CISUtility說明文件

[1 開發環境 2](#_Toc360094856)

[1.1 需要的專案/DLL 2](#_Toc360094857)

[1.2 引用外部DLL的存放位置 2](#_Toc360094858)

[1.3 測試專案 3](#_Toc360094859)

[2 開發流程 4](#_Toc360094860)

[2.1 參考 4](#_Toc360094861)

[2.2 實作OMSService 4](#_Toc360094862)

[2.3 實作PlugIn功能 5](#_Toc360094863)

[2.3.1 實作OMSService所需的設定檔 5](#_Toc360094864)

[2.3.2 實作設定檔載入器 5](#_Toc360094865)

[3 測試流程 6](#_Toc360094866)

[3.1 測試環境 6](#_Toc360094867)

[3.2 準備設定檔範本 6](#_Toc360094868)

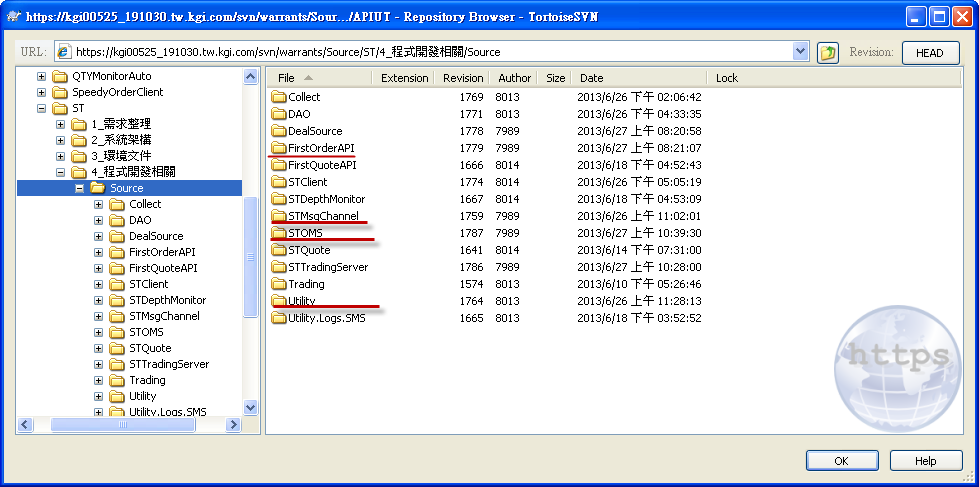
[3.3 準備動態載入所需項目並測試 7](#_Toc360094869)

[3.4 OrderAPIHost簡介 8](#_Toc360094870)

# 開發環境

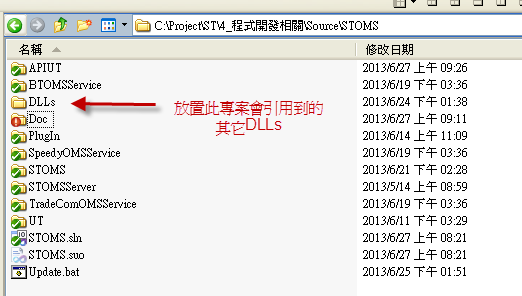
## 需要的專案/DLL

* ST/Source/Utility
  + KGI.TW.Der.Utility.dll
  + KGI.TW.Der.Utility.GUI.dll
* ST/Source/STMsgChannel
  + KGI.TW.Der.ST.MsgChannel.dll
* ST/Source/STOMS
  + KGI.TW.Der.STOMS.dll



## 引用外部DLL的存放位置

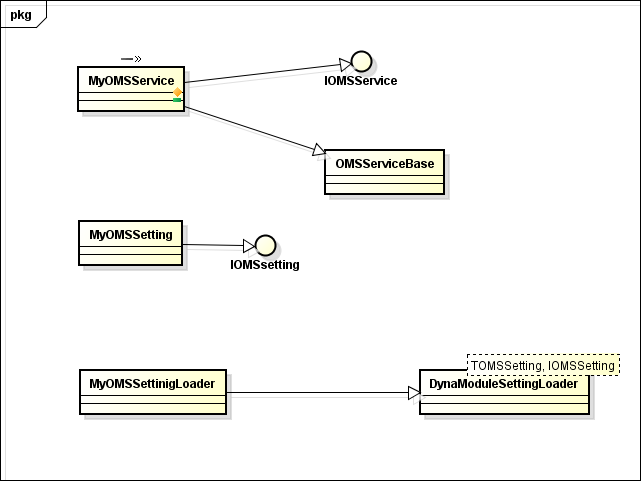
每個專案在引用外部DLL時，都是參考到本身sln目錄的次目錄DLLs裏。所以下載所需專案後，可以在專案的sln下新增一個DLLs的目錄，以便將所需引用到的其它DLL放到這個位置。



## 測試專案

* ST/Source/FirstOrderAPI/APIUT

# 開發流程



需實作的部份大致如上圖中的MyOMSService、MyOMSSetting、MyOMSSettingLoader。

## 參考

實作OMSService時可參考STOMS專案中的以下三個子專案：

* BTOMSService
* SpeedyOMSService
* TradeComOMSService

## 實作OMSService

1. 建立一個C#類別庫的專案
2. 引用1.1需要的專案/DLL
3. 新增一個類別並繼承及實作IOMSService介面
4. 新增的類別也可以同時繼承自OMSServiceBase，這個基底類別有提供實作IOMSService時所需的一些基本功能

其它

1. 在實作的OMSService中如果要觸發OnServiceStatus()事件時，最好是由另一個執行緒來負責通知。如果沒有的話，可以呼叫OMSServiceBase::NotifyWithThread()函式來觸發OnServiceStatus()事件

## 實作PlugIn功能

在提供OMSService實作的DLL中，同時也要實作動態載入OMSService的一些輔助類別。

### 實作OMSService所需的設定檔

1. 實作介面IOMSSetting：在這個實作的設定檔類別中，可以加入一些在設定新實作的OMSService時所需的參數
2. 參數的部份都以屬性的方式來實作，且要實作get/set，因為會是以XML的方式來讀寫這些內容

### 實作設定檔載入器

因為OrderAPI的下單模組是以動態載入的方式載入的，有關設定檔由XML讀出的部份底層無法預先實作，所以實作OMSService及所需設定檔的開發人員也需實作一個設定檔載入器類別。

1. 新增一個類別，並繼承自DynaModuleSettingLoader<TSetting,IOMSSetting>，其中TSetting要填入剛剛實作的設定檔類別的型別名稱

# 測試流程

## 測試環境

取出ST/Source/FirstOrderAPI專案，利用其中的APIUT專案進行測試

## 準備設定檔範本

參考SOMS專案中的UT子專案裏的TestOMSPlugin.TestSettingXML()這個函式。將其中的設定檔型別改成所實作的設定檔型別，並改寫所需的設定值，輸出一份xml。

這份xml就可以作為未來使用的設定檔範本

/// <summary>

/// 可用來產生實作自IOMSSetting的設定檔類別的第一份設定檔

/// </summary>

public static void TestSettingXML()

{

// 產生一個用來進行xml序列化的工具，型別先任意指定沒關係

XmlSerializer xml = new XmlSerializer(typeof(int));

// 產生一個設定檔的儲存體，用DataCollection<T>類別，它是以List<T>作為儲存的方式

// 可以支援多份的設定檔

DataCollection<BTSetting> btSettingPool = new DataCollection<BTSetting>();

// 新增一份設定檔

BTSetting bt = new BTSetting();

bt.AssemblyFile = "KGI.TW.Der.BTOMSService.dll";

bt.OMSType = "KGI.TW.Der.BTOMSService.BTOMSServiceAgent";

bt.ServiceCode = "BT1";

bt.BTService = "17001";

bt.BTNetwork = "";

bt.BTDaemon = "10.32.242.41:7500";

bt.BTSendSubject = "TW.PROP.BT.FE.SERVER.TEST";

bt.BTListenSubject = "TW.PROP.BT.OM.CLIENT.TEST";

bt.BTLoginUID = "7989";

bt.BTLoginPWD = "7989";

btSettingPool.Data.Add(bt); // 將設定檔加入到儲存體中

// 將設定檔寫出成xml檔案

xml.Serialize<DataCollection<BTSetting>>("BTOMSSetting.xml", btSettingPool);

}

## 準備動態載入所需項目並測試

1. 確認可以正常編譯ST/Source/FirstOrderAPI專案中的APIUT子專案
2. 將所實作的新的OMSService(1.1實作OMSService)的DLL及它所需引用到的其它第三方函式庫，放置到APIUT/bin/Debug目錄中，也就是跟測試程式的可執行檔相同的目錄
3. 將產生的設定檔範本(3.2準備設定檔範本)複製一份，並放置到APIUT/bin/Debug目錄中
4. 複製一份ST/Source/STOMS/Doc/OMSSettingSample/中的OMSSetting.xml到APIUT/bin/Debug目錄中
5. 編輯OMSSetting.xml：
   * AssemblyFile中填入新增的DLL名稱
   * CreateType中填入新增實作自IOMSService的類別完整名稱
   * SettingLoaderType中填入實作自DynaModuleSettingLoader的類別完整名稱
   * SettingFile中填入儲存所需設定值的xml檔名稱

<?xml version="1.0" encoding="big5"?>

<DataCollectionOfDynaModulePluginInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Data>

<DynaModulePluginInfo>

<AssemblyFile>KGI.TW.Der.BTOMSService.dll</AssemblyFile>

<CreateType>KGI.TW.Der.BTOMSService.BTOMSServiceAgent</CreateType>

<SettingLoaderType>KGI.TW.Der.BTOMSService.BTSettingsLoader</SettingLoaderType>

<SettingFile>BTOMSSetting.xml</SettingFile>

</DynaModulePluginInfo>

</Data>

</DataCollectionOfDynaModulePluginInfo>

1. 執行APIUT進行測試

## OrderAPIHost簡介

APIUT中的介面端主要使用的是實作自IOrderAPI的OrderAPIHost類別。

使用的流程如下：

1. 定義一個OrderAPIHost物別，並設定相關參數及接收事件
2. 連線。呼叫OrderAPIHost::Connect()
   * OrderAPIHost會開始用動態載入PlugIn的方式，由指定的要載入模組設定檔(請參考OMSSetting.xml)裏載入相關的模組
   * 載入模組的同時也會載入每個模組所指定的設定檔(也就是OMSSetting.xml裏的SettingFile屬性)
   * 載入成功後，開始呼叫每個所載入的IOMSService::Init()，並將設定檔傳送給它進行初始化的設定
   * 當設定都完成後，會呼叫IOMSService::Connect()進行連線
3. 登入。呼叫OrderAPIHost::Login()
   * 進行認證(目前還未實作，所以都可以過關)
   * 開始針對已載入且初始化成功的IOMSService呼叫其所實作的Login()函式
4. 下單
5. 登出。呼叫OrderAPIHost::Logout()
   * OrderAPIHost會呼叫已載入成功的IOMSService::Logout()
6. 離線。呼叫OrderAPIHost::DisConnect()
   * OrderAPIHost會呼叫已載入成功的IOMSService::DisConnect()