

自定义事件提供程序示例

*本示例适用于 2016 年 3 月 Xbox One XDK 或更高版本。*

# 描述 本示例演示如何在 Xbox One 上使用自定义 ETW 事件提供程序。 使用样本

本示例使用以下控件：

|  |  |
| --- | --- |
| 操作 | 游戏手柄 |
| 退出样品。 | 左侧扳机键 + 右侧扳机键 + 右肩部键 |

实施说明  
本示例采用传统 Windows ETW 提供程序的结构，但 Exclusive 分区中运行的标题无法将其事件提供程序添加至注册表，因此需要在主机上执行一些其他步骤才能正确解析生成的事件数据。

可手动编辑事件清单 **etwprovider.man**，也可使用 Windows SDK 随附的 ecmangen.exe 工具进行编辑。它通过 Visual Studio 消息编译器 (mc.exe) 被编译为资源文件 (**etwproviderGenerated.rc**) 和头 (**etwproviderGenerated.h**)。资源文件和头都包含在标题项目中。

在标题初始化期间，事件提供程序通过调用 **EventRegisterCEP\_Main** 进行注册，然后在关闭处理期间通过调用 **EventUnregisterCEP\_Main** 取消注册。“Mark”事件通过调用 **EventWriteMark** 发出，后者使用一个 Unicode 字符串作为参数。

本示例可通过与任何其他示例相同的方法进行构建、部署和激活。运行后，立即使用 tracelog 捕获事件（xbperf 无法指定自定义事件提供程序）。标题的事件提供程序尚未添加到注册表中，因此需要通过 GUID 而不是通过名称识别（请注意，GUID 必须与事件清单中指定的提供程序 GUID 匹配）：

C:\temp>xbrun /x/title /O tracelog -start CustomSession -f d:\custom03.etl -eflag PROC\_THREAD+LOADER+DPC+INTERRUPT+CSWITCH+PROFILE -guid #{A4A76336-4BA7-4CD9-85C3-B9C236D3041C} -stackwalk PROFILE+CSWITCH

所需数据被抓取后，会话就可以通过常规方式终止：

C:\temp>xbrun /x/title /O tracelog -stop CustomSession

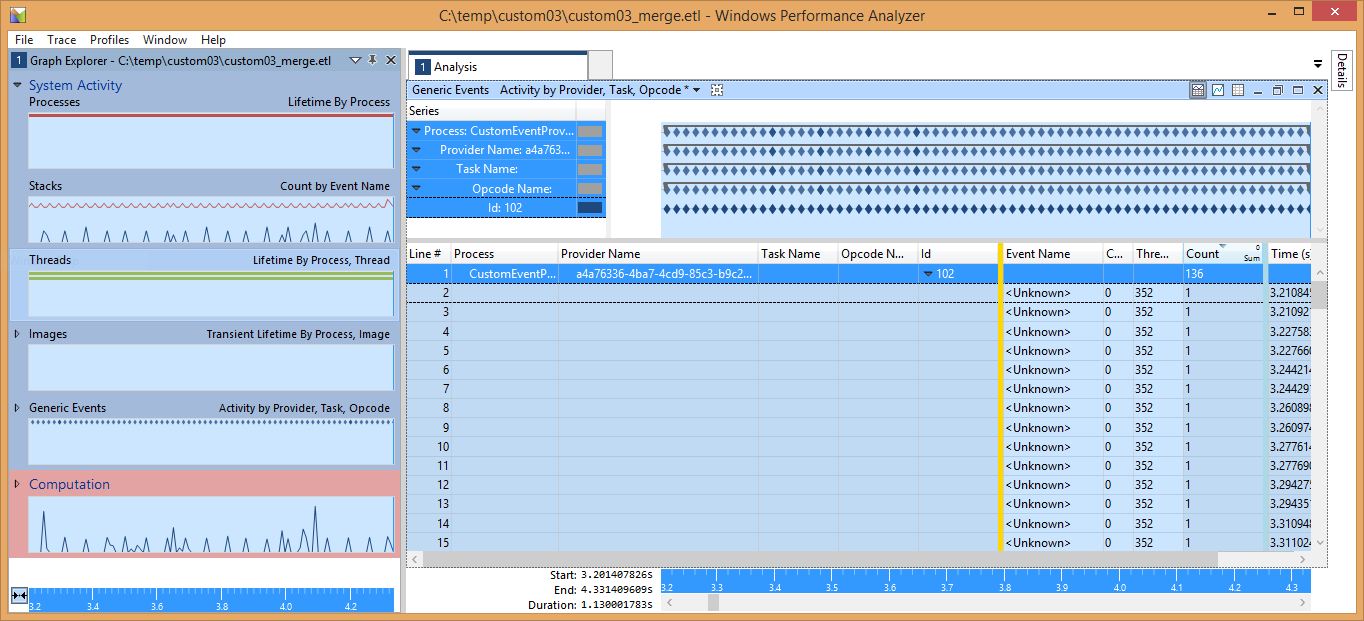
合并 devkit 上的 ETL 文件以解析系统事件提供程序。这将*不会*解析我们的自定义事件提供程序：

C:\temp>xbrun /x/title /O tracelog -merge d:\custom03.etl d:\custom03\_merge.etl

现在可将合并的文件复制回主机：

C:\temp\custom03>xbcp /x/title xd:\custom03\_merge.etl .

此文件可以加载到 WPA，然后自定义事件将出现在“系统活动”(System Activity) 组中的“通用事件”(Generic Events) 图中。但是，此时，该事件将仅通过 GUID 识别，诸如任务名称和操作代码名称之类的信息将不可见。更重要的是，也不会显示我们为每个事件提供的自定义数据（Unicode 字符串）。



为显示每个自定义事件的完整信息，我们使用了一个技巧 - *我们在主机上而不是 devkit 上注册事件提供程序和解析事件。*

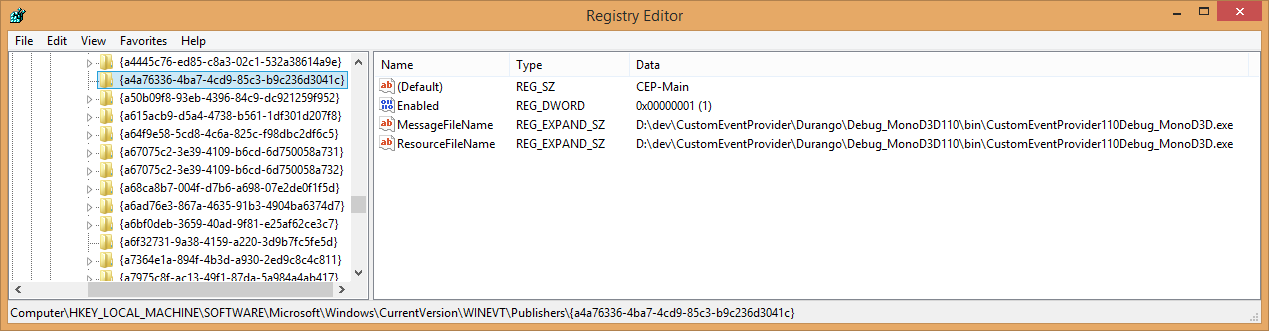
首先，编辑事件清单的提供程序节点 (evtprovider.man) 并确保 **resourceFileName** 和 **messageFileName** 属性指向开发电脑上构建 Xbox One 可执行文件的位置：

<provider name="CEP-Main" guid="{A4A76336-4BA7-4CD9-85C3-B9C236D3041C}"   
symbol="CEP\_MAIN"   
resourceFileName="D:\dev\CustomEventProvider\Durango\Debug\_MonoD3D110\bin\CustomEventProvider110Debug\_MonoD3D.exe"   
messageFileName="D:\dev\CustomEventProvider\Durango\Debug\_MonoD3D110\bin\CustomEventProvider110Debug\_MonoD3D.exe">

接下来，通过提升的命令提示符运行 wevtutil.exe 工具，在主机上注册事件提供程序：

D:\dev\CustomEventProvider>wevtutil im etwprovider.man

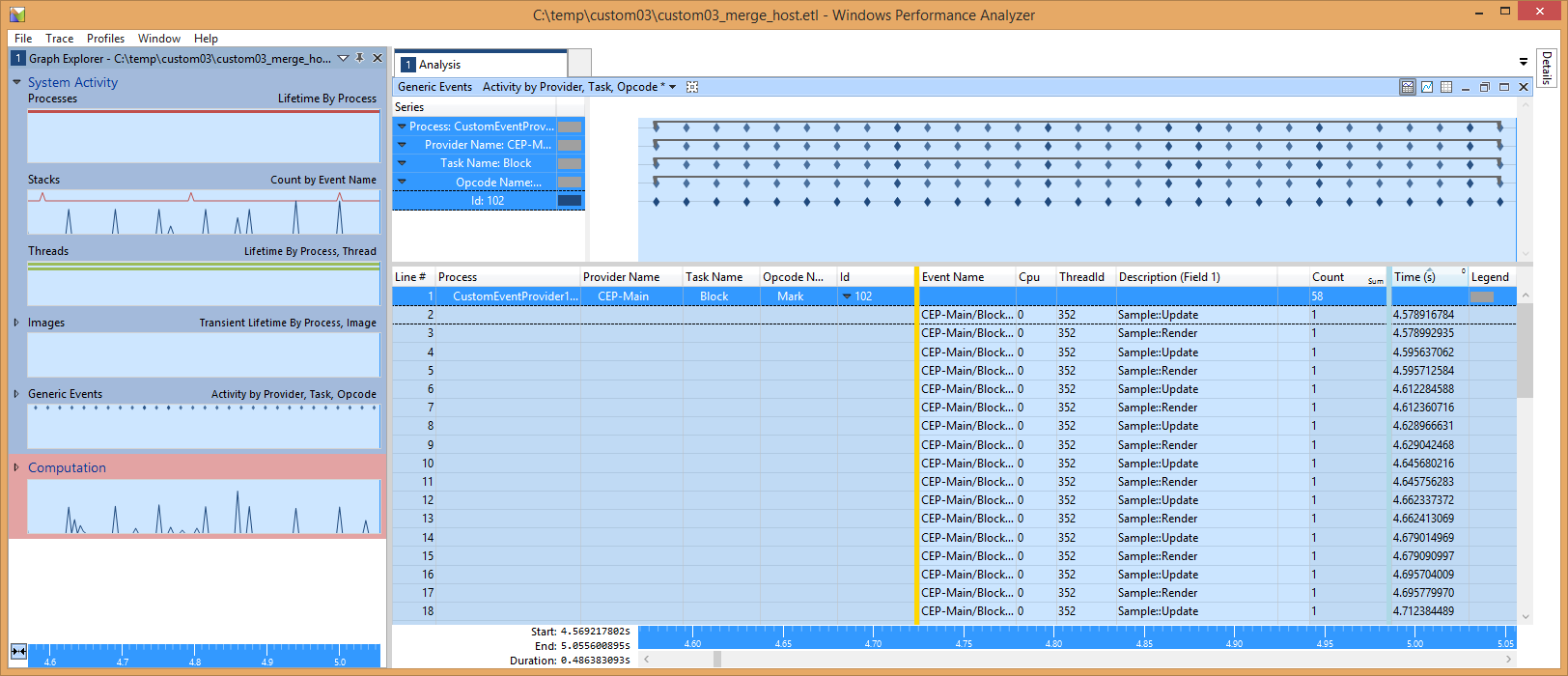
查看主机上的注册表时，应该可以看到 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers 下列出了该提供程序



最后，使用 xperf 解析在主机上的 ETL 文件：

C:\temp\custom03>xperf -merge custom03\_merge.etl custom03\_merge\_host.etl

如果主机合并的 ETL 文件已加载至 WPA，则您现在应该看到正确解析的事件：



请注意 Description (Field 1) 列现在如何包含随事件记录的字符串。我们还会看到任务和操作代码名称。

完成性能分析会话后，可从主机中删除提供程序：

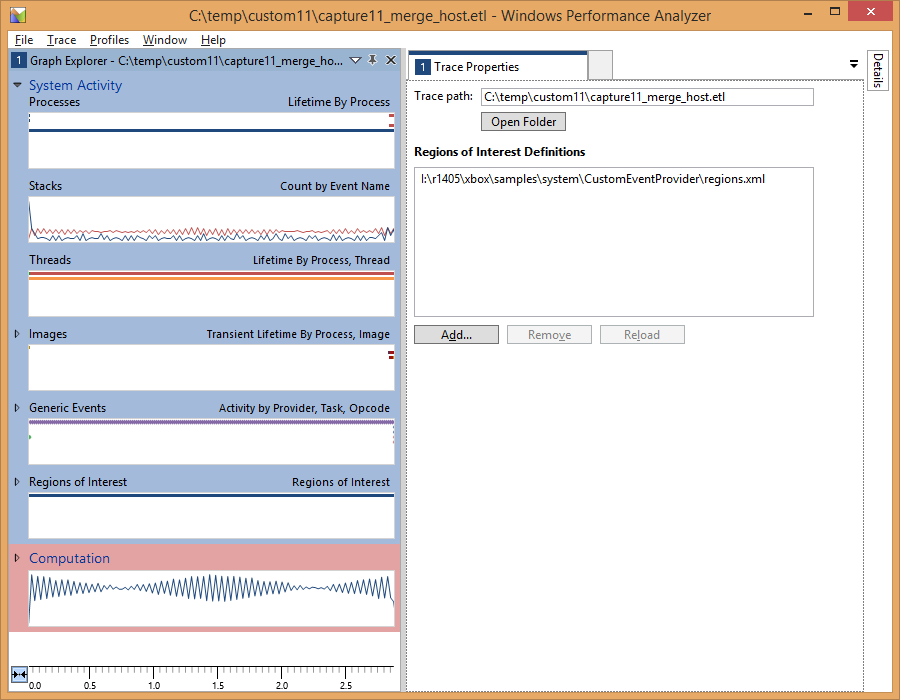
D:\dev\CustomEventProvider>wevtutil um etwprovider.man

BlockCulled 事件与 Mark 事件类似，但是它具有的是单个 UInt32 负载而不是字符串负载。很遗憾，目前无法在 WPA 中绘制自定义事件的数字字段。

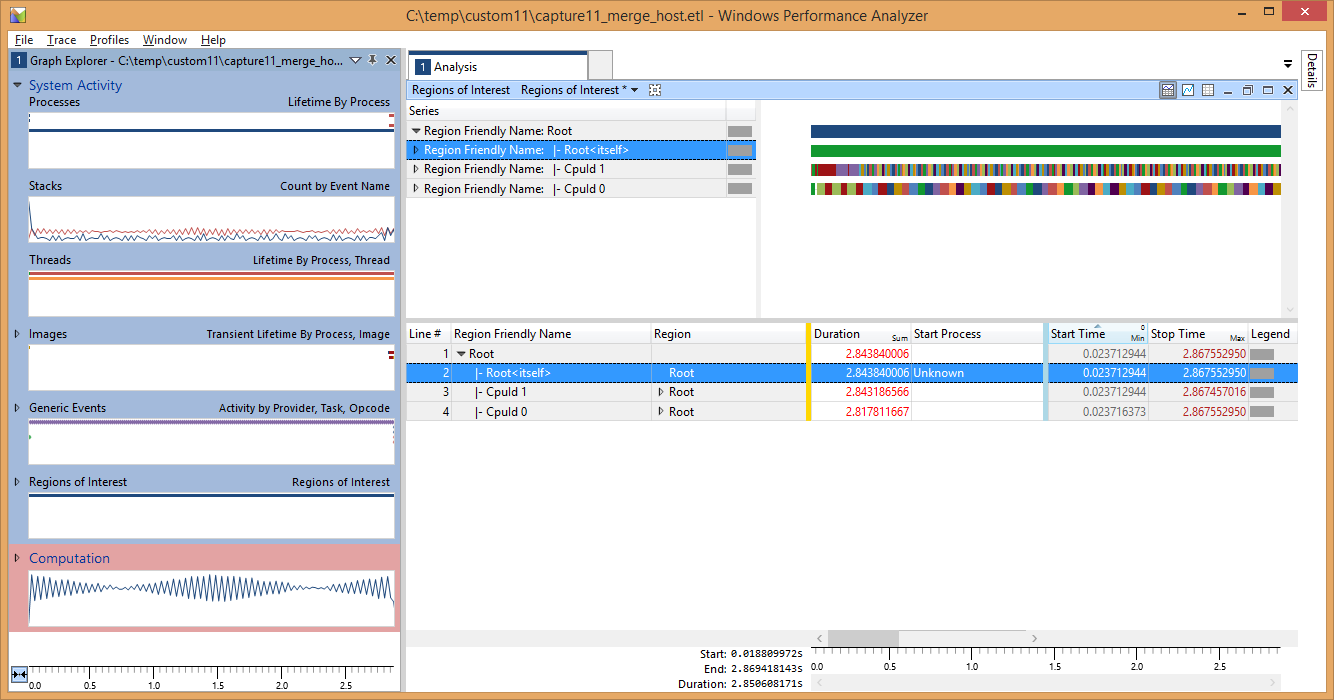
**感兴趣区域**

自 2013 年 10 月以来，WPA 一直支持[感兴趣区域](http://msdn.microsoft.com/zh-cn/library/windows/hardware/dn450838.aspx)这一概念：在捕获文件中表示和标记临时范围的功能。**EtwScopedEvent** 类和 **ETWScopedEvent()** 宏演示了如何利用适用载荷，通过感兴趣区域 (ROI) 提供类似于 **PIXBeginEvent()** 和 **PIXEndEvent()** 的包围功能。

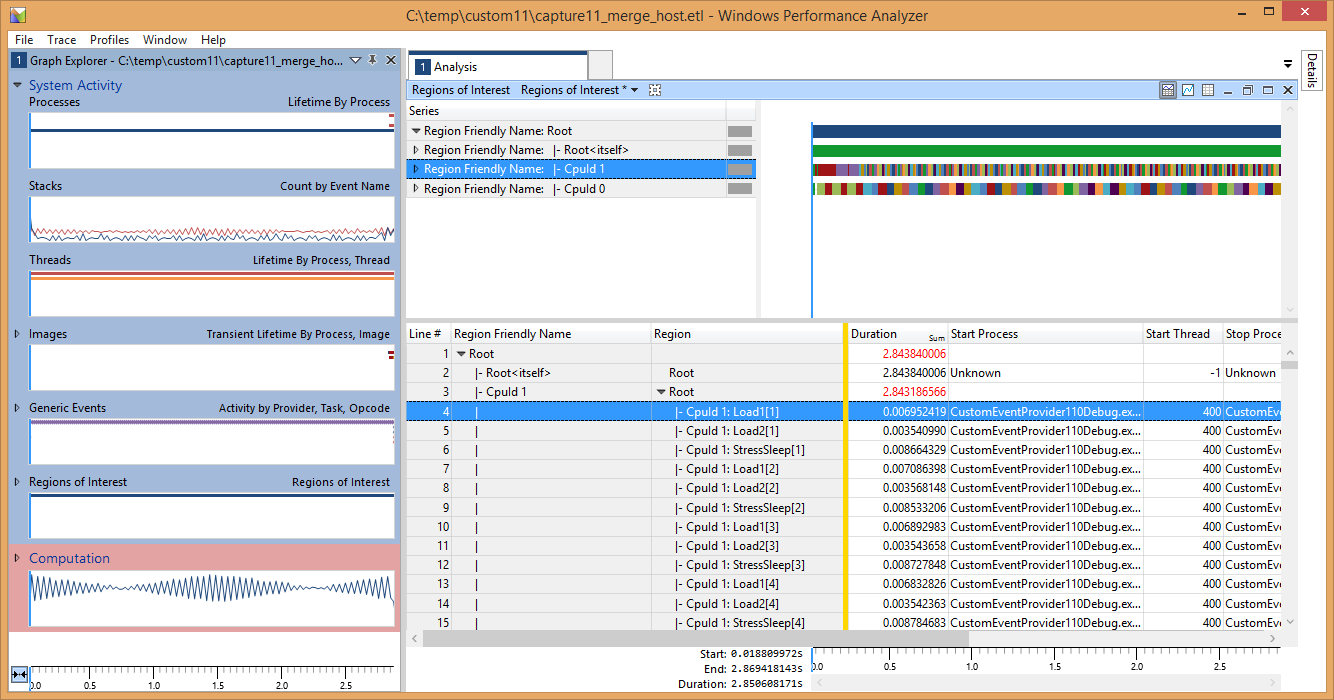
要显示 ROI，首先需要加载区域定义文件。从“跟踪”(Trace) 菜单中选择“跟踪属性”(Trace Properties)，然后加载本示例随附的 regions.xml 定义。



您现在应该可在“通用事件”图下看到“感兴趣区域”图。将 ROI 图拖至分析区域进行展开；默认视图预设（工具栏上）应为“感兴趣区域”。将 Region 列添加到表中（以便各区域颜色唯一），然后展开根节点；应该看到类似以下的显示：

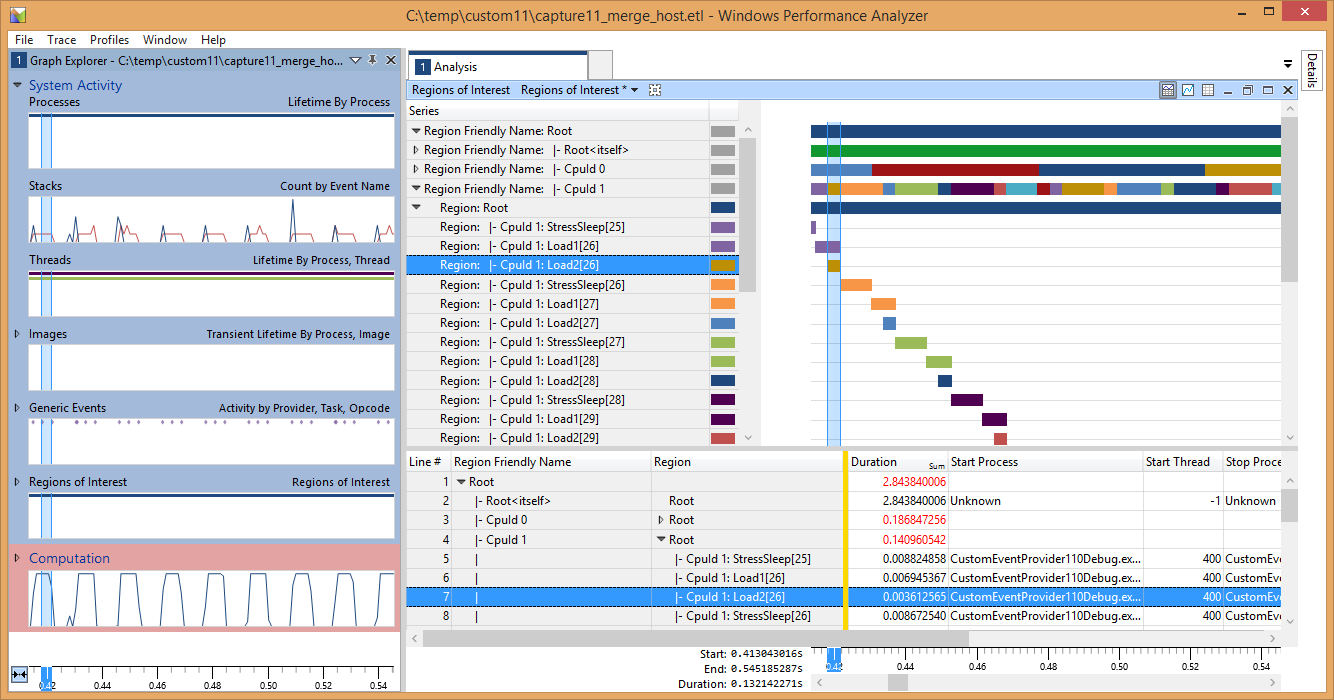


扩展表 Region 节点会显示有关各个支架的信息：

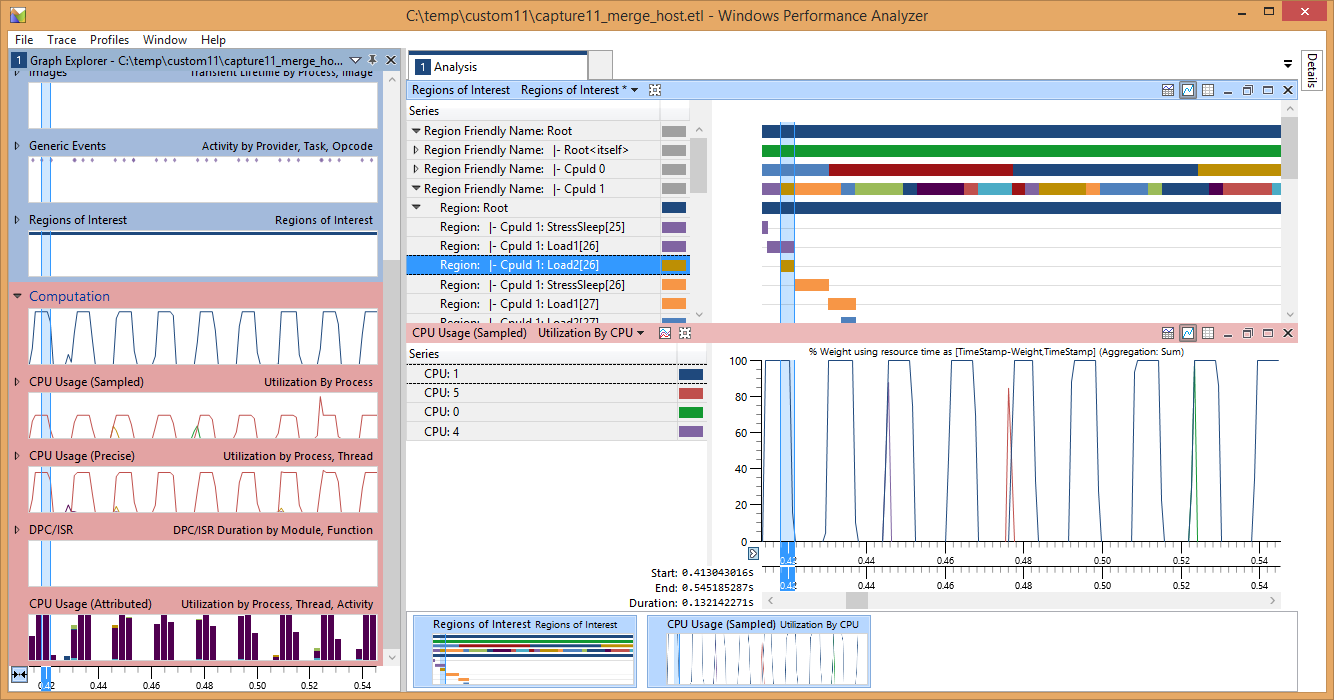


可以看到，**ETWScopedEvent()** 调用中提供的标签在这里可见（数字是特定标签的实例）。

展开图表将分别显示各区域的时间轴：



而且，现在您如果启用此项，可将区域与采样捕获中的数据相关联：



# 已知的问题

我们在主机上解析了事件提供程序 GUID，因此必须确保无已注册的具有该 ID 的 ETW 提供程序。如果使用本示例中的事件清单创建新事件清单，则在编辑清单时使用 ecmangen 重新生成 GUID，或使用 guidgen.exe（随 Visual Studio 提供）生成新 GUID。

在主机上解析 ETL 文件时，您可能会在“通用事件”视图中看到一些其他事件提供程序；这些可忽略。

# 隐私声明

编译和运行示例时，示例可执行文件的文件名将发送给Microsoft以帮助跟踪示例使用情况。要选择退出此数据收集，您可以删除Main.cpp中标记为“Sample Usage Telemetry”的代码块。

有关 Microsoft 隐私政策的更多信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/zh-cn/privacystatement/)。