OIF C#编程入门

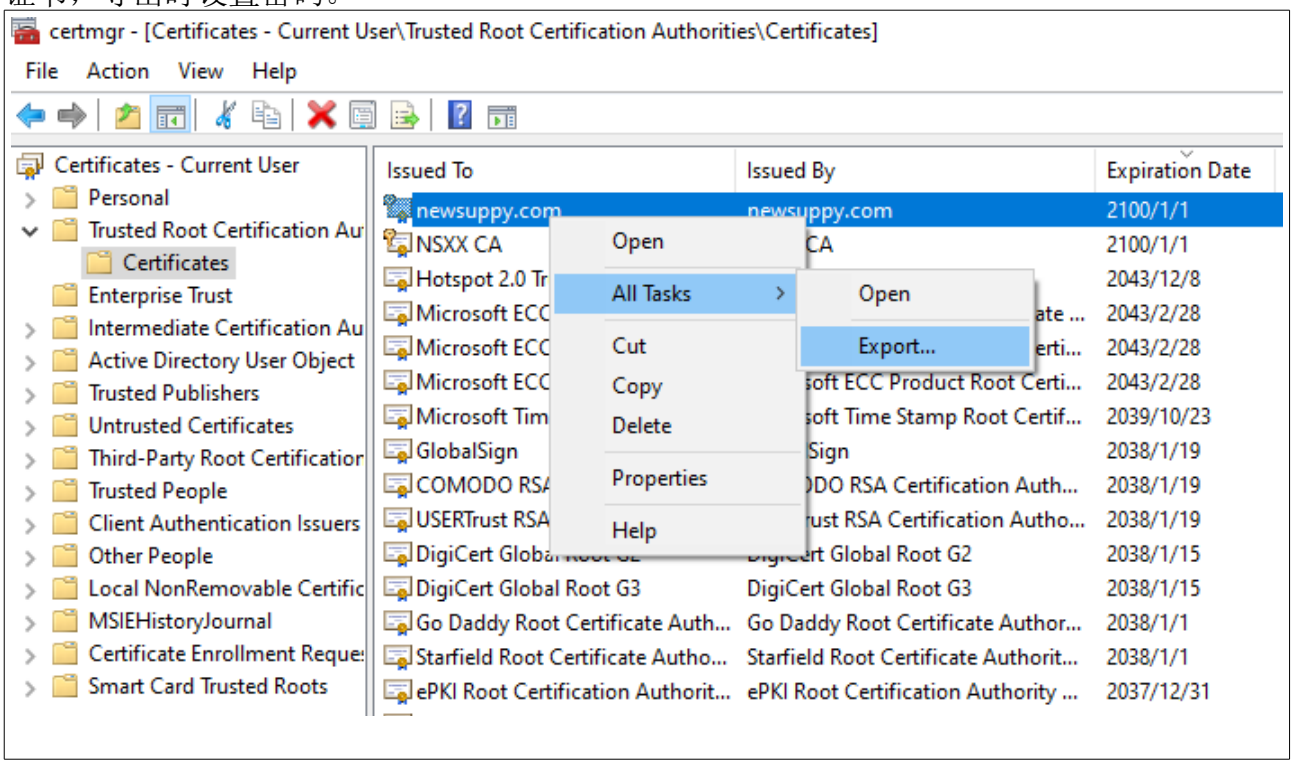
ZW Mar 8, 2022
v1.0

1，建立 OIF 的使用证书。

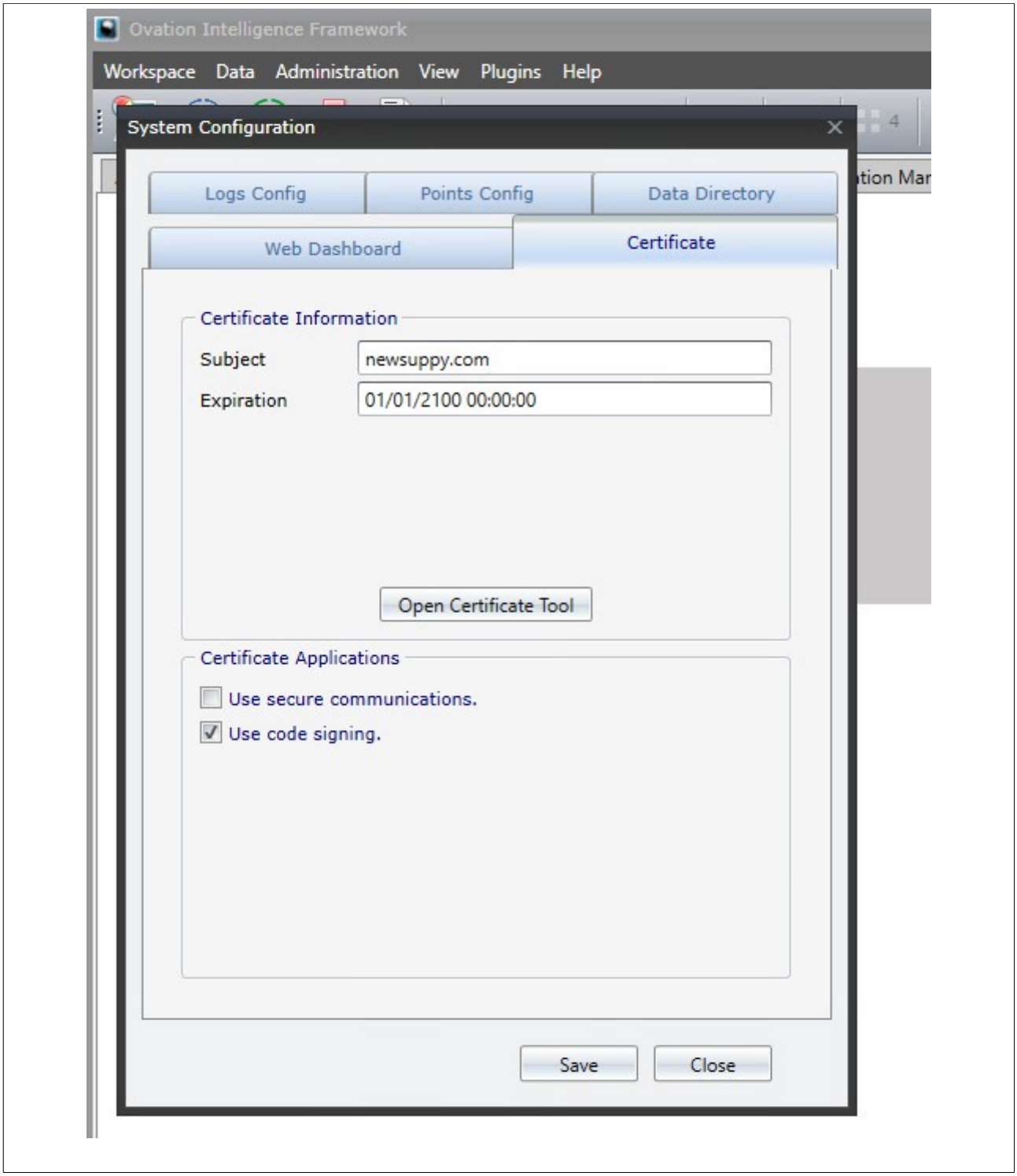
OIF 需要首先导入一个 pfx 的证书才能使用，我们可以自己创建一张证书。
在 Windows 上，可以用 makecert 先建立一张 cert 证书，用如下的命令：

```
makecert -r -pe -n "CN=newsuppy.com" -$ commercial -a sha1 -b 01/01/2020 -e 01/01/2100 -cy authority -ss root -sr currentuser
```

注意 makecert 已经过时，未来使用 New-SelfSignedCertificate 命令创建。
然后在 Windows 的 Run 中启动 certmgr.msc 证书管理程序。找到刚才新加的证书，并导出 pfx 证书，导出时设置密码。

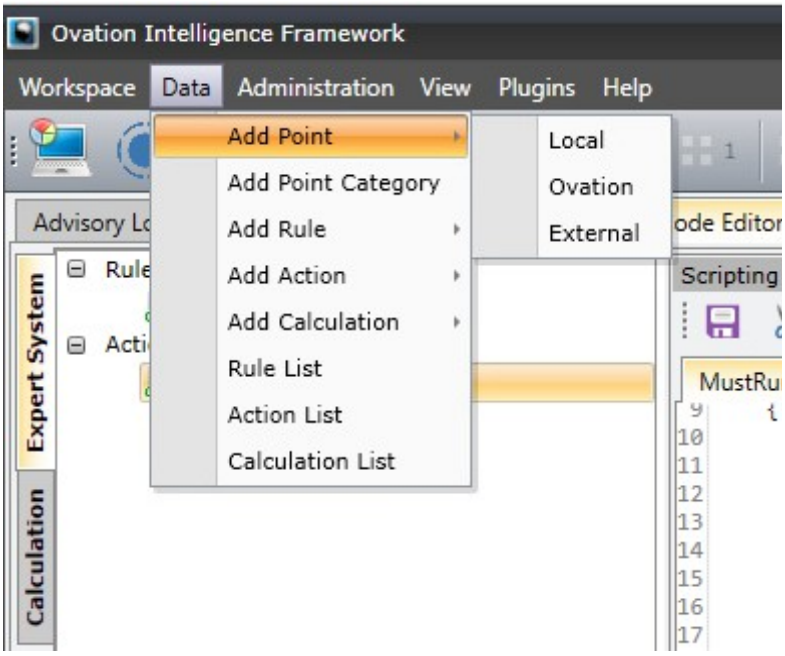


接下来将导出的 pfx 证书通过 OIF 的菜单中 Administration->System Configuration->Ceritifcate->Open Certificate Tool 导入证书。证书导入成功后就可以正常使用 OIF 了。

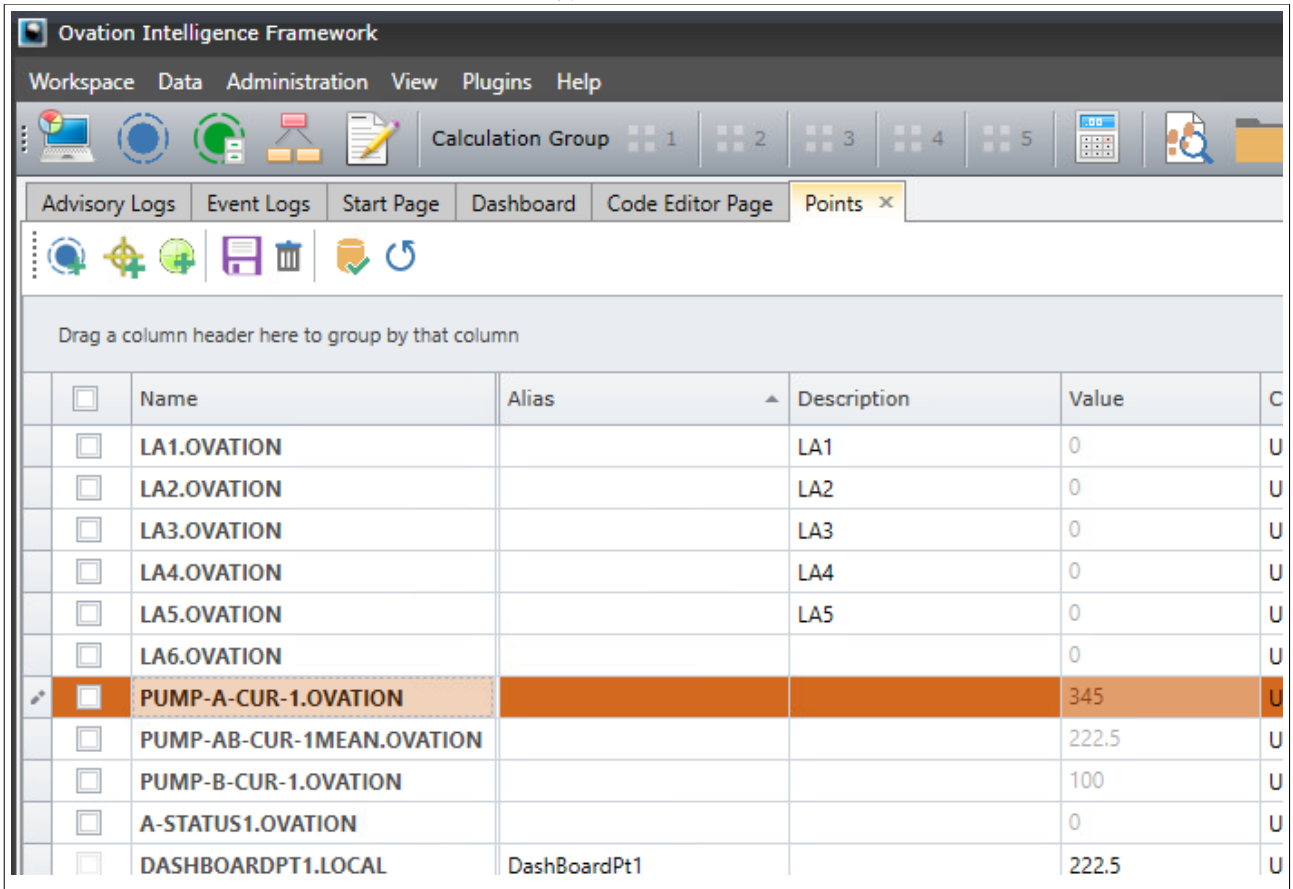


2，OIF 添加点

如下图可以添加三种类型的点，Local 只存在于 OIF 软件内部。这里我们先在 Ovation Studio 中添加 3 个 LA 点(先建在 OIF 站上)，PUMP-A-CUR-1/PUMP-B-CUR-1/PUMP-AB-CUR-1MEAN。然后在下图的 OIF 菜单 Data->Add Point->Ovation 添加到 OIF 点表中。

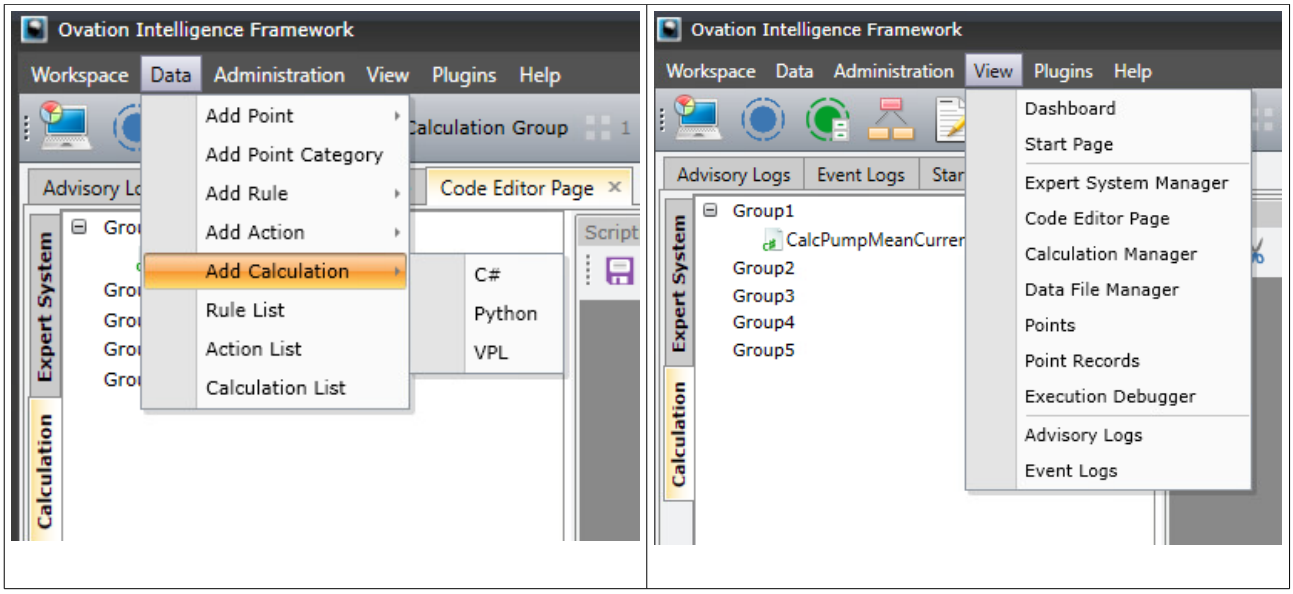


点击 OIF 工具栏的蓝色圆圈图标就可以看到 OIF 系统的点表了，如下图：

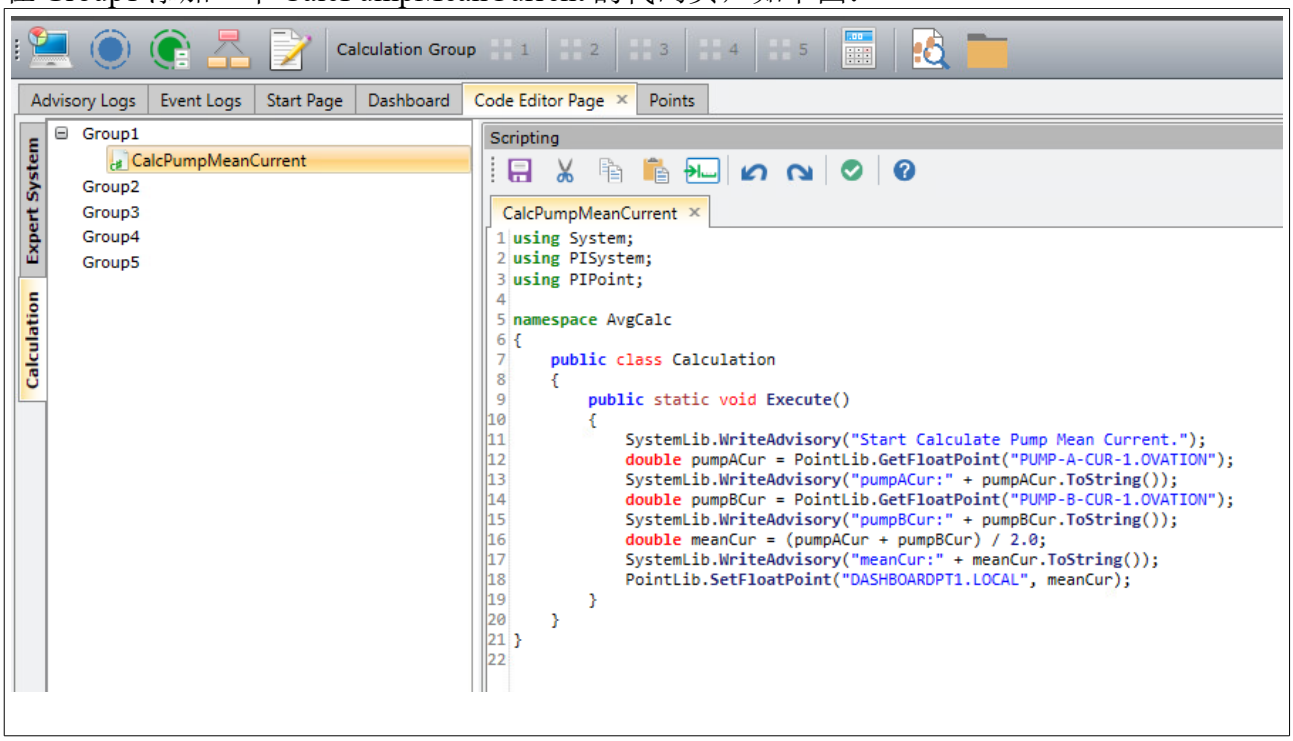


3，OIF 计算组（Calculation）

OIF 计算组是 OIF 的一种简易型编程计算工具，我们可以通过菜单 Data->Add Calculation 或者 View->Code Editor Page->Calculation 添加代码。



在 Group1 添加一个 CalcPumpMeanCurrent 的代码页，如下图：



实际的代码是这样的：

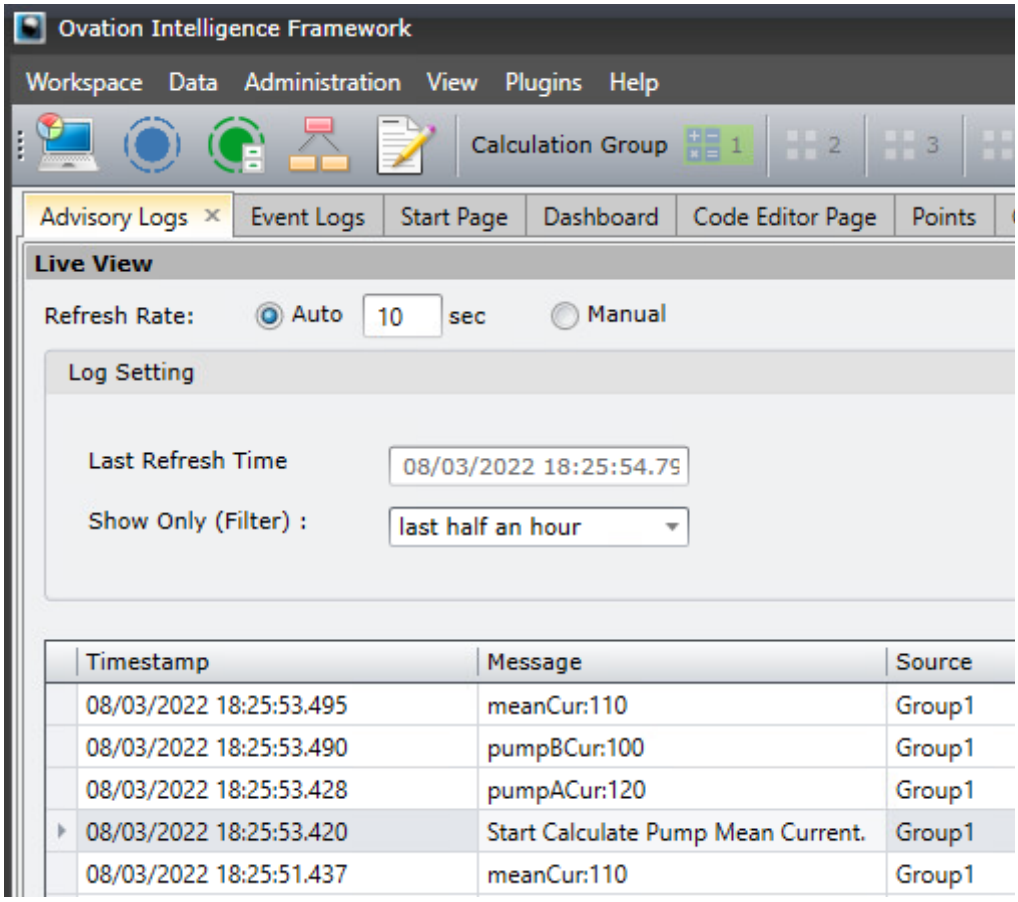
```
using System;
using PISystem;
using PIPoint;

namespace AvgCalc
{
    public class Calculation
    {
        public static void Execute()
        {
            SystemLib.WriteAdvisory("Start Calculate Pump Mean Current.");
            double pumpACur = PointLib.GetFloatPoint("PUMP-A-CUR-1.OVATION");
            SystemLib.WriteAdvisory("pumpACur:" + pumpACur.ToString());
            double pumpBCur = PointLib.GetFloatPoint("PUMP-B-CUR-1.OVATION");
            SystemLib.WriteAdvisory("pumpBCur:" + pumpBCur.ToString());
            double meanCur = (pumpACur + pumpBCur) / 2.0;
```

```
            SystemLib.WriteAdvisory("meanCur:" + meanCur.ToString());
            PointLib.SetFloatPoint("PUMP-AB-CUR-1MEAN.OVATION", meanCur);
        }
    }
}
```

这里简单解析一下上面的 C#代码，文件开头的 using 是用来导入程序库的，这个代码页导入了 System/PISystem/PIPoint 共 3 个程序库。在 C#中基本的单位是 Class 类，这里定义了一个 Calculation 类，这个类仅有一个函数 Execute（注意 C#代码是区分大小写的）。类是用户定义的数据类型，它可以包含属性和方法。比如定义一个 Dog 类，那么它有属性：眼睛、尾巴，腿；它有方法：奔跑，咬人。我们定义的 Calculation 类没用属性，只有一个方法 Execute。这个 Execute 方法也叫做类的成员函数，这个函数没有参数，返回值为 void，即没用返回值。它的属性是 public 和 static。public 说明是一个公开可以访问的函数（即其他类可以调用这个函数），static 说明是一个静态函数（调用这个函数不需要为类生成一个实例）。这个函数也是建立程序后自动生成的，必须存在的启动函数。在 Calculation 类的外层还有 namespace AvgCalc，namespace 是一种区分名称的方式，比如你的锅炉系统和汽机系统都有一个类叫 Calculation，你不想把锅炉的叫 BoilerCalculation，汽机的叫 TurbineCalculation，那么可以建立 Boiler 的 namespace 命名空间和 Turbine 的 namespace，在这两个 namespace 中都可以定义 Calculation 类。

在 Execute 函数体内，就是实际的执行代码。第 1 行代码调用 SystemLib 类的 WriteAdvisory 写一条信息。SystemLib 类属于 PISystem 库，我们可以将第 1 行写成 PISystem.SystemLib.WriteAdvisory，不过因为我们前面已经写了 using PISystem，就可以直接调用 SystemLib，而 WriteAdvisory 会在程序运行时输出日志给 Advisory Logs。

我们可以通过 OIF 的 View->Advisory Logs 查看 Advisory 日志，如下图（当然现在程序还没有运行呢）：



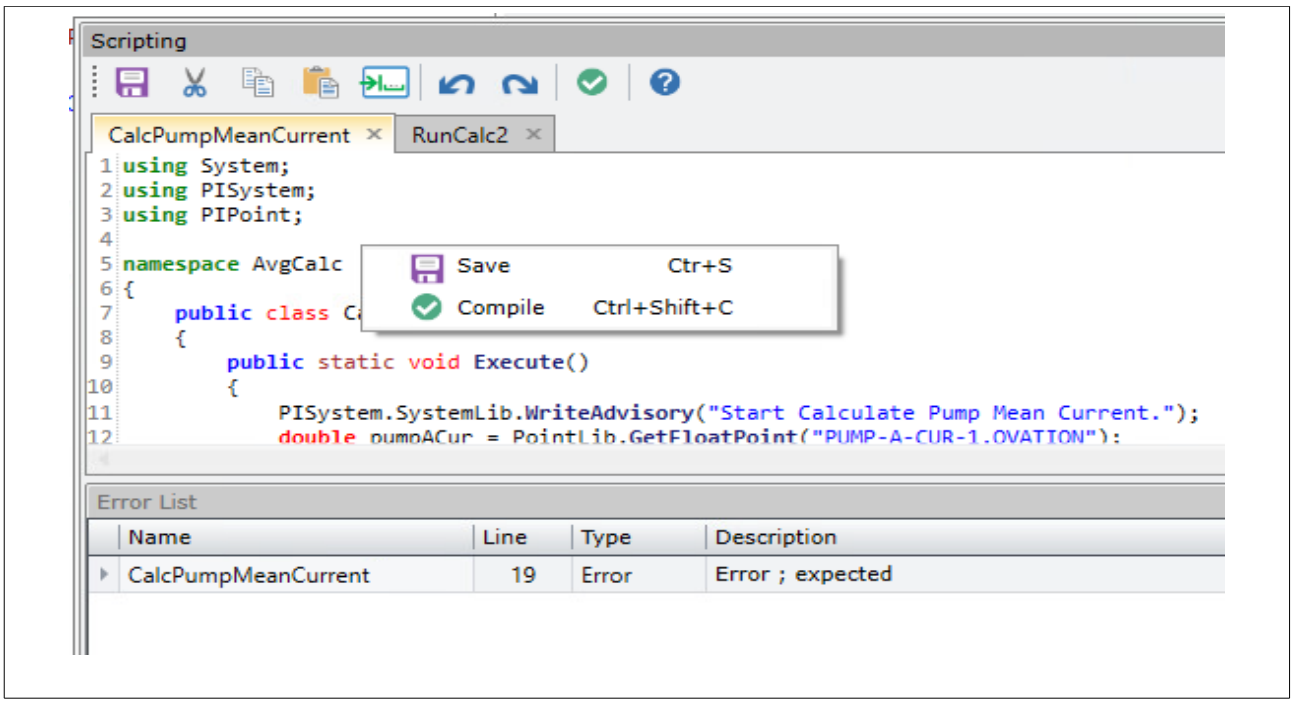
接下来第 2 行调用 PIPoint 库的 PointLib 类的 GetFloatPoint 函数，这个函数只有一个参数，即点名，点名需要用双引号括起来，函数执行后的返回值是一个浮点数，将这个数值赋值给 double 类型的 pumpACur 变量。在 C#中 double 就是一种浮点数类型。这里的代码:

```
double pumpACur = PointLib.GetFloatPoint("PUMP-A-CUR-1.OVATION");
```

将 double 类型实例化，这个实例化的 pumpACur 叫做对象(Object)。并且给 pumpACur 赋值为 Ovation 点 PUMP-A-CUR-1 的数值。接下来我们又将 pumpACur 的数值写入 Advisory 日志，这里调用了 double 类型的 ToString 函数，这是因为 WriteAdvisory 只接受字符串 string 参数，而 pumpACur 是 double 浮点数类型，用 ToString 函数将 double 类型转换为 string 类型后传给 Advisory。注意 C#的每行代码后用';'结束。

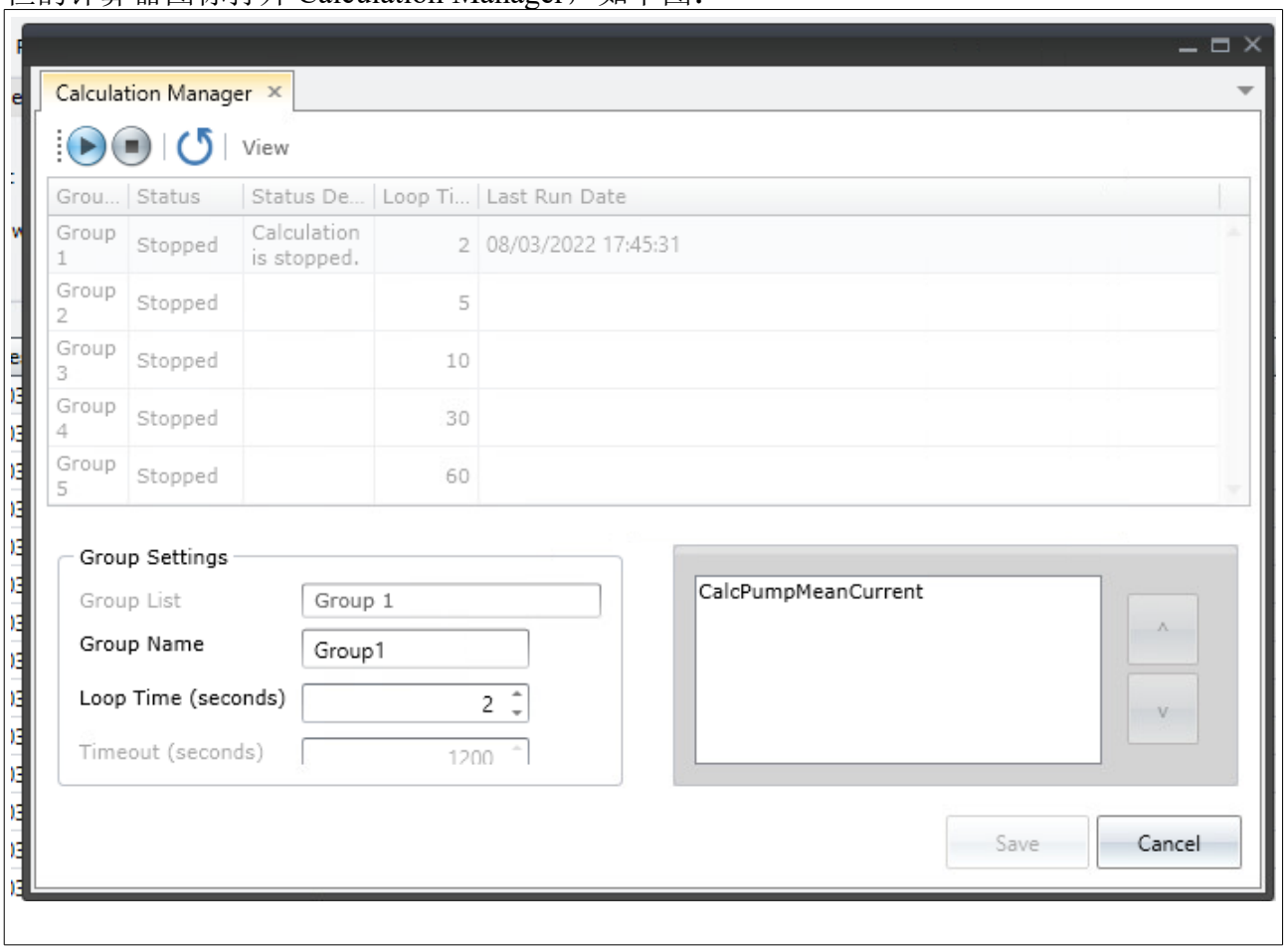
接下来的代码就是计算两个泵电流的平均值，计算结果赋值给 meanCur。最后使用 PointLib 的 SetFloatPoint 将 meanCur 的数值写给点 PUMP-AB-CUR-1MEAN。

代码完成后需要 Save 和 Compile，可以使用 Code Editor 的菜单栏，也可以直接使用鼠标右键的菜单，如下图：



如果程序有错误，下方的 ErrorList 会提示错误代码在哪一行，并给出错误原因。

保存编程完成后，就可以开始执行程序了。在 OIF 菜单 View->Calculation Manager 或者工具栏的计算器图标打开 Calculation Manager，如下图：

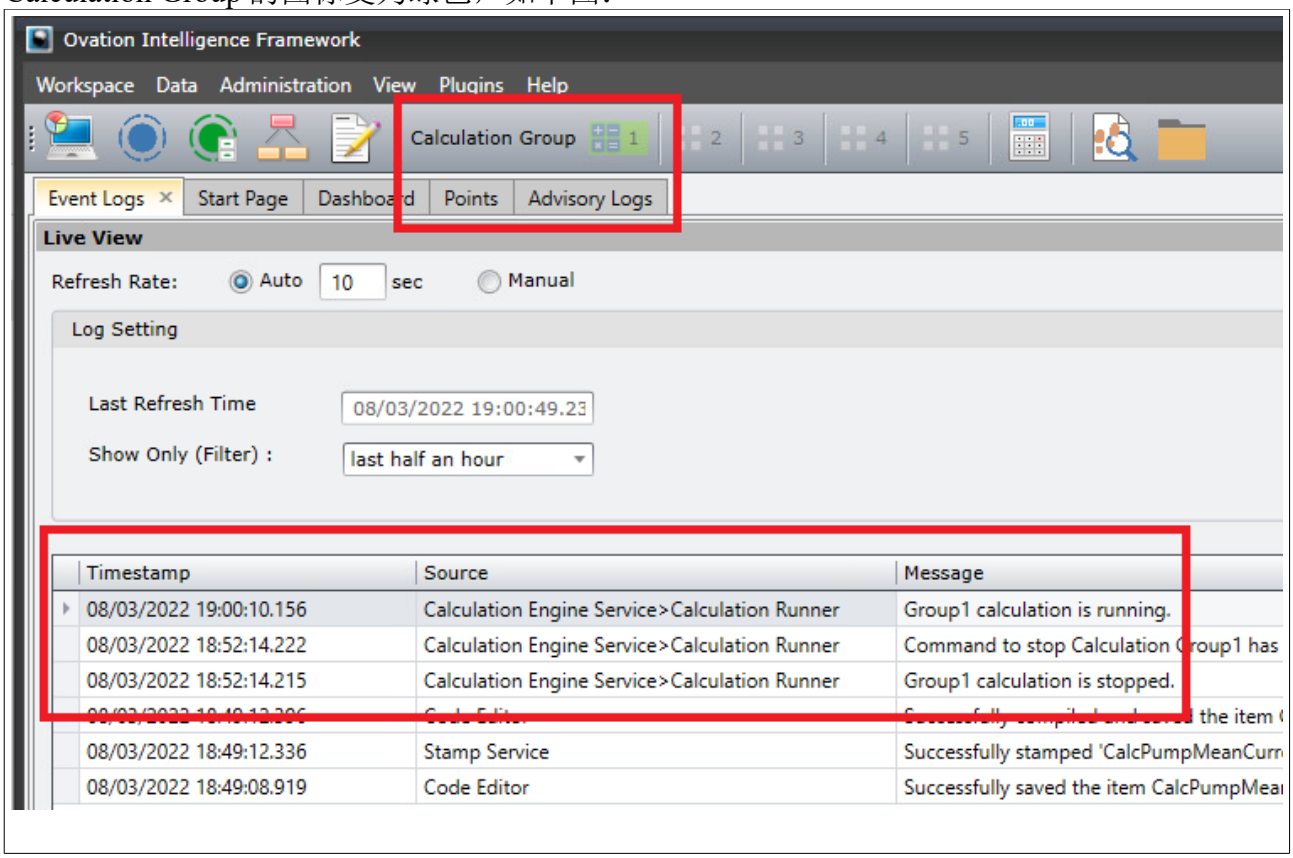


Calculation Manager 中可以看到系统定义了 5 个 Group 计算组，每个组的执行时间不同，在停止状态下，点击 View 按钮后，在下方出现的 Edit 可以修改 Group 的名字和 Loop Time。

点击播放按钮就可以直接执行程序了。

程序成功运行后，我们可以在 Event Logs 看到程序启动的日志，并且在工具栏可以看到相应

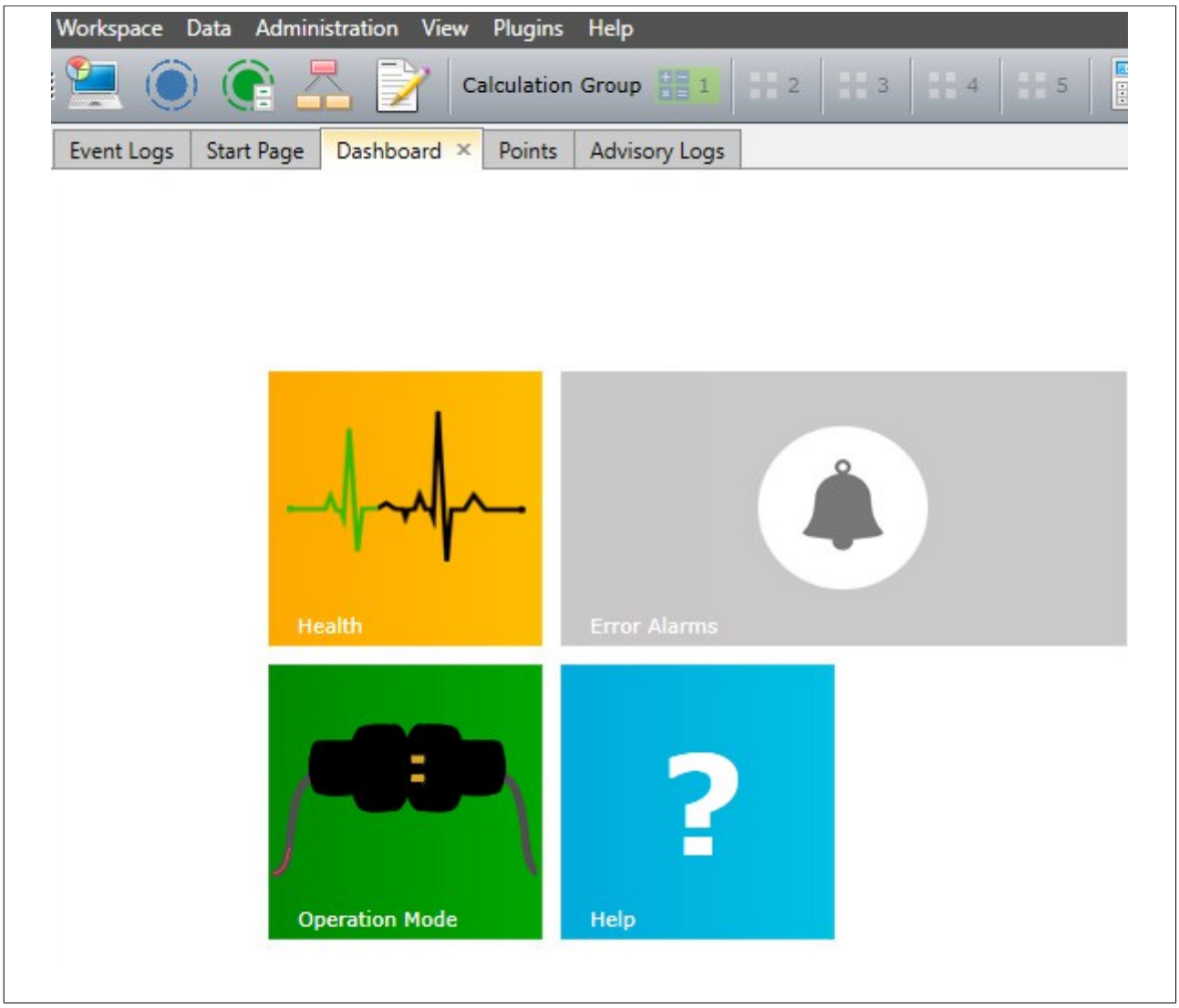
Calculation Group 的图标变为绿色，如下图：



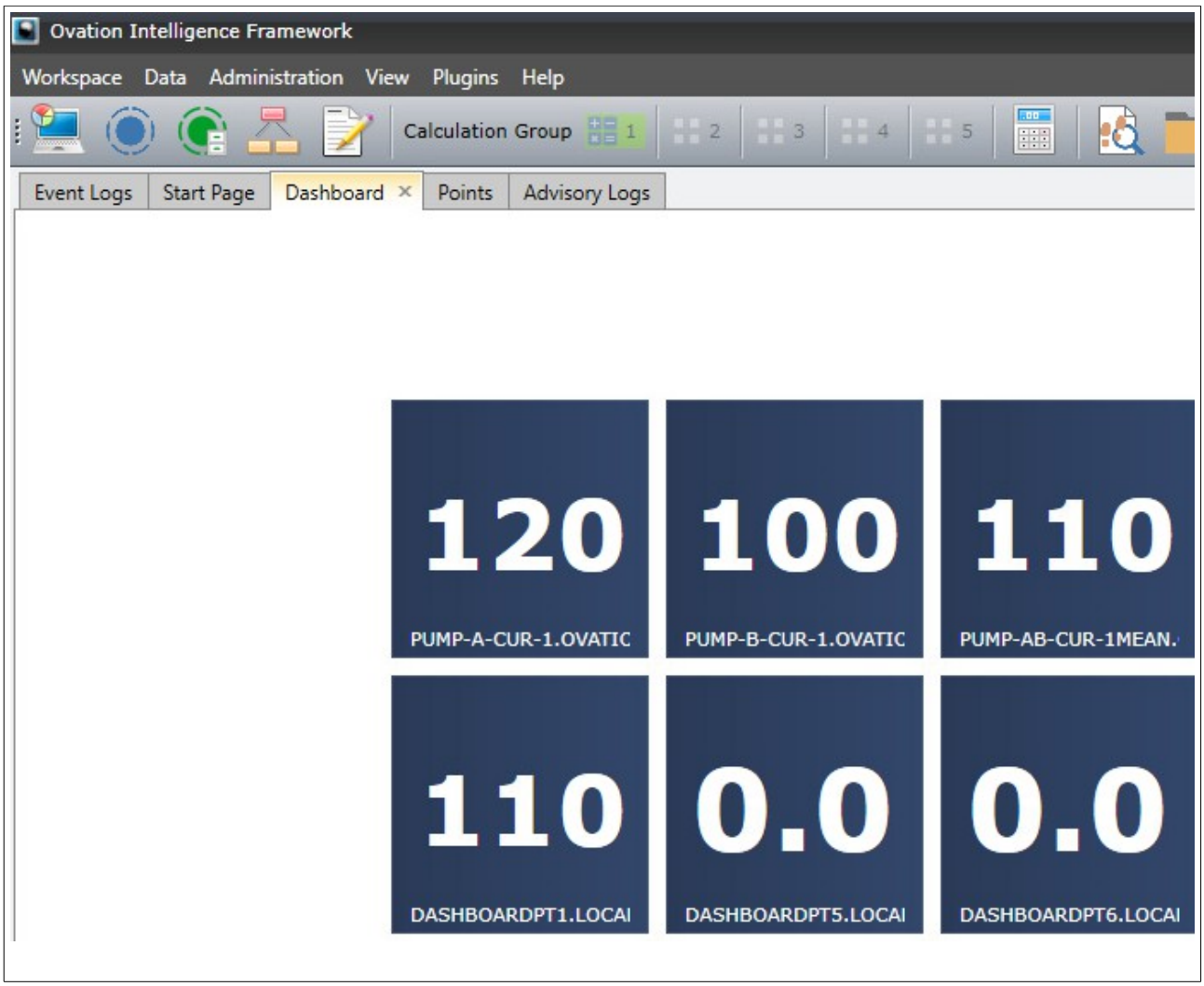
OIF 的运行状态

通过 OIF 菜单 View->Dashboard 或者工具栏的笔记本电脑图标可以进入 Dashboard 窗口。

Dashboard 窗口有两个页面，通过侧边的箭头切换。第一个页面如下图：



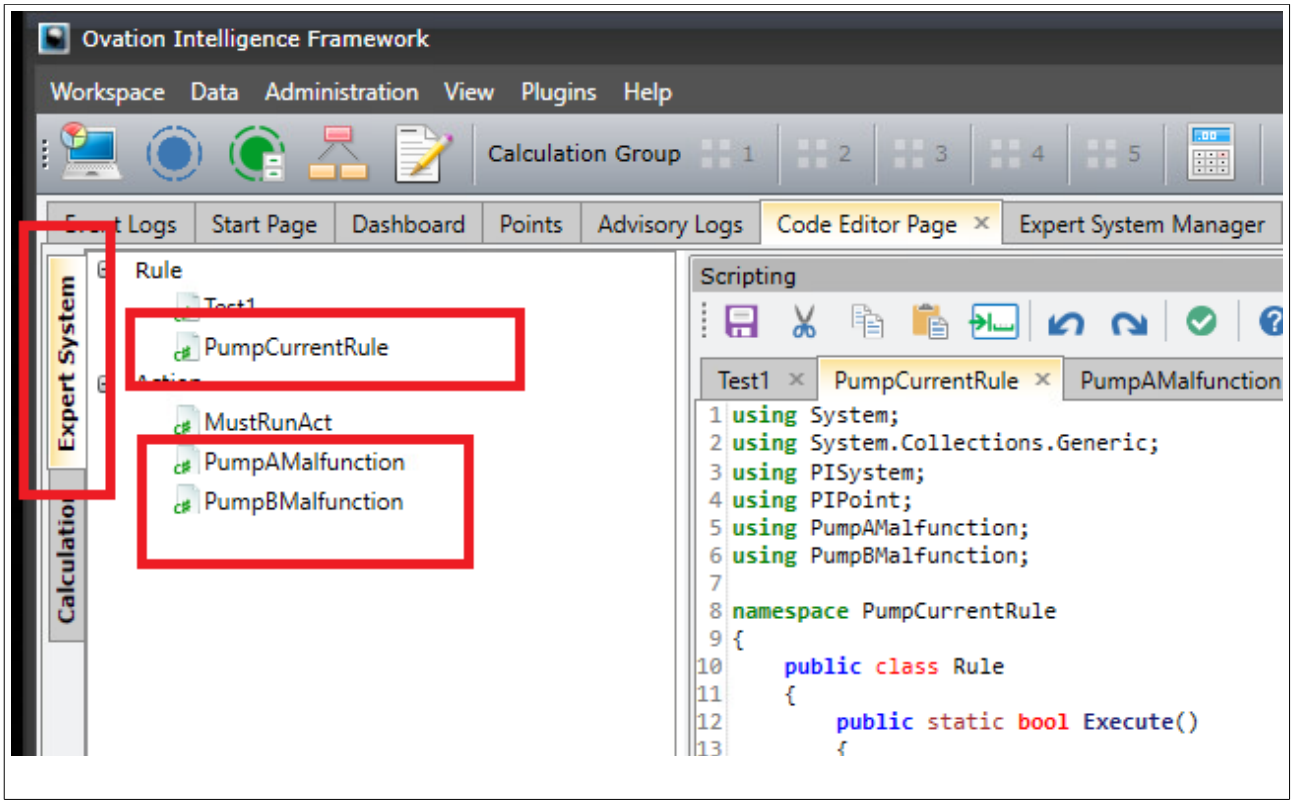
这个页面的 Operation Mode 是需要我们注意的，这里有个插头图标，默认情况下，插头是断开的（Disconnected Mode)，点击一下就可以让插头连接，并进入 Connected Mode。如果要通过 OIF 给 Ovation/External 的点写入数值，需要进入 Connected Mode。否则只能读取 Ovation 点的数值。而对于 OIF 的 Local 点，任何情况都可以读写。
切换到下一个界面，可以在 Dashboard 观察 6 个点的状态，最多只有 6 个点，无法增减，双击模块可以修改显示的点名。



4，OIF 专家系统（Expert System）

这次我们建立一个非常简单的泵健康监测专家系统。先在 Ovation Studio 中再增加 PUMP-A-FAULT/PUMP-B-FAULT 两个 LD 点，然后再导入 OIF。

打开 OIF 的 Code Editor，如下图：



选择 Expert System，在 Rule 中建立 PumpCurrentRule 的代码页，在 Action 建立 PumpAMalfunction 和 PumpBMalFucntion。在 Rule 里面，我们计算出 A,B 泵的平均电流，然后判断 A 或 B 泵的电流是否大于 1.5 倍的平均电流，如果大于，就判断泵故障。

代码如下：

```
using System;
using System.Collections.Generic;
using PIPoint;

namespace PumpAMalfunction
{
    public class Action
    {
        public static void Execute()
        {
            PointLib.SetBoolPoint("PUMP-A-FAULT.OVATION", true);
        }
    }
}
```

PumpAMalfunction.cs 代码页

```
using System;
using System.Collections.Generic;
using PIPoint;

namespace PumpBMalfunction
{
    public class Action
    {
        public static void Execute()
        {
            PointLib.SetBoolPoint("PUMP-B-FAULT.OVATION", true);
        }
    }
}
```

PumpBMalfunction.cs 代码页

```
using System;
using System.Collections.Generic;
using PISystem;
using PIPoint;
using PumpAMalfunction;
using PumpBMalfunction;

namespace PumpCurrentRule
{
    public class Rule
    {
        public static bool Execute()
        {
            //SystemLib.WriteAdvisory("Start PumpCurrentRule");
            double pumpACur = PointLib.GetFloatPoint("PUMP-A-CUR-1.OVATION");
            double pumpBCur = PointLib.GetFloatPoint("PUMP-B-CUR-1.OVATION");
            double meanCur = (pumpACur + pumpBCur) / 2.0;
            PointLib.SetFloatPoint("PUMP-AB-CUR-1MEAN.OVATION", meanCur);
            if (pumpACur > 1.5 * meanCur)
            {
                SystemLib.WriteAdvisory("Excute PumpAMalfunction Action");
                PumpAMalfunction.Action.Execute();
            }
            else
            {
                PointLib.SetBoolPoint("PUMP-A-FAULT.OVATION", false);
            }
            if (pumpBCur > 1.5 * meanCur)
            {
                SystemLib.WriteAdvisory("Excute PumpBMalfunction Action");
                PumpBMalfunction.Action.Execute();
            }
            else
            {
                PointLib.SetBoolPoint("PUMP-B-FAULT.OVATION", false);
            }

            return true;
        }
    }
}
```

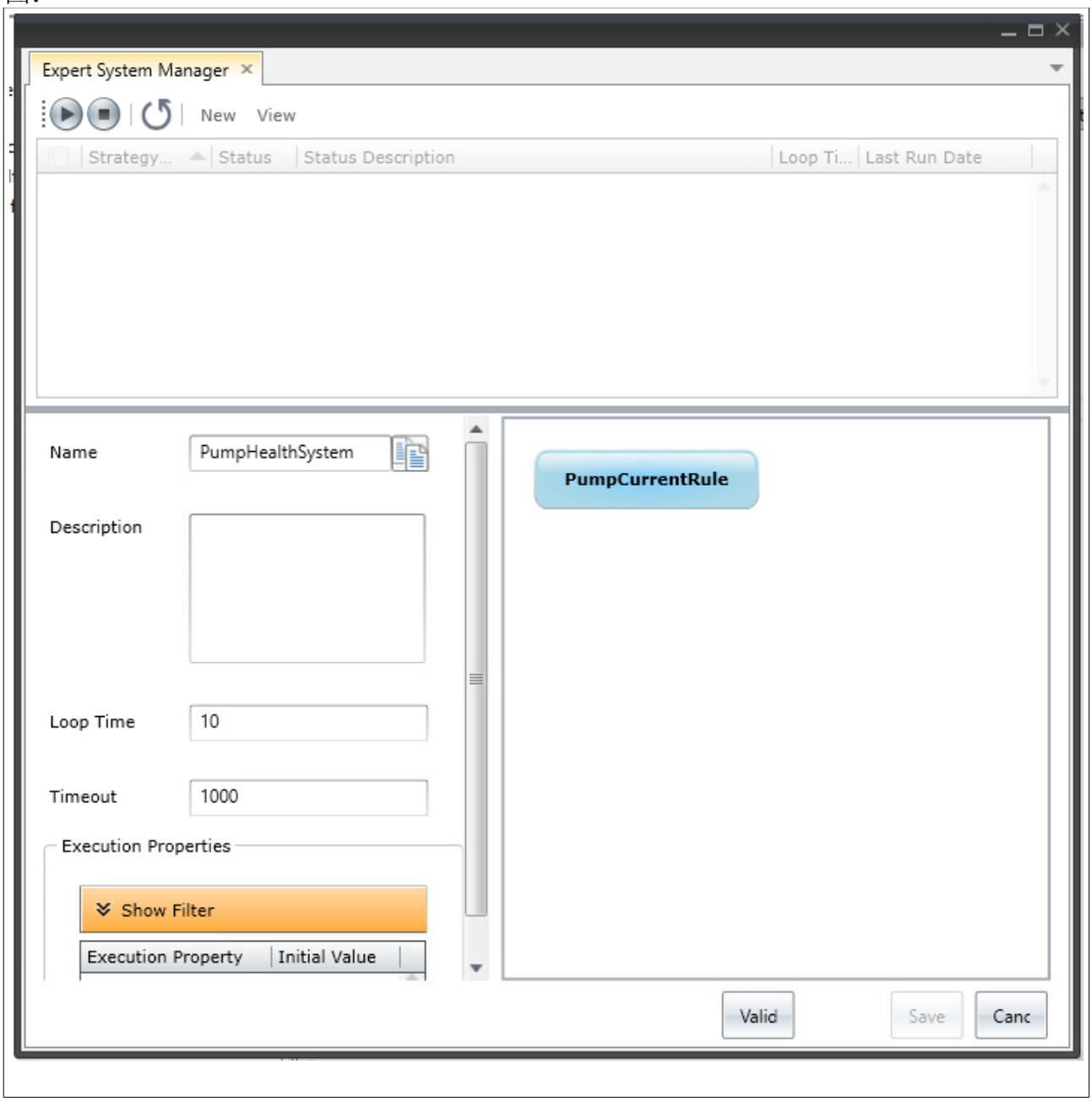
PumpCurrentRule.cs 代码页

这里 Action 的代码很简单，就是在 Execute 函数内将泵对应的 FAULT 点置为 1(true)。主要的判断逻辑在 PumpCurrentRule 的代码中，Rule 的 Execute 与之前的不一样，这个函数会返回一个 bool 类型，即 true 或者 false。在专家系统中，Rule 与 Rule 之间用判据连接起来，比如 Rule1 返回 true 则执行 Rule2，如果返回 false 则执行 Rule3。在本例中，只有一个 Rule，因此我们在函数的结尾直接 return true。在本例的 Rule 中，先获取 A 泵电流，然后是获取 B 泵电流，计算平均值后，将平均值电流回写给 Ovation 的平均值电流点。然后使用 if/else 语句。在 if 语句的括号中，判断泵的电流是否大于 1.5 倍的平均电流，如果是，执行 if 后大括号内的语句，输出一个 Advisory 以及执行相应泵的故障 Action。如果否，则直接将泵的 FAULT 点置为 0(false)，即复位故障。

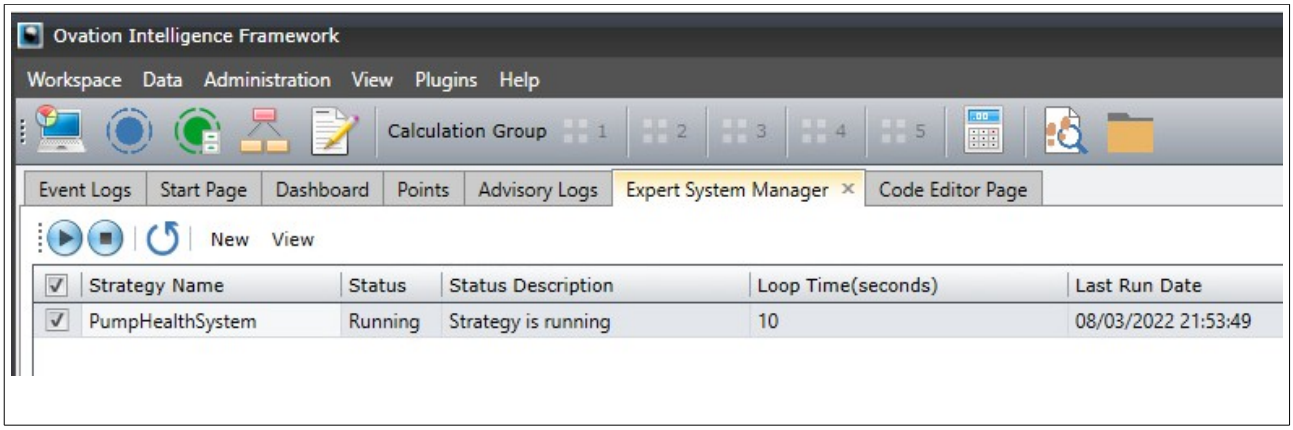
代码编写完成后，和计算组一样，将代码 Save 保存后，再 Compile 编译。如果没有任何问题，那么我们开始建立自己的专家系统。

在菜单 View->Expert System Manager 或者工具栏的相应图标，进入专家系统管理界面，如下

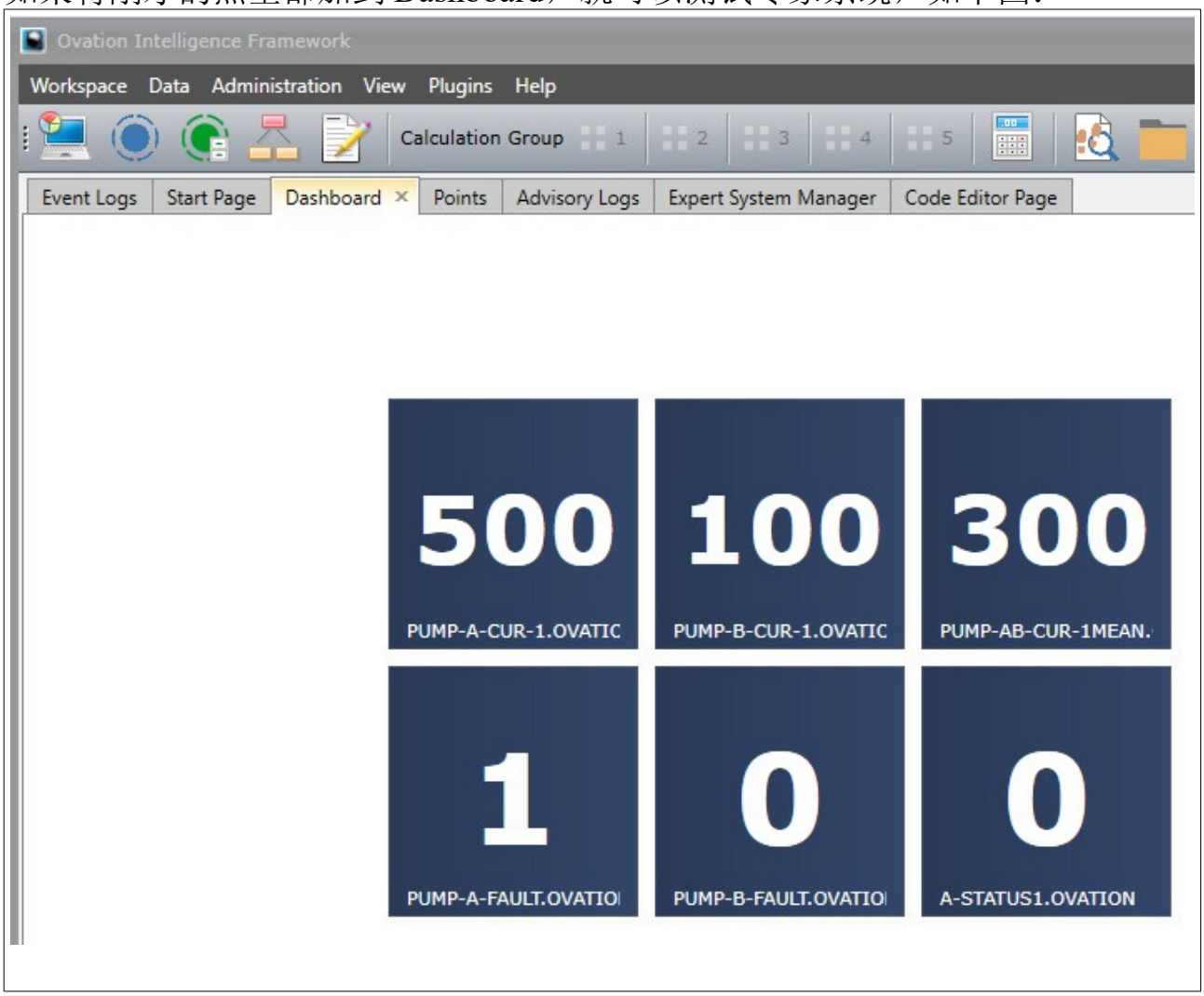
图：



如果有之前的专家系统，需要停止才可以管理，使用 New 按钮新建专家系统，View 按钮则可以编辑选择的专家系统。我们现在 New 一个 PumpHealthSystem，设置 Loop Time 为 10 秒(最大 120 秒)，Timeout 为 1000 秒（最大 1000 秒），然后双击右侧蓝色的 Rule 按钮，选择刚才我们编译成功的 PumpCurrentRule。然后按下面的 Valid 按钮，成功后 Save。专家系统建立成功后，可以看到列表里出现了刚才新建的 PumpHealthSystem，选中后，按播放按钮就可以启动专家系统，如下图：

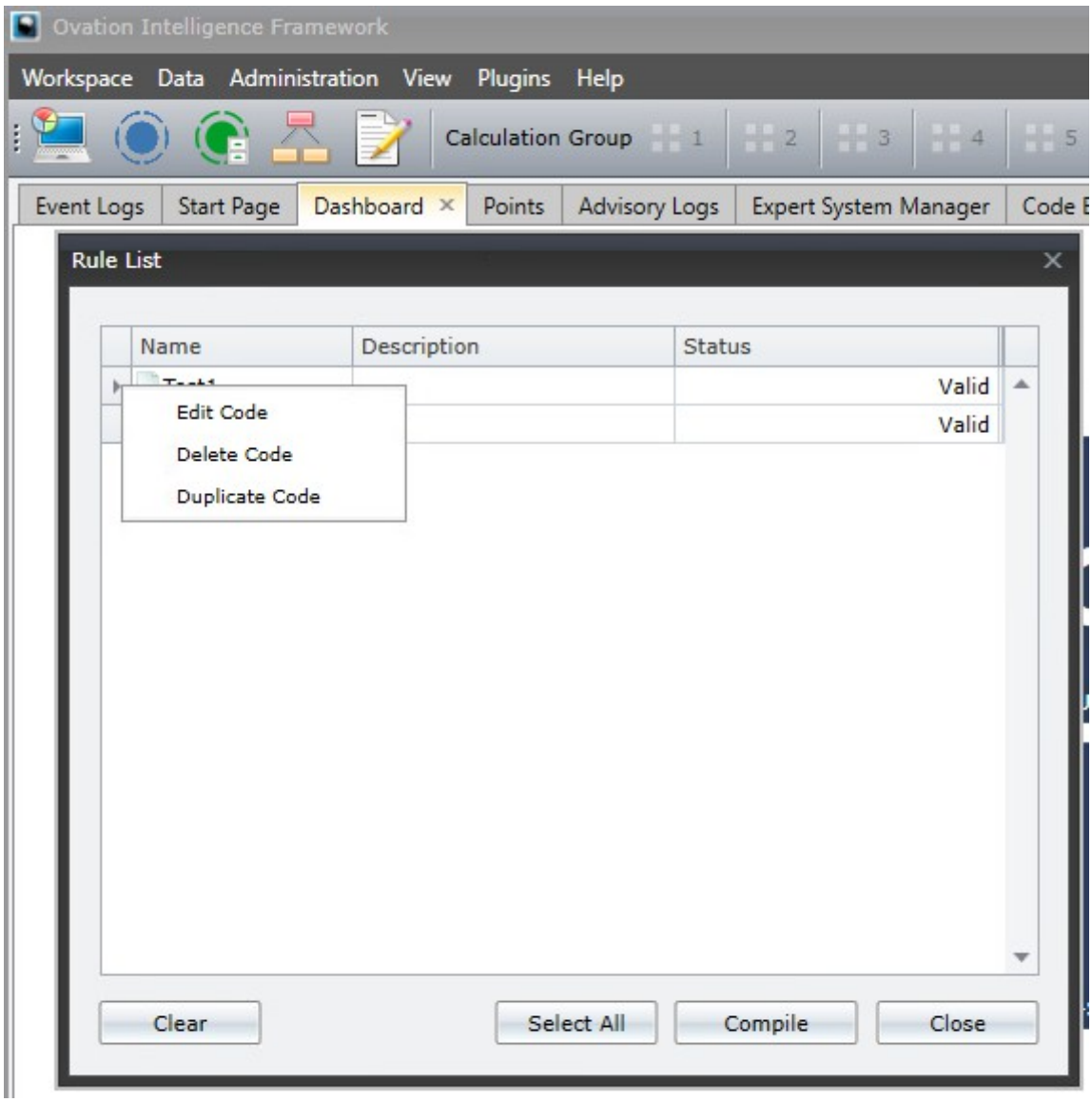


如果将刚才的点全部加到 Dashboard，就可以测试专家系统，如下图：



5，其他

（1）如果要删除代码，需要通过菜单 Data 进入代码的 List，比如删除不需要的 Rule，就进入 Rule List，如下图：



只有在这里才能删除代码，并且 List 会显示代码的状态 status，只有 Valid 的代码才能够被成功执行。