FDK API Manual for C

June 2015

ITS Company

Contents

Overview

This document is the API specification for C of the CallFDK

The CallFDK is the API Library for using FDK engine provided by ITS

System Environments

C compilers

Installation files

Linux

- callfdk shared object binary
  - libcallfdk.so
  - libcallfdk_dat.so
  - libxerces-c*
- callfdk header files/callfdk_c.h, svcxxxx.h …
- MsgDic.exml : FDK SErvice Message Dictionary file
- UserKey.lic : API identification key file
- callfdk.cfg : CallFDK API configuration file
- test_main.c : sample program source file
- Makefile : sample make file

Runtime Environments

Register the directory which the CallFDK API library is installed at LD_LIBRARY_PATH.

Linux/Unix

```
$ export LD_LIBRARY_PATH=/path/to/lib:$LD_LIBRARY_PATH
```

Sample codes for using CallFDK API

Example : test_main.c

```
#include <stdio.h>
#include "callfdk_c.h"

#include "svc2106.h"
```

```
int
main()
{
    CallFdk_Error tError;
    TMsg *pIn = NULL;
    TMsg *pOut = NULL;
    TMsg *pSub = NULL;
    TMsg *pAmort = NULL;
    int i;
    int iRet;
    int iArrSize;
    char *sBuf = NULL;
    double dVal;

    /* init CallFdk API */
    iRet = CallFdk_Initialize("callfdk.cfg", &tError);
    if (iRet < 0) {
        printf("Error: %s\n", tError.sErrText);
        return -1;
    }

    /* new & set Input message */
    pIn = MSG_GetNewMsg(MSG_IrNoteVanilla, &tError);
    if (pIn == NULL) {
        printf("Error: pIn: %s\n", tError.sErrText);
        return -1;
    }
    pSub = MSG_GetNewMsg(MSG_IrNotionalInfo, &tError);
    if (pSub == NULL) {
        printf("Error: pSub: %s\n", tError.sErrText);
        return -1;
    }
    iRet = MSG_SetDbl(pSub, TAG_dNotionAmt, 100000);
    if (iRet < 0) {
        printf("Error: %s\n", MSG_GetLastError(pSub));
        MSG_Destroy(pIn);
        MSG_Destroy(pSub);
        return -1;
    }
    iRet = MSG_SetStr(pSub, TAG_sCcy, "USD");
    if (iRet < 0) {
        printf("Error: %s\n", MSG_GetLastError(pSub));
        MSG_Destroy(pIn);
        MSG_Destroy(pSub);
        return -1;
    }
    for (i = 0; i < 3; i++) {
        pAmort = MSG_GetNewMsg(MSG_IrAmortSchedule, &tError);
        if (pAmort == NULL) {
            printf("Error: pAmort: %s\n", tError.sErrText);
            MSG_Destroy(pIn);
            MSG_Destroy(pSub);
            MSG_Destroy(pAmort);
            return -1;
        }
        iRet = MSG_SetInt(pAmort, TAG_iPeriodNo, i+1);
        if (iRet < 0) {
            printf("Error: %s\n", MSG_GetLastError(pSub));
```

```
            MSG_Destroy(pIn);
            MSG_Destroy(pSub);
            MSG_Destroy(pAmort);
            return -1;
        }
        iRet = MSG_SetDbl(pAmort, TAG_dAmortAmt, 3.14 * i);
        if (iRet < 0) {
            printf("Error: %s\n", MSG_GetLastError(pSub));
            MSG_Destroy(pIn);
            MSG_Destroy(pSub);
            MSG_Destroy(pAmort);
            return -1;
        }
        iRet = MSG_SetSub(pSub, TAG_maIrAmortSchedule, pAmort);
        if (iRet < 0) {
            printf("Error: %s\n", MSG_GetLastError(pSub));
            MSG_Destroy(pIn);
            MSG_Destroy(pSub);
            MSG_Destroy(pAmort);
            return -1;
        }
    }
    iRet = MSG_SetSub(pIn, TAG_mIrNotionalInfo, pSub);
    if (iRet < 0) {
        printf("Error: %s\n", MSG_GetLastError(pSub));
        MSG_Destroy(pIn);
        MSG_Destroy(pSub);
        return -1;
    }

    /* prepare output message */
    pOut = MSG_GetNewMsg(MSG_IrNoteOut, &tError);
    if (pOut == NULL) {
        printf("Error: pOut: %s\n", tError.sErrText);
        MSG_Destroy(pIn);
        return -1;
    }

    /* call service */
    iRet = CallFdk_CallService(2106, pIn, pOut, &tError);
    if (iRet < 0) {
        printf("Error: %s\n", tError.sErrText);
        MSG_Destroy(pIn);
        MSG_Destroy(pOut);
        return -1;
    }
    MSG_Destroy(pIn);

    /* refer to output message */
    iArrSize = MSG_GetArrSize(pOut, TAG_maIrNotePrice);
    for (i = 0; i < iArrSize; i++) {
        pSub = MSG_GetAtSubArr(pOut, TAG_maIrNotePrice, i, &iRet);
        if (pSub == NULL) {
            printf("Error: %s\n", MSG_GetLastError(pOut));
            MSG_Destroy(pOut);
            return -1;
        }
        sBuf = MSG_GetStrEx(pSub, TAG_sPriceType, &iRet);
        if (sBuf == NULL) {
```

```
            printf("Error: %s\n", MSG_GetLastError(pSub));
            MSG_Destroy(pOut);
            return -1;
        }
        printf("maIrNotePrice[%d].sPriceType : %s\n", i, sBuf);
        free(sBuf);
        iRet = MSG_GetDbl(pSub, TAG_dDirtyPrice, &dVal);
        if (iRet < 0) {
            printf("Error: %s\n", MSG_GetLastError(pSub));
            MSG_Destroy(pOut);
            return -1;
        }
        printf("maIrNotePrice[%d].dDirtyPrice : %e\n", i, dVal);

    }
    MSG_Destroy(pOut);

    return 0;
}
```

1. Include service header file(svc####.h)
2. Call CallFdk_Initialize()
3. Make new input message and setting data(pIn)
4. Call CallFdk_CallService()
5. Use output message data
6. Destroy message instances(pIn, pOut)

CallFdk functions

These functions are used for calling the service of FDK engine

int CallFdk_Initialize(...)

Description

This function is used to initialize the CallFDK API library with a given configuration file. This function must be called only once in the entire program.

Parameters

- const char *p_szCfgFile : Configuration file path
- CallFdk_Error *p_pError : CallFdk_Error structure

Return : int

If this return value is less than zero, it means that an error occurs in this function call.
If an error occurs in this function, the reason of error is stored in CallFdk_Error instance

Example

```
CallFdk_Error tError;
iRet = CCallFdk::Initialize("/path/to/callfdk.cfg", &tError);
if (iRet < 0) {
  printf("Error: %s\n", tError.szErrText);
  return -1;
}
```

Configuration file sample

```
key_file=UserKey.lic
msg_dic=MsgDic.exml
server=sq.fnpricing.com 5300
timeout=90
```

key_file and msg_dic keywords both in full or relative path can be expressed.

int CallFdk_InitializeEx(...)

Description

This function is the alternatives of the previous CallFdk_Initialize() function. This is used to intialize the CallFDK API library with each argument. This function must also be called only once in the entire program.

Parameters

- const char *p_szKeyFile : API identification key file path
- const char *p_szDicFile : FDK Service Message Dictionary file path
- const char *p_szServer : FDK server address(DNS name or IP address)
- int p_iPort : FDK server port number

- int p_iTimeout : timeout(seconds) which the API waits for the response of the FDK server
- CallFdk_Error *p_pError : CallFdk_Error instance for error

Return : int

If this value is less than zero, an error occurs in the function call. The reason of error is stored in CallFdk_Error instance.

Example

```
CallFdk_Error tError;
iRet = CCallFdk::InitializeEx("/path/to/UserKey.lic",
    "/path/to/MsgDic.exml", "sq.fnpricing.com", 5300, 90,&tError);
if (iRet < 0) {
  printf("Error: %s\n", tError.szErrText);
  return -1;
}
```

int CallFdk_CallService(...)

Description

This function is used to call service corresponding to p_iSvcNum with input message(p_pIn) instance. The response from the FDK engine is stored in output message(p_pOut) instance. Input and Output message pointer must be allocated by MSG_GetNewMsg() function before calling this function.

Parameters

- int p_iSvcNum : Service number
- TMsg *p_pIn : Input message instance
- TMsg *p_pOut : Output message instance
- CallFdk_Error *p_pError : CallFdk_Error instance

For the service number, input message and output message, refer to FDK Service Message Reference Manual

Return : int

If this return value is less than zero, an error occurs in this function call. The reason of error is stored in CallFdk_Error instance

Example

```
TMsg *pIn = NULL;
TMsg *pOut = NULL;

pIn = MSG_GetNewMsg(MSG_IrNoteVanilla);
if (pIn == NULL) {
    printf("Error: pOut\n");
    MSG_Destroy(pIn);
    return -1;
}
/* MSG_Setxxx() setting */
/* ... */
```

```
pOut = MSG_GetNewMsg(MSG_IrNoteOut);
if (pIn == NULL) {
    printf("Error: pOut\n");
    MSG_Destroy(pIn);
    return -1;
}

iRet = CallFdk_CallService(2106, pIn, pOut, &tError);
if (iRet < 0) {
    printf("Error: %s\n", tError.szErrText);
    return -1;
}
...
```

CallFdk_Error structure

This structure is used to handle errors occurring when CallFdk_* functions are called.

In case CallFdk_CallService(), this structure has the error from the FDK engine

```
typedef struct {
    int iCode;
    char sErrText[1024];
    int iRefSvcNum;
} CallFdk_Error;
```

- int iCode : Error Code
- char szErrText[] : Error Text
- int iRefSvcNum : Service number or message number for referring to this error

Service / Message

Service

The Service is a basic unit of the transaction with the FDK engine. This is represented by the service number. Services are classified by the product type, the algorithm and the feature in detail. Refer to Service Message Reference Manual.

Message Object

The Message Object is a basic unit of the data block which the CallFDK API exchanges with the FDK server. This is divided into two objects such as the request and the response message.
Input message : the data which the API sends to the FDK server.
Output message : the data which the API receives from the FDK server.
Each Message Object can hold other message object as sub message.

The Structure of Message Object

The Message Object can hold the data types as following.

Data Type	C Type	Getting Data	Setting Data	prefix	TAG_* example
string	char *	MSG_GetStr	MSG_SetStr	s	TAG_sCcy
integer	int	MSG_GetInt	MSG_SetInt	i	TAG_iPeriodNo
double	double	MSG_GetDbl	MSG_SetDbl	d	TAG_dFixedRate
boolean	int	MSG_GetBool	MSG_SetBool	b	TAG_bDiffYN
date(YYYYMMDD)	char[9]	MSG_GetDate	MSG_SetDate	n	TAG_nDate
sub message	TMsg *	MSG_GetSub	MSG_SetSub	m	TAG_mIrVanilla
string array	char *	MSG_GetAtStrArr	MSG_SetStr	sa	TAG_saCode
integer array	int	MSG_GetAtIntArr	MSG_SetInt	ia	TAG_iaSeqNum
double array	double	MSG_GetAtDblArr	MSG_SetDbl	da	TAG_daPrice
boolean array	int	MSG_GetAtBoolArr	MSG_SetBool	ba	TAG_baCallYN
date array	char[9]	MSG_GetAtDateArr	MSG_SetDate	na	TAG_naCpnDate
sub message array	TMsg *	MSG_GetAtSubArr	MSG_SetSub	ma	TAG_maNoteCF

The functions which are written in the above table are used to set or get data with message object. The message object can have other messages as sub messages. The service numbers, message numbers and tag numbers are defined in the svc####.h file. The message names and the tag names can be referred to Service Message Reference Manual.

Message Getting/Setting functions

This section explains about how to create and delete message object, how to set and get data from message object.

TMsg *MSG_GetNewMsg(int p_iMsgNum, CallFdk_Error *p_pError)

Description

This function is used to create new message instance by the given message number(p_iMsgNum). The message number is predefined at svc####.h. For more information about message number, refer to FDK Service Message Manual.

Parameters

- int p_iMsgNum : message number
- CallFdk_Error *p_pError : CallFdk_Error structure

Return : TMsg *

The pointer to the newly created message instance

Example

```
TMsg *pMsg = NULL;
CallFdk_Error tError;
pMsg = MSG_GetNewMsg(MSG_IrNoteVanilla, &tError);
if (pMsg == NULL) {
  printf("Error: MSG_GetNewMsg failed(%s)\n", tError.sErrText);
  return -1;
}
```

void MSG_Destroy(TMsg *p_pMsg)

Description

This function is used to destroy the given message instance. The message instance which is created by MSG_GetNewMsg() must be destroyed by this function. Otherwise, memory leak may occur. But if the message instance which is created by MSG_GetNewMsg() is used as sub message, you don't have to destroy it. The message instances which are used as sub message are destroyed automatically when the parent message is destroyed.

Parameters

- TMsg *p_pMsg : message instance pointer

Example

```
TMsg *pMsg = NULL;
CallFdk_Error tError;
pMsg = MSG_GetNewMsg(MSG_IrNoteVanilla, &tError);
if (pMsg == NULL) {
```

```
  printf("Error: MSG_GetNewMsg failed(%s)\n", tError.sErrText);
  return -1;
}
...
MSG_Destroy(pMsg);
```

char *MSG_GetLastError(TMsg *p_pMsg)

Description

This function is used to get what kind of error occurs when the message functions are failed.

The message functions return negative value when they failed.

Parameters

- TMsg *p_pMsg : message instance pointer

Return : char *

Error text string. This pointer does not have to be freed.

Example

```
int iRet;
...
iRet = MSG_SetStr(pMsg, TAG_sCcy, "USD");
if (iRet < 0) {
  printf("Error: MSG_SetStr failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_Dump(TMsg *p_pMsg, const char *p_sFileName, CallFdk_Error *p_pError)

Description

This function is used to dump the data which the given messages are hold.

If the p_sFileName is NULL, the function dumps data to the standard output, otherwise writes data to the given file. This is usually used for debugging.

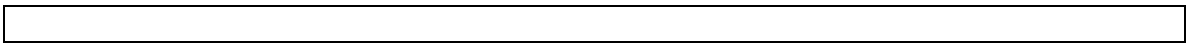
Parameters

- TMsg *p_pMsg : message instance pointer
- const char *p_sFileName : filename path
- CallFdk_Error *p_pError : CallFdk_Error structure

Example

```
...
MSG_Dump(pMsg, NULL, NULL);  /* dump pMsg to stdout */

iRet = MSG_Dump(pMsg, "msg_dump.log", &tError);
if (iRet < 0) {
  printf("Error: MSG_Dump failed(%s)\n", tError.sErrText);
  return -1;
}
```



int MSG_SetStr(TMsg *p_pMsg, int p_iTagNum, char *p_sVal)

Description

This function is used to set a string type value to the tag value of the given message instance. If the tag is defined as an array field and this function is called multiple times, the values are stored as the array data in message object.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- char *p_sVal : string type value to set

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
...
iRet = MSG_SetStr(pMsg, TAG_sCcy, "USD");
if (iRet < 0) {
  printf("Error: MSG_SetStr failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_SetInt(TMsg *p_pMsg, int p_iTagNum, int p_iVal)

Description

This function is used to set an integer type value to the tag value of the given message instance. If the tag is defined as an array field and this function is called multiple times, the values are stored as the array data in message object.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iVal : integer type value to set

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
...
iRet = MSG_SetInt(pMsg, TAG_iDiscrvId, 3);
if (iRet < 0) {
  printf("Error: MSG_SetInt failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
```

```
}
```

int MSG_SetDbl(TMsg *p_pMsg, int p_iTagNum, double p_dVal)

Description

This function is used to set a double type value to the tag value of the given message instance. If the tag is defined as an array field and this function is called multiple times, the values are stored as the array data in message object.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- double p_dValue : double type value to set

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;

...
iRet = MSG_SetDbl(pMsg, TAG_dFairPrice, 3);
if (iRet < 0) {
  printf("Error: MSG_SetDbl failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_SetBool(TMsg *p_pMsg, int p_iTagNum, int p_bVal)

Description

This function is used to set a boolean type value to the tag value of the given message instance. If the tag is defined as an array field and this function is called multiple times, the values are stored as the array data in message object.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_Value : integer type value to set(0 : false, others : true)

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;

...
iRet = MSG_SetBool(pMsg, TAG_bDiffSensYN, 1);
if (iRet < 0) {
  printf("Error: MSG_SetBool failed(%s)\n",MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_SetDate(TMsg *p_pMsg, int p_iTagNum, char *p_sDate)

Description

This function is used to set a date type value to the tag value of the given message instance. If the tag is defined as an array field and this function is called multiple times, the values are stored as the array data in message object. The format of a date type is 8 bytes character as 'YYYYMMDD'. (YYYY : year, MM : month, DD : day)

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- char *p_sDate : string type value to set (YYYYMMDD)

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
...
iRet = MSG_SetDate(pMsg, TAG_nDate, "20150615");
if (iRet < 0) {
  printf("Error: MSG_SetDate failed(%s)\n",MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_SetSub(TMsg *p_pMsg, int p_iTagNum, TMsg *p_pSub)

Description

This function is used to set a sub message instance to the tag value of the given message instance.

If the tag is defined as an array field and this function is called multiple times, the values are stored as the array data in message object.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- TMsg *p_pSub : message instance to set

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
TMsg *pSub = NULL;
...
pSub = MSG_GetNewMsg(MSG_IrVanilla, &tError);
...
iRet = MSG_SetInt(pMsg, TAG_mIrVanilla, pSub);
if (iRet < 0) {
  printf("Error: MSG_SetInt failed(%s)\n",MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_GetStrLen(TMsg *p_pMsg, int p_iTagNum)

Description

This function is used to get the length of the string field corresponding to p_iTagNum from the message instance(p_pMsg). This length is used to determine the size of the string buffer which is used by the MSG_GetStr() function

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)

Return : int

If this value is more than zero, it is the length of the string of the field.
If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iLen;
...
iLen = MSG_GetStrLen(pMsg, TAG_sCcy);
if (iLen < 0) {
  printf("Error: MSG_GetStrLen failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_GetStr(TMsg *p_pMsg, int p_iTagNum, char *p_sVal)

Description

This function is used to get a string value corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- char *p_sVal : string buffer to get(this pointer must be allocated)

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iLen;
int iRet;
char *sCcy = NULL;
...
iLen = MSG_GetStrLen(pMsg, TAG_sCcy);
if (iLen < 0) {
  printf("Error: MSG_GetStrLen failed(%s)\n", MSG_GetLastError(pMsg));
```

```
  return -1;
}
sCcy = (char *)malloc(iLen + 1);
iRet = MSG_GetStr(pMsg, TAG_sCcy, sCcy);
if (iRet < 0) {
  printf("Error: MSG_GetStr failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
...
free(sCcy);
```

char *MSG_GetStrEx(TMsg *p_pMsg, int p_iTagNum, int *p_pErrCode)

Description

This function is used to get a string value corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int *p_pErrCode : error code value(negative if this function failed)

Return : char *

This function returns the pointer to the string corresponding to p_iTagNum. This pointer must be freed after it is used. If this value is NULL, it means that the function call is failed.

Example

```
int iRet;
char *sCcy = NULL;
...
sCcy = MSG_GetStr(pMsg, TAG_sCcy, &iRet);
if (iRet < 0 || sCcy == NULL) {
  printf("Error: MSG_GetStr failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
...
free(sCcy);
```

int MSG_GetInt(TMsg *p_pMsg, int p_iTagNum, int *p_iVal)

Description

This function is used to get an integer type value corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer

- int p_iTagNum : tag number (TAG_{tagname} macros)
- int *p_iVal : integer type variable to get

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
int iDiscrvId;
...
iRet = MSG_GetInt(pMsg, TAG_iDiscrvId, &iDiscrvId);
if (iRet < 0) {
  printf("Error: MSG_GetInt failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_GetDbl(TMsg *p_pMsg, int p_iTagNum, double *p_dVal)

Description

This function is used to get an double type value corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- double *p_dVal : double type variable to get

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
double dFixedRate;
...
iRet = MSG_GetDbl(pMsg, TAG_dFixedRate, &dFixedRate);
if (iRet < 0) {
  printf("Error: MSG_GetDbl failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_GetBool(TMsg *p_pMsg, int p_iTagNum, int *p_bVal)

Description

This function is used to get a boolean type value corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int *p_bVal : integer type variable

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
int bDiffSensYN;
...
iRet = MSG_GetBool(pMsg, TAG_bDiffSensYN, &bDiffSensYN);
if (iRet < 0) {
  printf("Error: MSG_GetBool failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
if (bDiffSensYN) {
  /* this means true */
}
```

int MSG_GetDate(TMsg *p_pMsg, int p_iTagNum, char *p_sVal)

Description

This function is used to get a date type value corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- char *p_sVal : char[9] type variable

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
char nDate[8+1];
...
iRet = MSG_GetDate(pMsg, TAG_nDate, &nDate); /* YYYYMMDD */
if (iRet < 0) {
  printf("Error: MSG_GetDate failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

TMsg *MSG_GetSub(TMsg *p_pMsg, int p_iTagNum, int *p_pErrCode)

Description

This function is used to get a sub message instance corresponding to p_iTagNum from the message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int *p_pErrCode : integer variable to get error code

Return : TMsg *

This return value means the pointer to sub message instance. If this is NULL, it means the function call failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
TMsg *pSub = NULL;
...
pSub = MSG_GetSub(pMsg, TAG_mIrVanilla, &iRet);
if (iRet < 0 || pSub == NULL) {
  printf("Error: MSG_GetSub failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
```

int MSG_GetArrSize(TMsg *p_pMsg, int p_iTagNum)

Description

This function is used to get the size of array corresponding to p_iTagNum from message instance(p_pMsg).

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)

Return : int

This return value is the size of array

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
int iArrSize;
int i;
...
iArrSize = MSG_GetArrSize(pMsg, TAG_sCcy);
if (iArrSize < 0) {
  printf("Error: MSG_GetArrSize failed(%s)\n", MSG_GetLastError(pMsg));
  return -1;
}
for (i = 0; i < iArrSize; i++) {
  iRet = MSG_GetAtIntArr(pMsg, TAG_iDisrcrvId, i, &iId);
  ...
}
```

int MSG_GetAtStrLen(TMsg *p_pMsg, int p_iTagNum, int p_iIdx)

Description

This function is used to get the length of the string corresponding to p_iTagNum and p_iIdx which specifies an index of array elements. This value is used to determine the size of the string buffer which is used by the MSG_GetAtStrArr() function

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)

Return : int

This return value is the length of the string.

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iLen;
...
for (i = 0; i < iArrSize; i++) {
  iLen = MSG_GetAtStrLen(pMsg, TAG_sCcy, i);
  if (iLen < 0) {
    printf("Error: MSG_GetAtStrLen failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
}
```

int MSG_GetAtStrArr(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, char *p_sVal)

Description

This function is used to get the string value corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)
- char *p_sVal : string buffer to get(this pointer must be allocated)

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iLen;
int iRet;
char *sCcy = NULL;
...
for (i = 0; i < iArrSize; i++) {
  iLen = MSG_GetAtStrLen(pMsg, TAG_sCcy);
  if (iLen < 0) {
    printf("Error: MSG_GetAtStrLen failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
  sCcy = (char *)malloc(iLen + 1);
  iRet = MSG_GetAtStrArr(pMsg, TAG_sCcy, sCcy);
  if (iRet < 0) {
    printf("Error: MSG_GetAtStrArr failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
```

```
  }
  ...
  free(sCcy);
}
```

char *MSG_GetAtStrArrEx(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, int *p_pErrCode)

Description

This function is used to get the string value corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)
- int *p_pErrCode : error code value(negative if this function failed)

Return : char *

This function returns the pointer to the string corresponding to p_iTagNum. This pointer must be freed after it is used. If this value is NULL, it means that the function call is failed.

Example

```
int iRet;
char *sCcy = NULL;
...
for (i = 0; i < iArrSize; i++) {
  sCcy = MSG_GetAtStrArrEx(pMsg, TAG_sCcy, &iRet);
  if (iRet < 0 || sCcy == NULL) {
    printf("Error: MSG_GetAtStrArrEx failed(%s)\n",
           MSG_GetLastError(pMsg));
    return -1;
  }
  ...
  free(sCcy);
}
```

int MSG_GetAtIntArr(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, int *p_iVal)

Description

This function is used to get the integer value corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer

- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)
- int *p_iVal : integer type variable to get

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
int iDiscrvId;
...
for (i = 0; i < iArrSize; i++) {
  iRet = MSG_GetAtIntArr(pMsg, TAG_iDiscrvId, &iDiscrvId);
  if (iRet < 0) {
    printf("Error: MSG_GetAtIntArr failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
}
```

int MSG_GetAtDblArr(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, double *p_dVal)

Description

This function is used to get the double value corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)
- double *p_dVal : double type variable to get

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
double dFixedRate;
...
for (i = 0; i < iArrSize; i++) {
  iRet = MSG_GetAtDblArr(pMsg, TAG_dFixedRate, &dFixedRate);
  if (iRet < 0) {
    printf("Error: MSG_GetAtDblArr failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
```

```
}
```

int MSG_GetAtBoolArr(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, int *p_bVal)

Description

This function is used to get the boolean value corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)
- int *p_bVal : integer type variable to get

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
int bDiffSensYN;
...
for (i = 0; i < iArrSize; i++) {
  iRet = MSG_GetAtBoolArr(pMsg, TAG_bDiffSensYN, &bDiffSensYN);
  if (iRet < 0) {
    printf("Error: MSG_GetAtBoolArr failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
  if (bDiffSensYN) {
    /* this means true */
  }
}
```

int MSG_GetAtDateArr(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, char *p_sVal)

Description

This function is used to get the date value corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)

- int p_iIdx : array index to get(index starts at zero)
- char *p_sVal : char[9] type variable

Return : int

If this return value is less than zero, it means that the function call is failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
char nDate[8+1];
...
for (i = 0; i < iArrSize; i++) {
  iRet = MSG_GetAtDateArr(pMsg, TAG_nDate, &nDate); /* YYYYMMDD */
  if (iRet < 0) {
    printf("Error: MSG_GetAtDateArr failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
}
```

TMsg *MSG_GetAtSubArr(TMsg *p_pMsg, int p_iTagNum, int p_iIdx, int *p_pErrCode)

Description

This function is used to get the sub message instance corresponding to p_iTagNum and p_iIdx which specifies an index of array elements.

Parameters

- TMsg *p_pMsg : message instance pointer
- int p_iTagNum : tag number (TAG_{tagname} macros)
- int p_iIdx : array index to get(index starts at zero)
- int *p_pErrCode : integer variable to get error code

Return : TMsg *

This return value means the pointer to sub message instance. If this is NULL, it means the function call failed. Msg_GetLastError() function let you know the reason why it failed.

Example

```
int iRet;
TMsg *pSub = NULL;
...
for (i = 0; i < iArrSize; i++) {
  pSub = MSG_GetAtSubArr(pMsg, TAG_mIrVanilla, &iRet);
  if (iRet < 0 || pSub == NULL) {
    printf("Error: MSG_GetAtSubArr failed(%s)\n",
            MSG_GetLastError(pMsg));
    return -1;
  }
}
```

The End.