbSetVideoCodecCnt(ZULONG dwCodecCnt);
```

[Parameters]

Input parameters:

```
ZULONG dwCodecCnt
```

The number of the used video codecs.

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
ZFAILED: Operation failed.
```

[Example]

```

ZULONG dwCodecCnt;
ZINT iRet;
.....
/* Set the number of used video codecs. */
iRet = Jpf_DbSetVideoCodecCnt(dwCodecCnt);

```

4.5.3.4.22 Jpf_DbSetVideoCodec

Sets the video codec by the index.

```
ZINT Jpf_DbSetVideoCodec(ZULONG dwIdx, ST_JPF_DB_VCODEC *pstCodec);
```

[Parameters]

Input parameters:

ZULONG dwIdx
The codec index.

[ST_JPF_DB_VCODEC](#) *pstCodec
The video codec.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

4.5.3.4.23 Jpf_DbSetVideoCodec

Sets the used video codec.

```
ZINT Jpf_DbSetSuptVideoCodec(ZULONG dwIdx,
                             ST\_JPF\_DB\_VCODEC *pstCodec);
```

[Parameters]

Input parameters:

ZULONG dwIdx
The codec index.

[ST_JPF_DB_VCODEC](#) *pstCodec
The used video codec.

Output parameters:

None.

[Return value]

ZOK: Operation succeeded.
ZFAILED: Operation failed.

[Example]

```
ZULONG dwIdx;
ST\_JPF\_DB\_VCODEC *pstCodec;
ZINT iRet;
.....
/* Set the supported video codec. */
iRet = Jpf_DbSetSuptVideoCodec(dwIdx, pstCodec);
```

4.5.3.4.24 Jpf_DbSetDtmfType

Jpf_DbSetDtmfType sets the type of DTMF data.

```
ZINT Jpf_DbSetDtmfType(ZULONG dwDtmfType)
```

[Parameters]

Input parameters:

```
ZULONG dwDtmfType  
The type of DTMF data.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwDtmfType;  
ZINT iRet;  
.....  
/* Set the type of DTMF data.*/  
iRet = Jpf_DbSetDtmfType(dwDtmfType);
```

4.5.3.4.25 Jpf_DbSetAecEnable

Jpf_DbSetAecEnable sets the AEC enabling flag.

```
ZINT Jpf_DbSetAecEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The AEC enabling flag.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the AEC enabling flag. */  
Jpf_DbSetAecEnable(bEnable);
```

4.5.3.4.26 Jpf_DbSetAnrEnable

Jpf_DbSetAnrEnable sets the audio noise reduction enabling flag.

```
ZINT Jpf_DbSetAnrEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The audio noise reduction enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
.....  
/* Set the audio noise reduction enabling flag. */  
Jpf_DbSetAnrEnable(bEnable);
```

4.5.3.4.27 Jpf_DbSetAgcEnable

Jpf_DbSetAgcEnable sets the AGC enabling flag.

```
ZINT Jpf_DbSetAgcEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The AGC enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Set the a AGC enabling flag. */  
Jpf_DbSetAgcEnable(bEnable);
```

4.5.3.4.28 Jpf_DbSetAsdEnable

Jpf_DbSetAsdEnable sets the ASD enabling flag.

```
ZINT Jpf_DbSetAsdEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The ASD enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Set the ASD enabling flag. */  
Jpf_DbSetAsdEnable(bEnable);
```

4.5.3.4.29 Jpf_DbSetDisplayRect

Sets the video display rectangle.

```
ZINT Jpf_DbSetDisplayRect(ZULONG dwTop, ZULONG dwLeft,  
                          ZULONG dwRight, ZULONG dwBottom);
```

[Parameters]

Input parameters:

ZULONG dwTop

The topmost location of the rectangle.

ZULONG dwLeft

The leftmost location of the rectangle.

ZULONG dwRight

The rightmost location of the rectangle.

ZULONG dwBottom

The bottom location of the rectangle.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.3.4.30 Jpf_DbSetPreviewRect

Sets the video preview rectangle.

```
ZINT Jpf_DbSetPreviewRect(ZULONG dwTop, ZULONG dwLeft,  
                          ZULONG dwRight, ZULONG dwBottom);
```

[Parameters]

Input parameters:

ZULONG dwTop

The topmost location of the rectangle.

ZULONG dwLeft

The leftmost location of the rectangle.

ZULONG dwRight

The rightmost location of the rectangle.

ZULONG dwBottom

The bottom location of the rectangle.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.3.4.31 Jpf_DbSetLiveToneEnable

Jpf_DbSetLiveToneEnable sets the live tone enabling flag.

```
ZINT Jpf_DbSetLiveToneEnable(ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZBOOL bEnable  
The live tone enabling flag.
```

Output parameters:

```
None.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZBOOL bEnable;  
  
.....  
/* Set the live tone enabling flag. */  
Jpf_DbSetLiveToneEnable(bEnable);
```

4.5.3.5 RTP Data Interfaces

4.5.3.5.1 Jpf_DbGetAudioPort

Gets the maximum and minimum audio port numbers.

```
ZINT Jpf_DbGetAudioPort(ZULONG *pdwMinPort, ZULONG *pdwMaxPort)
```

[Parameters]

Input parameters:

```
None.
```

Output parameters:

```
ZULONG *pdwMinPort
Pointer to the minimum audio port number.
```

```
ZULONG *pdwMaxPort
Pointer to the maximum audio port number.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwMinPort;
ZULONG dwMaxPort;
ZINT iRet;
.....
/* Get the maximum and minimum port numbers. */
iRet = Jpf_DbGetAudioPort(&dwMinPort, &dwMaxPort);
```

4.5.3.5.2 Jpf_DbGetVideoPort

Gets the maximum and minimum video port numbers.

```
ZINT Jpf_DbGetVideoPort(ZULONG *pdwMinPort, ZULONG *pdwMaxPort);
```

[Parameters]**Input parameters:**

None.

Output parameters:

```
ZULONG *pdwMinPort
Pointer to the minimum video port number.
```

```
ZULONG *pdwMaxPort
Pointer to the maximum video port number.
```

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwMinPort;
ZULONG dwMaxPort;
ZINT iRet;
.....
/* Get the maximum and minimum video port numbers. */
iRet = Jpf_DbGetVideoPort(&dwMinPort, &dwMaxPort);
```

4.5.3.5.3 ZINT Jpf_DbSetAudioPort

Sets the maximum and minimum audio port numbers.

```
ZINT Jpf_DbSetAudioPort(ZULONG dwMinPort, ZULONG dwMaxPort);
```

[Parameters]

Input parameters:

```
ZULONG dwMinPort  
The minimum audio port number.
```

```
ZULONG dwMaxPort  
The maximum audio port number.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwMinPort;  
ZULONG dwMaxPort;  
ZINT iRet;  
.....  
/* Set the maximum and minimum audio port number. */  
iRet = Jpf_DbSetAudioPort(dwMinPort, dwMaxPort);
```

4.5.3.5.4 ZINT Jpf_DbSetVideoPort

Sets the maximum and minimum video port numbers.

```
ZINT Jpf_DbSetVideoPort(ZULONG dwMinPort, ZULONG dwMaxPort)
```

[Parameters]

Input parameters:

```
ZULONG dwMinPort  
The minimum video port number.
```

```
ZULONG dwMaxPort  
The maximum video port number.
```

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

[Example]

```
ZULONG dwMinPort;
ZULONG dwMaxPort;
ZINT iRet;
.....
/* Set the maximum and minimum video port number. */
Jpf_DbSetVideoPort(dwMinPort, dwMaxPort);
```

4.5.3.6 STUN Data Interfaces

4.5.3.6.1 Jpf_DbGetStunServName

Jpf_DbGetStunServName gets the STUN server name.

```
ZCHAR * Jpf_DbGetStunServName()
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

The STUN server name if found, or ZNULL.

4.5.3.6.2 Jpf_DbGetStunServPort

Jpf_DbGetStunServPort gets the STUN server UDP port.

```
ZINT Jpf_DbGetStunServPort(ZULONG *pdwPort)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwPort
A pointer to the STUN server UDP port.
```

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwIp;  
  
.....  
/* Get the STUN server UDP port. */  
Jpf_DbGetStunServPort(dwPort);
```

4.5.3.6.3 Jpf_DbGetStunTryTimeLen

Gets the STUN time length for which one query try is waited.

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

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTimeLen  
The STUN time length.
```

[Return value]

Returns ZOK.

4.5.3.6.4 Jpf_DbGetStunTryCnt

Gets the number of tries for a query.

```
ZINT Jpf_DbGetStunTryCnt(ZULONG *pdwTryCnt);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwTryCnt  
The number of tries.
```

[Return value]

Returns ZOK.

4.5.3.6.5 Jpf_DbSetStunServName

Jpf_DbSetStunServName sets the STUN server name.

```
ZINT Jpf_DbSetStunServName ( ZCHAR *pcServName )
```

[Parameters]

Input parameters:

```
ZCHAR *pcServName  
The STUN server name.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZCHAR *pcServName;  
ZINT iRet;  
.....  
/* Set the STUN server name.*/  
iRet = Jpf_DbSetStunServName (pcServName);
```

4.5.3.6.6 Jpf_DbSetStunServPort

Jpf_DbSetStunServPort sets the STUN server UDP port.

```
ZINT Jpf_DbSetStunServPort ( ZULONG dwPort )
```

[Parameters]

Input parameters:

```
ZULONG dwPort  
The STUN server UDP port.
```

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.  
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwPort;  
.....  
/* Set the STUN server UDP port. */  
Jpf_DbSetStunServPort(dwPort);
```

4.5.3.6.7 Jpf_DbSetStunTryTimeLen

Sets the STUN time length for which one query try is waited.

```
ZINT Jpf_DbSetStunTryTimeLen(ZULONG dwTimeLen);
```

[Parameters]

Input parameters:

```
ZULONG dwTimeLen  
The time length.
```

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.3.6.8 Jpf_DbSetStunTryCnt

Sets the number of tries for one query.

```
ZINT Jpf_DbSetStunTryCnt(ZULONG dwTryCnt);
```

[Parameters]

Input parameters:

```
ZULONG dwTryCnt  
The number of tries.
```

[Return value]

Returns ZOK.

4.5.3.7 DNS Data Interfaces

4.5.3.7.1 Jpf_DbGetDnsListenPort

Jpf_DbGetDnsListenPort gets the DNS listen port.

```
ZINT Jpf_DbGetDnsListenPort(ZULONG *pdwPort)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwPort

A pointer to the DNS listen port.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG dwPort;  
  
.....  
/* Get the DNS listen port. */  
Jpf_DbGetDnsListenPort(&dwPort);
```

4.5.3.7.2 Jpf_DbGetDnsServIp

Jpf_DbGetDnsServIp gets the DNS server IP address.

```
ZINT Jpf_DbGetDnsServIp(ZULONG *pdwIp)
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwIp

A pointer to the DNS server IP address.

[Return value]

ZOK: Operation succeeded.

ZFAILED: Operation failed.

[Example]

```
ZULONG dwIp;  
  
.....  
/* Get the DNS server IP address. */  
Jpf_DbGetDnsServIp(&dwIp);
```

4.5.3.7.3 Jpf_DbGetDnsServIpv6

Jpf_DbGetDnsServIpv6 gets the IPv6 address of the DNS server.

```
ZINT Jpf_DbGetDnsServIpv6 ( ZCHAR **ppucIpv6 )
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZCHAR **ppucIpv6
```

The pointer to the pointer which points to the IPv6 address.

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZCHAR *pucIpv6;
```

```
.....
```

```
/* Get the IPv6 address of the DNS server. */
```

```
Jpf_DbGetDnsServIpv6 (&pucIpv6);
```

4.5.3.7.4 Jpf_DbGetDnsServPort

Jpf_DbGetDnsServPort gets the DNS server UDP port.

```
ZINT Jpf_DbGetDnsServPort ( ZULONG *pdwPort )
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwPort
```

A pointer to the DNS server UDP port.

[Return value]

```
Returns ZOK.
```

[Example]

```
ZULONG dwPort;  
.....  
/* Get the DNS server UDP port. */  
Jpf_DbGetDnsServPort(&dwPort);
```

4.5.3.7.5 Jpf_DbGet2ndDnsServIp

Jpf_DbGet2ndDnsServIp gets the DNS secondary server IP address.

```
ZINT Jpf_DbGet2ndDnsServIp(ZULONG *pdwIp)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwIp  
A pointer to the DNS secondary server IP address.
```

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwIp;  
.....  
/* Get the DNS secondary server IP address. */  
Jpf_DbGet2ndDnsServIp(&dwIp);
```

4.5.3.7.6 Jpf_DbGet2ndDnsServIpv6

Jpf_DbGet2ndDnsServIpv6 gets the IPv6 address of the DNS secondary server.

```
ZINT Jpf_DbGet2ndDnsServIpv6(ZCHAR **ppucIpv6)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZCHAR **ppucIpv6  
The pointer to the pointer which points to the IPv6 address.
```

[Return value]

Returns ZOK.

[Example]

```
ZCHAR *pucIpv6;
.....
/* Get the IPv6 address of the DNS secondary server. */
Jpf_DbGet2ndDnsServIpv6 (&pucIpv6);
```

4.5.3.7.7 Jpf_DbGet2ndDnsServPort

Jpf_DbGet2ndDnsServPort gets the DNS secondary server UDP port.

```
ZINT Jpf_DbGet2ndDnsServPort(ZULONG *pdwPort)
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwPort
A pointer to the DNS secondary server UDP port.
```

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwPort;
.....
/* Get the DNS secondary server UDP port. */
Jpf_DbGet2ndDnsServPort (&dwPort);
```

4.5.3.7.8 Jpf_DbSetDnsListenPort

Jpf_DbSetDnsListenPort set the DNS listen port.

```
ZINT Jpf_DbSetDnsListenPort(ZULONG dwPort)
```

[Parameters]

Input parameters:

```
ZULONG dwPort
The DNS listen port.
```

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwPort;
.....
/* Set the DNS listen port. */
Jpf_DbSetDnsListenPort(dwPort);
```

4.5.3.7.9 Jpf_DbSetDnsServIp

Jpf_DbSetDnsServIp sets the DNS server IP address.

```
ZINT Jpf_DbSetDnsServIp(ZULONG dwIp)
```

[Parameters]

Input parameters:

```
ZULONG dwIp
The DNS server IP address.
```

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwIp;
.....
/* Set the DNS server IP address. */
Jpf_DbSetDnsServIp(dwIp);
```

4.5.3.7.10 Jpf_DbSetDnsServIpv6

Jpf_DbSetDnsServIpv6 sets the IPv6 address of the DNS server.

```
ZINT Jpf_DbSetDnsServIpv6(ZUCHAR *pucIpv6)
```

[Parameters]

Input parameters:

```
ZUCHAR *pucIpv6
```

The pointer to the IPv6 address.

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZUCHAR *pucIpv6;
.....
/* Set the IPv6 address of the DNS server. */
Jpf_DbSetDnsServIpv6(pucIpv6);
```

4.5.3.7.11 Jpf_DbSetDnsServPort

Jpf_DbSetDnsServPort sets the DNS server UDP port.

```
ZINT Jpf_DbSetDnsServPort(ZULONG dwPort)
```

[Parameters]**Input parameters:**

```
ZULONG dwPort
```

The DNS server UDP port.

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwPort
.....
/* Set the DNS server UDP port. */
Jpf_DbSetDnsServPort(dwPort);
```

4.5.3.7.12 Jpf_DbSet2ndDnsServIp

Jpf_DbSet2ndDnsServIp sets the DNS secondary server IP address.

```
ZINT Jpf_DbSet2ndDnsServIp(ZULONG dwIp)
```

[Parameters]

Input parameters:

ZULONG dwIp
The DNS secondary server IP address.

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZULONG dwIp;
.....
/* Set the DNS secondary server IP address. */
Jpf_DbSet2ndDnsServIp(dwIp);
```

4.5.3.7.13 Jpf_DbSet2ndDnsServIpv6

Jpf_DbSet2ndDnsServIpv6 sets the IPv6 address of the DNS secondary server.

```
ZINT Jpf_DbSet2ndDnsServIpv6(ZUCHAR *pucIpv6)
```

[Parameters]

Input parameters:

ZUCHAR *pucIpv6
The pointer to the IPv6 address.

Output parameters:

None.

[Return value]

Returns ZOK.

[Example]

```
ZUCHAR *pucIpv6;
.....
/* Set the IPv6 address of the DNS secondary server. */
Jpf_DbSet2ndDnsServIpv6(pucIpv6);
```

4.5.3.7.14 Jpf_DbSet2ndDnsServPort

Jpf_DbSet2ndDnsServPort sets the DNS secondary server UDP port.

```
ZINT Jpf_DbSet2ndDnsServPort(ZULONG dwPort)
```

[Parameters]

Input parameters:

```
ZULONG dwPort
```

The DNS secondary server UDP port.

Output parameters:

None.

[Return value]

```
ZOK: Operation succeeded.
```

```
ZFAILED: Operation failed.
```

[Example]

```
ZULONG dwPort;
```

```
.....
```

```
/* Set the DNS secondary server UDP port. */
```

```
Jpf_DbSet2ndDnsServPort(dwPort);
```

4.5.4 Interfaces for Data Management

These interfaces are included in `jpf_db.h`.

4.5.4.1 *Jpf_DbAppMain*

Applies the main settings in modules including SIP, DNS, STUN, RTP, and Media.

```
ZINT Jpf_DbAppMain();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.4.2 *Jpf_DbAppUser*

Applies the latest user information that has been set on the endpoint.

```
ZINT Jpf_DbAppUser();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.4.3 *Jpf_DbAppSip*

Applies the latest SIP information that has been set.

```
ZINT Jpf_DbAppSip();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.4.4 *Jpf_DbAppMedia*

Applies the latest Media information that has been set.

```
ZINT Jpf_DbAppMedia();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.4.5 *Jpf_DbAppRtp*

Applies the latest RTP information that has been set.

```
ZINT Jpf_DbAppRtp();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.4.6 *Jpf_DbAppStun*

Applies the latest STUN information that has been set.

```
ZINT Jpf_DbAppStun();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK.

4.5.4.7 *Jpf_DbAppDns*

Applies the latest DNS information that has been set.

```
ZINT Jpf_DbAppDns();
```

[Parameters]

Input parameters:

None.

Output parameters:

None .

[Return value]

Returns ZOK .