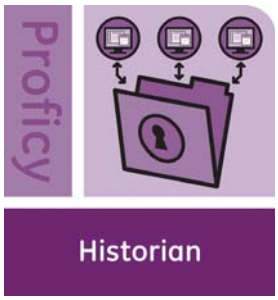

Proficy Historian 4.5 – SDK, czyli „zrób to sam”

Proficy Historian to przemysłowa baza danych obecna na polskim rynku od wielu lat. Niedawno do sprzedaży trafiła wersja 4.5, w której oprócz zwiększonej wydajności oraz ilości zbieranych danych znajdziemy nowy interfejs tworzenia aplikacji (*User API*) dla języków z platformy .NET.

Historian SDK (*Software Developement Kit*) jest biblioteką rozwojową udostępnianą przez producenta od samego początku wydawania programu. Biblioteka jest rejestrowana w systemie w modelu COM przy instalacji narzędzi Proficy Historian i może być użyta z poziomu każdego języka programowania, umożliwiającego dołączanie komponentów modelu COM. Wraz z Proficy Historian klient otrzymuje także obszerną instrukcję dotyczącą SDK oraz przykłady zastosowania, co znacznie upraszcza tworzenie własnych aplikacji. Wystarczy dołączyć odpowiednią referencję oraz w przypadku platformy .NET odpowiedni wpis do przestrzeni nazw. Dzięki temu można rozpocząć budowanie własnej aplikacji klienckiej o dowolnej funkcjonalności.

W wersji Proficy Historian 4.5 producent dołączył także API oparte na .NET w postaci przestrzeni nazw *Proficy.Historian.UserAPI*, pozwalającej na odwołanie do wszystkich funkcji SDK bez konieczności pracy z niezarządzalnym kodem. Dzięki temu dostępnych jest kilka sposobów tworzenia aplikacji w oparciu o interfejsy udostępniane przez GE Intelligent Platforms:

- Pierwszy z nich to model COM, adekwatny głównie dla języków typu VBA, VB 6.0, C++ itd. W rodzinie Visual Basic wystarczy w referencjach dodać „*Proficy Historian Software Development Kit*” z listy, aby móc rozpocząć tworzenie własnej aplikacji. W językach typu C# poza dodaniem referencji należy w sekcji *Using* dodać odpowiedni wpis „*using iHistorian_SDK;*”. Należy jednak pamiętać, że w tym wypadku wykorzystuje się tzw. kod niezarządzalny, a co za tym idzie, wiele typów danych przekazywanych jest przez tzw. *dynamic expression* na zasadzie późnego wiązania (typ danych ustalany jest podczas wykonywania, a nie kompilacji).

- Kolejną metodą jest wykorzystanie *ihuapi.cs – wrappera C#*, który wystarczy wprowadzić do projektu, a następnie dodać przestrzeń nazw na której on operuje, tj: w sekcji *Using* dodać „*using Proficy.Historian.UserAPI;*”. Rozwiązanie to jest o tyle wygodne, że nie trzeba już tworzyć własnej klasy, która będzie odpowiedzialna za współpracę z SDK. Praktycznie w modelu warstwowym aplikacja ma gotową warstwę najniższą – komunikacyjną. Zaletą jest brak konieczności załączania jakichkolwiek referencji, ponieważ biblioteka pobiera dane wprost z zarejestrowanej biblioteki *dll,* stosując *marshalling*. Dodatkowo plik kompleksowo obsługuje SDK – w przypadku samodzielnej implementacji istnieje ryzyko nie wychwycenia wszystkich przypadków powodujących wyjątki.

- Ostatnią metodą jest wykorzystanie przestrzeni nazw (dołączenie referencji i wpisu w sekcji *Using*) oraz samodzielne napisanie biblioteki obsługującej połączenie.

Obok przedstawiono kod wykorzystujący *ihuapi.cs* do połączenia i pobrania serii danych z archiwum Proficy Historian. Podobny przykład znaleźć można na DVD Proficy Historian. W celu uzyskania dodatkowych informacji – prosimy o kontakt *vix@vix.com.pl*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Proficy.Historian.UserAPI;

namespace VIX
{
  class ihCWrapperTest
  {
    static void ReadRawDataByTime(int handle, string tag)
    {
      IHU_TIMESTAMP start = new IHU_TIMESTAMP(
        DateTime.Now.AddMinutes(-1));

      IHU_TIMESTAMP end = new IHU_TIMESTAMP(
        DateTime.Now);

      IHU_DATA_SAMPLE[] values;

      Console.WriteLine(Environment.NewLine+"Czytanie
        metodą RawDataByTime");

      ihuErrorCode result = IHUAPI.ihuReadRawDataByTime(
        handle, tag, start, end, out values);

      Console.WriteLine("{0} wartości={1} [{2}]", tag,
        values == null ? 0 : values.Length, result);

      foreach (IHU_DATA_SAMPLE sample in values)
      {
        Console.WriteLine("{0} {1} wartość={2}",
          sample.TimeStamp.ToDateTime(),sample.Quality,
          sample.ValueObject);
      }
    }

    static void Main(string[] args)
    {
      int handle;
      string tag = "HISTORIAN_PC.Simulation00001";
      ihuErrorCode result = IHUAPI.ihuConnect("", "", "",
        out handle);

      Console.WriteLine();
      Console.WriteLine("Połączono={0} handle={1}",
        result, handle);

      Console.WriteLine("ihuIsServerConnected({0})={1}",
        handle, IHUAPI.ihuIsServerConnected(handle));

      if (result == ihuErrorCode.OK)
      {
        Console.WriteLine("Rozpocząć czytanie Tagów?");
        Console.ReadKey();
        ReadRawDataByTime(handle, tag);
        Console.WriteLine("Rozłączyć?");
        Console.ReadKey();
        result = IHUAPI.ihuDisconnect(handle);
        Console.WriteLine("Rozłączono={0}", result);
      }
    }
  }
}
```

Tomasz Duda
Inżynier Wsparcia Technicznego
VIX Automation sp. z o.o