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

To use CallFDK API, a developer needs the machine which is installed Microsoft .Net Framework 4.0. Visual Studio 2010 is used as a compiler.

Installation files

- callfdk.dll : CallFDK API assembly dll
- msg_class.dll : Service Message assembly dll
- MsgDic.exml : FDK Service Message Dictionary file
- UserKey.lic : API identification key file
- callfdk.cfg : CallFDK API configuration file

Sample codes

Example : TestMain.cs

```
using System;
using System.Collections.ObjectModel;

using callfdk;

namespace test
{
  class TestMain
  {
      static void Main(string[] args)
      {
          try {
              CCallFdk.Initialize("callfdk.cfg");

              CallfdkTest();
          } catch (CMsgException e) {
              Console.WriteLine("Error: " + e.Message);
          }
      }

      static void CallfdkTest()
      {
          M_IrNoteVanilla mIn = new M_IrNoteVanilla();
```

```
        mIn.mIrNotionalInfo = new M_IrNotionalInfo();
        mIn.mIrNotionalInfo.dNotionAmt = 10000.0;
        mIn.mIrNotionalInfo.sCcy = "KRW";

        mIn.mIrNotionalInfo.maIrAmortSchedule =
                  new ObservableCollection<M_IrAmortSchedule>();
        M_IrAmortSchedule pAmort = new M_IrAmortSchedule();
        pAmort.iPeriodNo = 7;
        pAmort.dAmortAmt = 3000.0;
        mIn.mIrNotionalInfo.maIrAmortSchedule.Add(pAmort);
        pAmort = new M_IrAmortSchedule();
        pAmort.iPeriodNo = 8;
        pAmort.dAmortAmt = 7000.0;
        mIn.mIrNotionalInfo.maIrAmortSchedule.Add(pAmort);

        mIn.mIrNotionalInfo.iDisCrvId = 1;

        mIn.mIrVanilla = new M_IrVanilla();
        mIn.mIrVanilla.sPayRcvSect = "P";
        mIn.mIrVanilla.sPayTmPtC = "R";
        mIn.mIrVanilla.iFixedPeriodNo = 1;
        mIn.mIrVanilla.dFixedRate = 0.03;
        mIn.mIrVanilla.sDayCntConv = "A365";

        mIn.mIrVanilla.maIrCpnSchedule =
                  new ObservableCollection<M_IrCpnSchedule>();
        M_IrCpnSchedule pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 1;
        pSch.nFormerDate = "20150625";
        pSch.nAfterDate = "20150925";
        pSch.nCashflowDate = "20150925";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 2;
        pSch.nFormerDate = "20150925";
        pSch.nAfterDate = "20151225";
        pSch.nCashflowDate = "20151225";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 3;
        pSch.nFormerDate = "20151225";
        pSch.nAfterDate = "20160325";
        pSch.nCashflowDate = "20160325";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 4;
        pSch.nFormerDate = "20160325";
        pSch.nAfterDate = "20160625";
        pSch.nCashflowDate = "20160625";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 5;
        pSch.nFormerDate = "20160625";
        pSch.nAfterDate = "20160925";
        pSch.nCashflowDate = "20160925";
```

```
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 6;
        pSch.nFormerDate = "20160925";
        pSch.nAfterDate = "20161225";
        pSch.nCashflowDate = "20161225";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 7;
        pSch.nFormerDate = "20160925";
        pSch.nAfterDate = "20170325";
        pSch.nCashflowDate = "20170325";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);

        pSch = new M_IrCpnSchedule();
        pSch.iPeriodNo = 8;
        pSch.nFormerDate = "20170325";
        pSch.nAfterDate = "20170625";
        pSch.nCashflowDate = "20170625";
        mIn.mIrVanilla.maIrCpnSchedule.Add(pSch);


        mIn.mIrVanilla.mIrVanillaCpn = new M_IrVanillaCpn();
        mIn.mIrVanilla.mIrVanillaCpn.mFdkRateIndexB =
                                    new M_FdkRateIndexB();
        M_FdkRateIndexB p =mIn.mIrVanilla.mIrVanillaCpn.mFdkRateIndexB;
        p.sRateIndexCode = "CD91D";
        p.sCcy = "KRW";
        p.sRateType = "S";
        p.sRateMatur = "91D";
        p.sCmpndFreq3 = "I";
        p.sDayCntConv = "A365";
        p.iRateCrvId = 1;
        p.dGearing = 1.0;
        mIn.mIrVanilla.mIrVanillaCpn.dMargin = 0.0;
        mIn.mIrVanilla.mIrVanillaCpn.sDayCntConv = "A365";

        mIn.mIrVanilla.maIrVanillaMon =
                  new ObservableCollection<M_IrVanillaMon>();
        M_IrVanillaMon pMon = new M_IrVanillaMon();
        pMon.iPeriodNo = 2;
        pMon.bCouponFixingYN = true;
        pMon.dCouponRate = 0.023;
        mIn.mIrVanilla.maIrVanillaMon.Add(pMon);


        // Parameter setting
        mIn.mIrParam = new M_IrParam();
        mIn.mIrParam.maFdkRateCrvB =
                  new ObservableCollection<M_FdkRateCurveB>();
        M_FdkRateCurveB pCrv = new M_FdkRateCurveB();
        pCrv.iRateCrvId = 1;
        pCrv.sCcy = "KRW";
        pCrv.sInterpMethod = "L";

        pCrv.maRateCrvTenor =
                  new ObservableCollection<M_RateCurveTenor>();
```

```
        M_RateCurveTenor pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "S";
        pTnr.dRate = 0.0223;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "1D";
        pTnr.sCmpndFreq3 = "I";
        pTnr.sDayCntConv = "ACTB";
        pCrv.maRateCrvTenor.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.0235;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "3M";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.022775;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "6M";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.022575;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "9M";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor.Add(pTnr);


        pCrv.maFwdCrvTenor =
                  new ObservableCollection<M_FwdCurveTenor>();
        M_FwdCurveTenor pFwd = new M_FwdCurveTenor();
        pFwd.dRate = 0.027;
        pFwd.nFwdEndDate = "20160126";
        pFwd.sMatur = "3M";
        pFwd.sCmpndFreq3 = "I";
        pFwd.sDayCntConv = "A365";
        pCrv.maFwdCrvTenor.Add(pFwd);

        pFwd = new M_FwdCurveTenor();
        pFwd.dRate = 0.029;
        pFwd.nFwdEndDate = "20160226";
        pFwd.sMatur = "3M";
        pFwd.sCmpndFreq3 = "I";
        pFwd.sDayCntConv = "A365";
        pCrv.maFwdCrvTenor.Add(pFwd);
```

```
        pCrv.maRateCrvTenor2 =
                  new ObservableCollection<M_RateCurveTenor>();
        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.022375;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "1Y";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.022725;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "2Y";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.02305;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "2Y6M";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.023375;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "3Y";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.024725;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "5Y";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.02575;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "7Y";
```

```
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        pTnr = new M_RateCurveTenor();
        pTnr.sRateType = "Y";
        pTnr.dRate = 0.02705;
        pTnr.mTivMatur = new M_Tiv();
        pTnr.mTivMatur.sDateType = "S";
        pTnr.mTivMatur.sMatur = "10Y";
        pTnr.sCmpndFreq3 = "Q";
        pTnr.sDayCntConv = "A365";
        pCrv.maRateCrvTenor2.Add(pTnr);

        mIn.mIrParam.maFdkRateCrvB.Add(pCrv);

        mIn.mIrParam.nCalcDate = "20150625";

        mIn.mIrEffSensTreeParam = new M_IrEffSensTreeParam();
        mIn.mIrEffSensTreeParam.bDeltaYN = false;
        mIn.mIrEffSensTreeParam.sRDeltaType = "RP";
        mIn.mIrEffSensTreeParam.bVegaYN = false;
        mIn.mIrEffSensTreeParam.bFxVegaYN = false;
        mIn.mIrEffSensTreeParam.bThetaYN = true;
        mIn.mIrEffSensTreeParam.bReCalibYN = true;
        mIn.mIrEffSensTreeParam.bSpotDeltaYN = true;
        mIn.mIrEffSensTreeParam.bAccumDeltaYN = true;
        mIn.mIrEffSensTreeParam.sThetaType = "T";

        mIn.bDiffSensYN = true;

        M_IrNoteOut mOut = new M_IrNoteOut();
        CCallFdkError pError = CCallFdk.CallService(2106, mIn, mOut);
        if (pError != null) {
            Console.WriteLine("Error: " + pError.tostring());
            return;
        } else {
            Console.WriteLine(mOut);
            Console.WriteLine("OK.");
        }
    }
  }
}
```

1. Add using callfdk.*;
2. Call CCallFdk.Initialize()
3. Make new input message and setting data
4. Make new output message
5. Call CCallFdk.CallService()
6. Get data from output message and use it

CCallFdk

This class is the main class for calling service of the FDK engine.

static void Initialize(string p_sCfgFile)

Description

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

Parameters

- string p_sCfgFile : Configuration file path

Example

```
try {
  CCallFdk.Initialize("/path/to/callfdk.cfg");
} catch (CMsgException e) {
  Console.WriteLine("Error: " + e.Message);
}
```

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.

static void Initialize(...)

Description

This method is an alternatives of the previous Initialize() method. This is used to initialize an API library with each argument. This method must be called only once in the entire program.
This is useful when a developer want to manage own configuration file.

Parameters

- string p_sKeyFile : API Identificaiton key file path
- string p_sDicFile : FDK Service Message Dictionary file path
- string p_sServer : FDK server address(DNS name or IP address)
- int p_iPort : FDK server port number
- int p_iTimeout : timeout(seconds) which API waits for the response of the FDK server

Example

```
try {
  CCallFdk.Initialize(
      "/path/to/UserKey.lic",
      "/path/to/MsgDic.exml",
      "sq.fnpricing.com",
```

```
      5300,
      90
  );
} catch (CMsgException e) {
  Console.WriteLine("Error: " + e.Message);
}
```

static CCallFdkError CallService(...)

Description

This method is used to call service which falls on p_iSvcNum with input request Msg.

Parameters

- int p_iSvcNum : Service number
- CMsgBase p_pIn : Request message instance
- CMsgBase p_pOut : Response message instance

For Service number and Msg instance name, refer to FDK Service Message Reference Manual

Return : CCallFdkError

If this return value is null, it means that a service is called successfully.

Otherwise, it means that the call was failed and the reason why it failed is stored in CCallFdkError instance.(refer to CCallFdkError class)

Example

```
try {
  M_IrNoteVanilla mIn = new M_IrNoteVanilla();
  mIn.mIrNotionalInfo.dNotionAmt = 1000000;
  mIn.sCcy = "USD";
  mIn.maIrAmortSchedule = new ObservableCollection<M_IrAmortSchedule>();
  M_IrAmortSchedule mAmort = new M_IrAmortSchdule();
  mAmort.iPeriodNo = 1;
  mAmort.dAmortAmt = 10000.;
  mIn.maIrAmortSchedule.Add(mAmort);
  // ...
  M_IrNoteOut mOut = new M_IrNoteOut();
  CCallFdkError pError = CCallFdk.CallService(2106, mIn, mOut);
  if (pError != null) { // Error
    Console.WriteLine("Error: " + pError.tostring());
  } else { // Success
    Console.WriteLine(mOut);
    if (mOut.maIrNotePrice != null) {
      for (M_IrNotePrice m : mOut.maIrNotePrice) {
        Console.WriteLine(m.dCleanPrice);
        Console.WriteLine(m.dDirtyPrice);
      }
    }
  }
} catch (CMsgException e) {
  Console.WriteLine("Error: " + e.Message);
}
```

CCallFdkError

This class is used to handle errors occurring when CCallFdk method is called.

int GetCode()

Description

This method returns the error code which CCallFdkError instance hold.

Return : int

Error code value

Example

```
CCallFdkError pError = CCallFdk.CallService(...);
if (pError != null) {
  Console.WriteLine(pError.GetCode());
}
```

string GetMessage()

Description

This method returns the error text which CCallFdkError instance hold.

Return : string

Error text string

Example

```
CCallFdkError pError = CCallFdk.CallService(...);
if (pError != null) {
  Console.WriteLine(pError.GetMessage());
}
```

string tostring()

Description

This method returns the error text and the code as the formatted string which CCallFdkError instance hold.

Example

```
CCallFdkError pError = CCallFdk.CallService(...);
if (pError != null) {
  Console.WriteLine(pError.tostring());
}
```

CMsgException

This class is the exception class which the CallFDK API occurs. This is inherited from Exception

class.

Service / Message

Service

The Service is a basic unit of the transaction with 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.
Request message : the data which the API sends to the FDK server.
Response message : the data which the API receives from the FDK server.
Each Message Object can hold other message object as sub message.
All of Message Objects are implemented as classes and they are in assembly dll file which the API provides.
The name of Message classes starts with "M_".

The Structure of Message Object

The Message Object can hold the data types as following.

Variable type	C# Data Type	API defined prefix	Example of variable
string	string	s	sPriceType
integer	int?	i	iPreiodNo
double	double?	d	dNotionalAmt
boolean	bool?	b	bFixedYN
date(YYYYMMDD)	string (YYYYMMDD)	n	nMaturDate
sub message	M_* class	m	mNotionalInfo
string array	ObservableCollection<string>	sa	saCcyList
integer array	ObservableCollection<int>	ia	iaTermNo
double array	ObservableCollection<double>	da	daGearing
boolean array	ObservableCollection<bool>	ba	baGreeksYN
date array	ObservableCollection<string>	na	naExpiryDate
sub message array	ObservableCollection<M_*>	ma	maCashFlowInfo

All of data fields in which a message class holds are declared as public.

Setting/Getting of Message data

Example of setting a single message

```
M_IrNoteVanilla mIn = new M_IrNoteVanilla();
mIn.mIrNotionalInfo = new M_IrNotionalInfo();
mIn.mIrNotionalInfo.dNotionAmt = 100000.;
mIn.mIrNotionalInfo.sCcy = "USD";
```

Example of setting sub message array

```
M_IrNoteVanilla mIn = new M_IrNoteVanilla();
mIn.mIrNotionalInfo = new M_IrNotionalInfo();
mIn.mIrNotionalInfo.dNotionAmt = 100000.;
mIn.mIrNotionalInfo.sCcy = "USD";
mIn.maIrAmortSchedule = new ObservableCollection<M_IrAmortSchedule>();
M_IrAmortSchedule mAmort = new M_IrAmortSchedule();
mAmort.iPeroidNo = 1;
mAmort.dAmortAmt = 100000.;
mIn.maIrAmortSchedule.Add(mAmort);

mAmort = new M_IrAmortSchedule();
mAmort.iPeroidNo = 2;
mAmort.dAmortAmt = 50000.;
mIn.maIrAmortSchedule.Add(mAmort);
...
```

Example of getting message data

```
// ...
M_IrNoteOut mOut = new M_IrNoteOut();
CCallFdkError pError = CCallFdk.CallService(2106, mIn, mOut);
//...

for (M_IrNotePrice m : mOut.maIrNotePrice) {
  Console.WriteLine("PriceType : " + m.sPriceType);
  Console.WriteLine("dDirtyPrice : " + m.dDirtyPrice);
  // ...
  for (M_IrNoteCF m2 : m.maIrNoteCF) {
    Console.WriteLine("PeriodNo : " + m2.iPeriodNo);
    Console.WriteLine("nFormerDate : " + m2.nFormerDate);
  }
}
Console.WriteLine("Test : " + mOut.sDummyStr);
```

The End.