

# CAT.net 客户端用户手册

## 目录

CAT.net 客户端用户手册.....	1
简介 .....	1
编译工程 .....	1
配置 .....	2
执行工程自带的测试用例 .....	2
在其他应用中引用 Cat.dll, 调用 CAT API .....	4
日志输出 .....	7
为心跳报表获取 .NET 性能数据 .....	7

## 简介

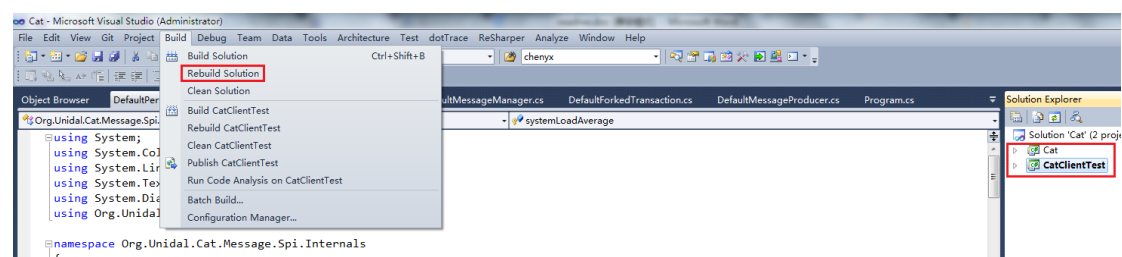
CAT.net 客户端旨在为 .net 应用提供接入 CAT 的 API。  
其 API 设计、客户端配置方式, 与 CAT Java 客户端基本一致。

## 编译工程

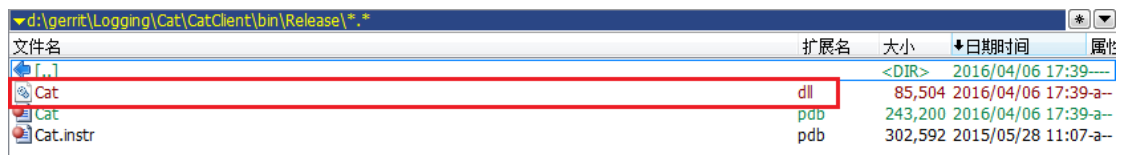
CAT.net 客户端要求 .NET Framework 4.0 或更高版本。  
用 Visual Studio 2010 或更高版本, 打开 Cat\Cat.sln。可以看到 Solution 中包括两个工程:

1. Cat: CAT.net 客户端实现代码
2. CatClientTest: 示例程序和测试用例。

单击 Rebuild Solution 编译这两个工程:



编译的输出是 **Cat.dll**, 如下图。在业务应用的工程中, 通过引用这个 dll, 调用其中的 API, 来接入 CAT。



## 配置

- 1) 创建以下目录，确保执行 CAT 客户端的帐户有权限读写它们：
  - d:\data\appdatas\cat\ (CAT 客户端使用的临时数据目录)
  - d:\data\applogs\cat\ (CAT 客户端的日志输出目录)
- 2) 创建 d:\data\appdatas\cat\client.xml。在其中配置 Domain ID 和 CAT 服务器地址。推荐 client.xml 用 UTF-8 编码。client.xml 内容如下：

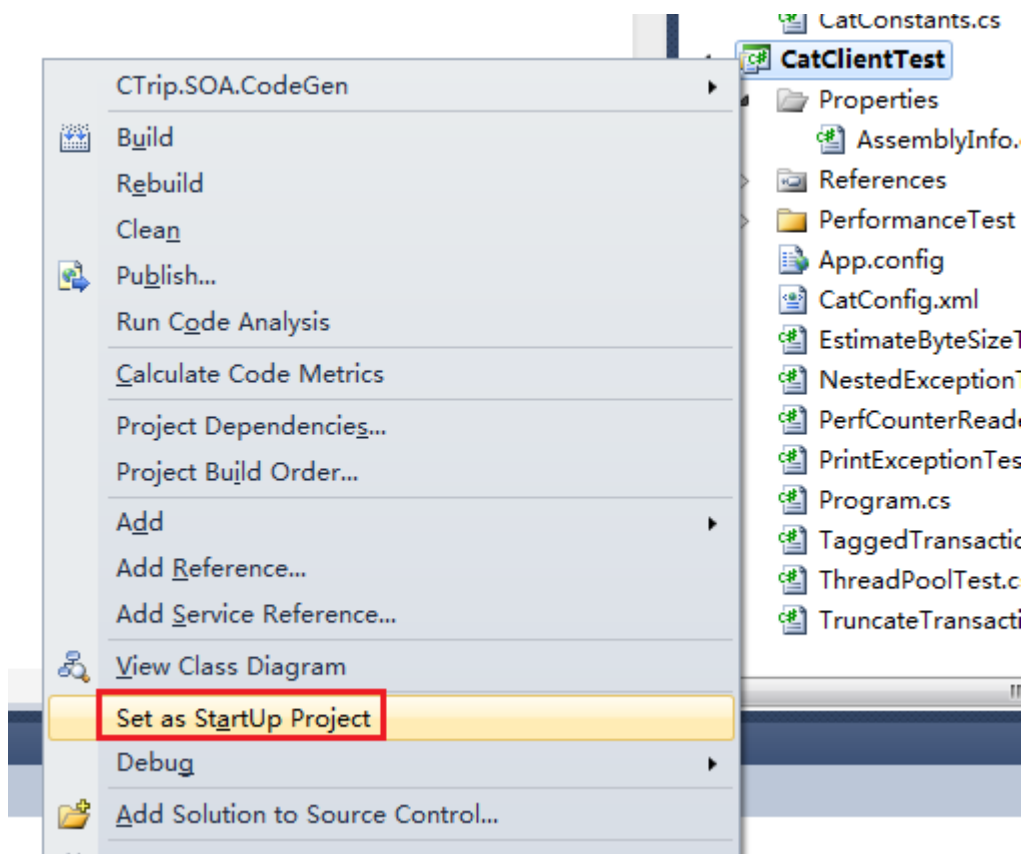
```
<?xml version="1.0" encoding="utf-8"?>
<config mode="client" enabled="true" queue-size="123">
  <!--logEnabled enabled="true"></logEnabled-->

  <!-- 配置 Domain ID-->
  <domain id="1237" enabled="true" max-message-size="1000"/>

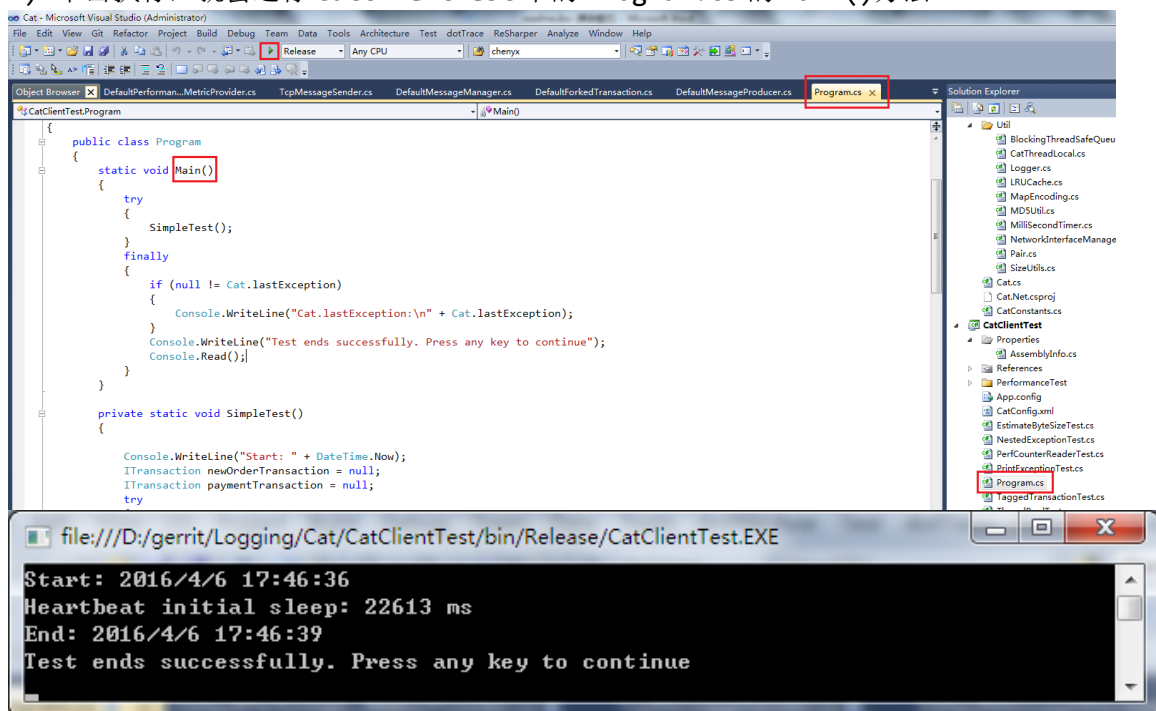
  <servers>
    <!-- 配置 CAT 服务器地址-->
    <server ip="10.2.6.98" port="2280" http-port="8080"></server>
  </servers>
</config>
```

## 执行工程自带的测试用例

- 1) 设置 CatClientTest 工程为默认启动工程：



2) 单击执行，就会运行 CatClientTest 中的 Program.cs 的 Main() 方法。

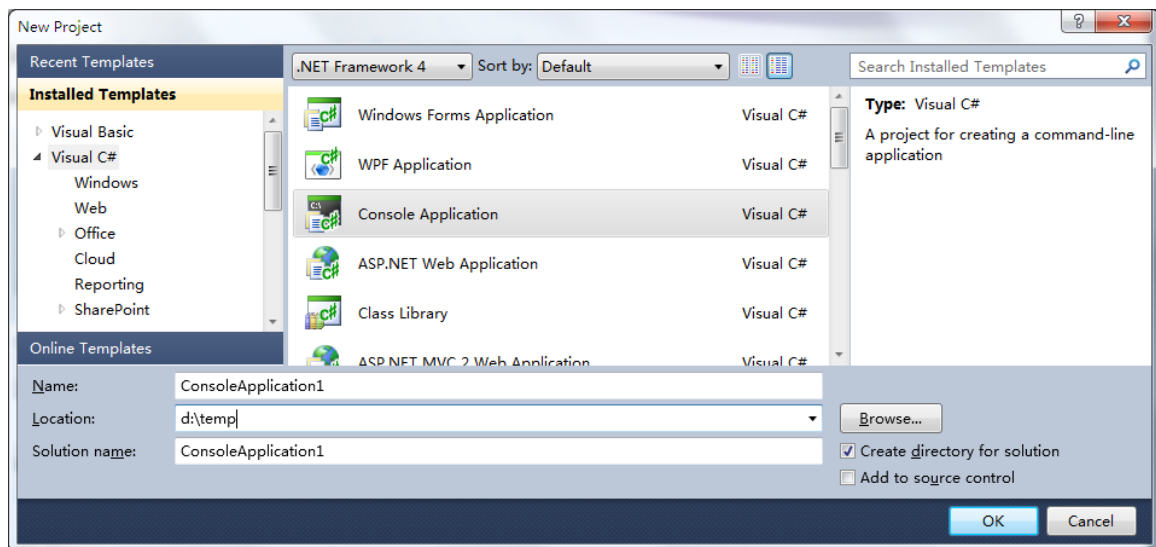


3) 在 CAT 中可以看到测试程序的 CAT 埋点，如下图。其中的 CAT 服务器地址、Domain ID 应该与 client.xml 中的配置一致。

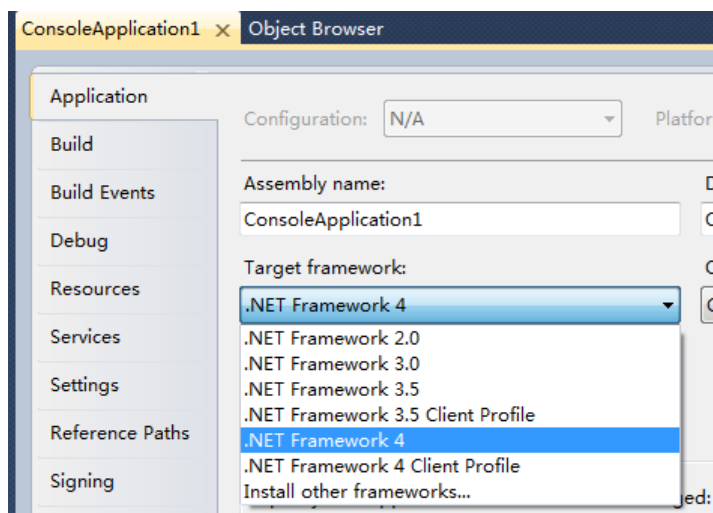
Type	Total	Failure	Failure%	Sample Link	Slowest	Min(ms)	Max(ms)	Avg(ms)	95Line(ms)	99.9Line(ms)	Std(ms)	QPS
System	2	0	0.0000%	[:: show ::]	Log	1	75	38.0	75.0	75.0	37.0	0.0
SimpleTest	1	0	0.0000%	[:: show ::]	Log	12	12	12.0	12.0	12.0	0.0	0.0
NewPayment	1	0	0.0000%	[:: show ::]	Log	0	0	0.0	0.0	0.0	0.0	0.0

## 在其他应用中引用 Cat.dll，调用 CAT API

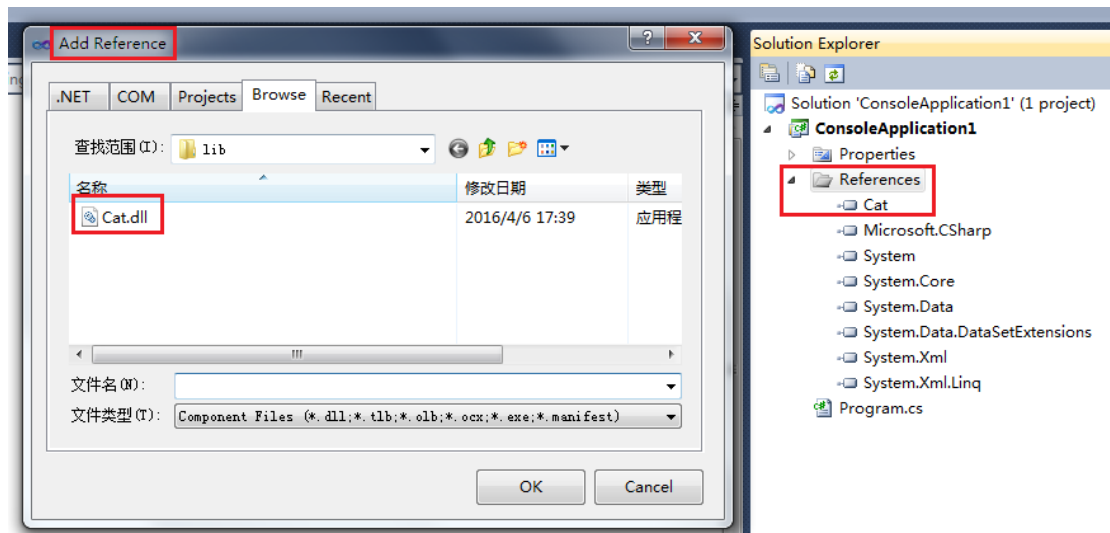
1) 假设我们有一个 Console 应用：



确保工程使用了 .NET Framework 4.0 或更高版本的服务端 Profile，而不是 **Client Profile**。



2) 添加对 Cat.dll 的引用：



3) 调用 CAT API 埋点。示例代码:

```
class Program
{
    static void Main(string[] args)
    {
        ITransaction transaction = null; ;
        try
        {
            transaction = Cat.NewTransaction("Order", "Cash");

            // Do your business...

            Cat.LogEvent("City", "Shanghai");
            transaction.Status = CatConstants.SUCCESS;
        }
        catch (Exception ex)
        {
            transaction.SetStatus(ex);
        }
        finally
        {
            transaction.Complete();

            // 程序退出前睡一会儿。使得 CAT 客户端有时间发出最后一批消息到网络。
            Thread.Sleep(1000);
        }
    }
}
```

4) 执行以上 Main() 方法。

5) 在 CAT 中可以看到埋点效果:



The screenshot shows the CAT web interface with a table of transaction data. The table has columns for Name, Total, Failure, Failure%, Sample Link, Slowest, Min(ms), Max(ms), Avg(ms), 95Line(ms), 99.9Line(ms), Std(ms), QPS, and Percent%. The data is filtered by 'Order' and 'Cash'.

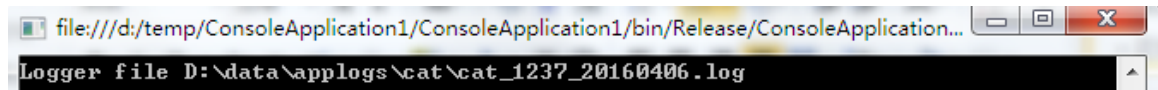
Name	Total	Failure	Failure%	Sample Link	Slowest	Min(ms)	Max(ms)	Avg(ms)	95Line(ms)	99.9Line(ms)	Std(ms)	QPS	Percent%
Type:Order	1	0	0.0000%	[:: show ::]	Log	3	3	3.0	-	-	0.0	0.0	100.00%
Cash	1	0	0.0000%	[:: show ::]	Log	3	3	3.0	3.0	3.0	0.0	0.0	100.00%

日期: 2016-04-06 IP: 10.32.21.24

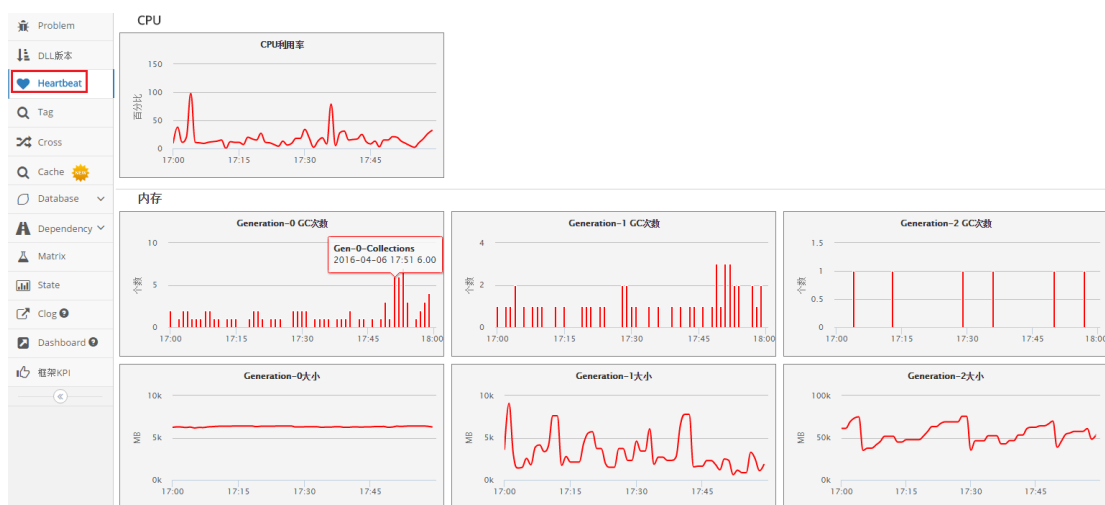
t18:05:55.321	Order	Cash
E18:05:55.323	City	Shanghai
T18:05:55.324	Order	Cash

## 日志输出

- 1) 在 client.xml 中, 启用<logEnabled enabled="true"></logEnabled> XML 元素, 以启用日志输出。
- 2) 日志输出位于 D:\data\applogs\cat 目录中:

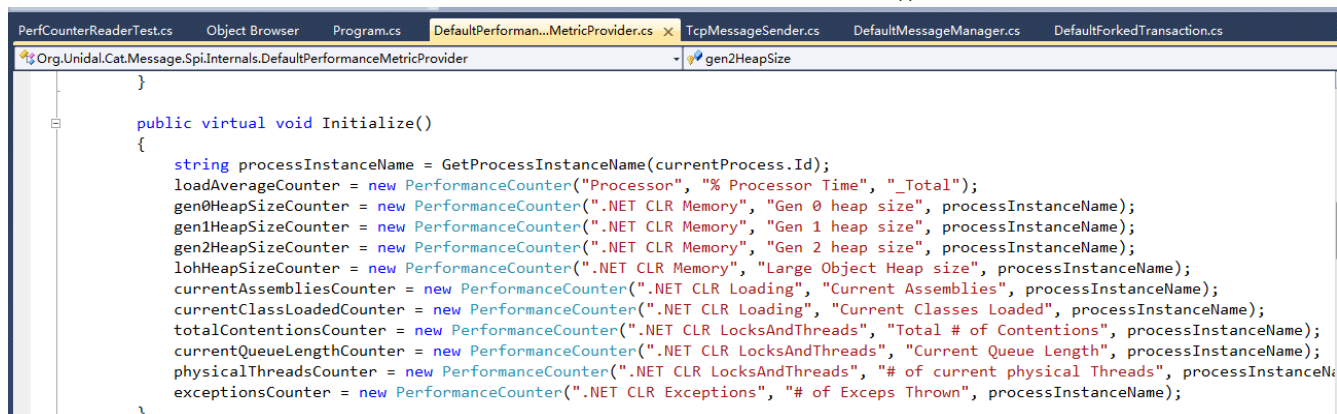


## 为心跳报表获取 .NET 性能数据



如上图, CAT.net 客户端每分钟会自动抓取一次机器级别的性能数据, 展现在 Heartbeat 报表中, 包括 CPU 利用率、GC 次数、Heap 各代大小、锁竞争次数、锁请求队列大小等。

- 1) 这些性能指标中的一部分, 是通过读取 .NET Performance Counter 实现的。  
(见 DefaultPerformanceMetricProvider.cs 中的 Initialize())。

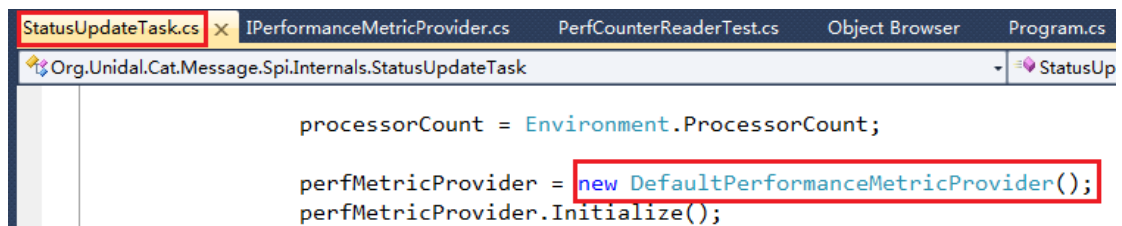


- 2) 然而, 读取 .NET Performance Counter 需要执行应用的帐号是“Performance Log Users”用户组的成员 (参考[这里](#))。特别地, 在 IIS 中以普通账户 (IIS\_USRS) 执行的 Web 应用, 是没有这个权限的。

所以，您可能需要提供自己的 `IPerformanceMetricProvider` 实现，它通过其他有权限读取的数据源（如 `Zabbix Agent`，`Salt Agent`，或自己实现的一个有更高执行权限的 `Performance Counter` 输出程序）来获取这些性能指标。

在您的实现中，对于 `IPerformanceMetricProvider` 接口中的 `Get*()` 方法，它应当返回过去 1 分钟内的某个性能指标的累积值（如 GC 次数），或过去 1 分钟内的平均值（如 CPU 利用率）。不推荐它返回从程序启动至今的累积值/平均值。请参考 `DefaultPerformanceMetricProvider` 是如何遵循这一语义的。

通过修改 `StatusUpdateTask.cs`，用您的 `IPerformanceMetricProvider` 实现，来替换掉 `DefaultPerformanceMetricProvider` 这个默认实现，如下图。



The screenshot shows the Visual Studio IDE with the `StatusUpdateTask.cs` file open. The file is part of the `Org.Unidal.Cat.Message.Spi.Internals` namespace. The code snippet shows the initialization of `perfMetricProvider` using `new DefaultPerformanceMetricProvider()`, which is highlighted with a red box. The `StatusUp` property is also visible in the Solution Explorer.

```
processorCount = Environment.ProcessorCount;

perfMetricProvider = new DefaultPerformanceMetricProvider();
perfMetricProvider.Initialize();
```