ENTERPRISE ARCHITECT

用户指南系列

编译和调试

Author: Sparx Systems

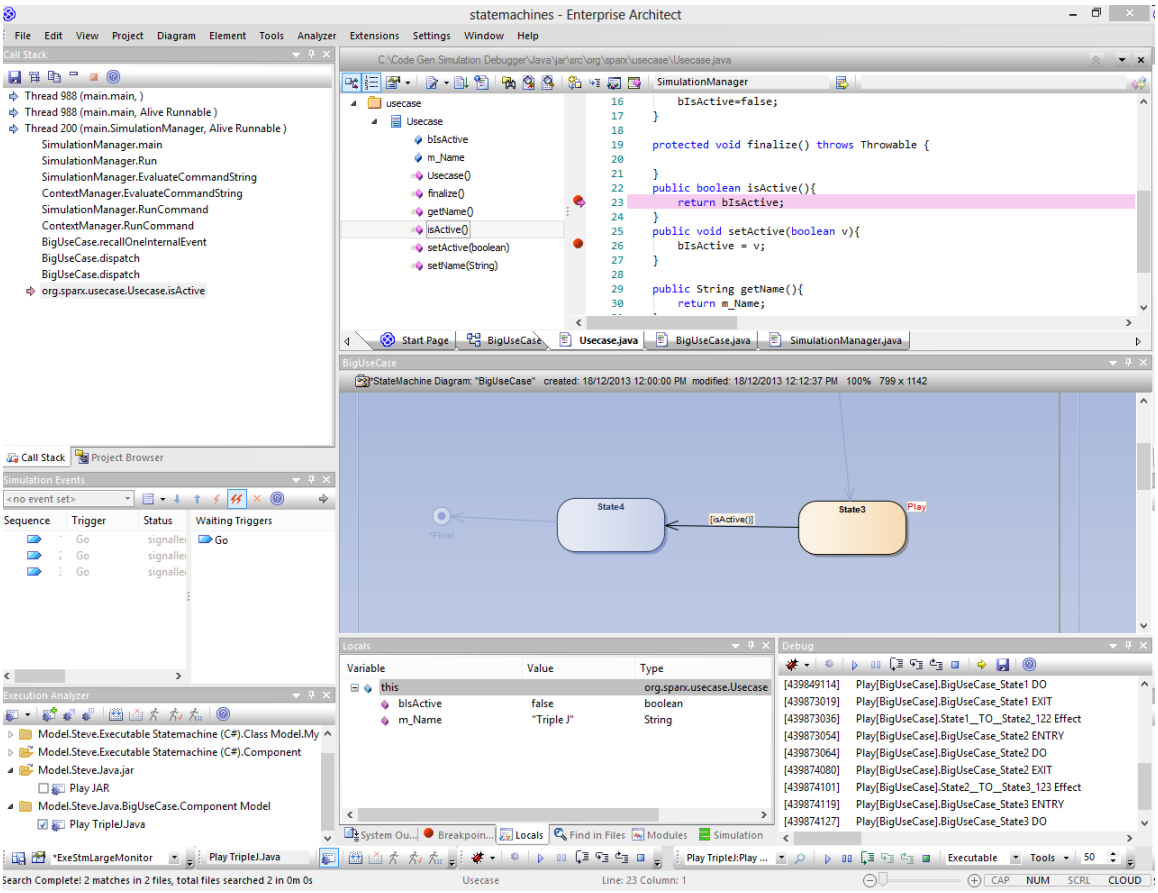
Date: 20/06/2023

Version: 16.1

创建于 ENTERPRISE ARCHITECT

目录

编译和调试



Enterprise Architect建立在其已经卓越的代码生成、图表和设计能力之上，具有完成的工具来构建、调试、可视化、记录、测试、配置文件以及以其他方式构建和验证软件应用程序。该工具集与建模和设计功能密切相关，并提供了一种独特而实用的方法来从模型构建软件并保持模型和代码同步。

Enterprise Architect帮助您定义链接到模型包的 分析器脚本”，描述如何编译应用程序、使用哪个调试器以及其他相关信息，例如模拟命令。分析器脚本将代码链接到Enterprise Architect中的构建、调试、测试、分析和部署功能的核心配置项。

作为工具集能力的衡量标准，应该注意的是， Enterprise Architect实际上是在Enterprise Architect开发环境中完全构建、调试、分析、测试和构建的。许多高级调试工具（例如行动点）已被开发用于解决构建大型复杂软件应用程序（例如Enterprise Architect ）中固有的问题，并且Sparx Systems开发团队每天都会使用这些工具。

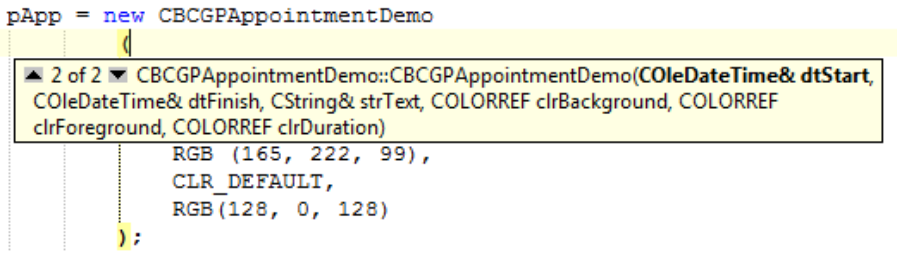
建议新用户花时间充分了解分析器脚本的使用以及他们如何将模型与代码、编译器和构建软件所需的其他工具联系起来。

集成模型和代码

模型驱动工程是一种现代的软件开发方法，承诺更高的生产力和更高质量的代码，从而使系统更快地进入市场并减少故障。使这种方法引人注目的是架构和系统设计能够在模型中进行描述和维护，然后生成编程代码和模式，这些代码和模式可以与模型同步并在模型中可视化。

Enterprise Architect的模型驱动开发环境(MDDE) 支持这种方法并提供一组灵活的工具来提高生产力和减少错误。其中包括在模型中定义架构和设计、从这些模型生成代码、将代码与模型同步以及在复杂的代码编辑器中维护代码的能力。也可以导入源代码或二进制文件，用户可以记录和记录预- 现有或最近开发的代码。分析器脚本帮助您描述如何构建、调试、测试和部署应用程序。

功能	描述

模型驱动开发	与传统的编码驱动周期相比，模型驱动开发提供了更强大、更易于访问和更快的开发周期。 A良好，与源代码构建、运行、调试、测试和部署密切相关的能力，提供丰富、易于导航和易于理解的目标架构可追溯性模型，与使用案例、组件和其他模型的链接，以及能力轻松记录和记录预先存在或最近开发的代码，使Enterprise Architect的开发环境具有独特的有效性。 Enterprise Architect结合了行业标准的智能编辑、调试器和建模语言。
模型驱动开发环境（MDDE）	MDDE 提供了设计、可视化、构建和调试应用程序的工具： • UML技术和工具到模型软件 • 用于生成/逆向工程源代码的代码生成工具 • 导入源代码和二进制文件的工具 • 支持不同编程语言的代码编辑器 • 智能感知帮助编码 • 分析器使用户能够描述如何构建、调试、测试和部署应用程序的脚本 • 与Java 、.Net、Microsoft C++ 等编译器集成 • Java 、 .NET 、Microsoft C++ 等的调试功能 • 高级可视化、记录、检查、测试和分析能力 

服务

Enterprise Architect提供了两种服务来促进远程脚本执行和远程调试。这些服务主要支持在 Linux 上运行的 Enterprise Architect ，以允许用户运行原生 Linux shell 脚本和调试 Linux 程序。卫星服务支持分析器脚本，代理人服务支持调试。

访问

功能区	执行 > 工具 > 服务

卫星服务

卫星服务负责在其运行的机器上执行分析器脚本。该特征可以帮助 Linux 用户绕过Wine直接执行本机 Linux 程序和 shell 命令。该服务可以从功能区进行管理，也可以运行于终端运行。

Linux 外壳

Enterprise Architect使用的默认 shell 是 "bash"。要覆盖Enterprise Architect使用的 Linux Shell，请打开 Linux 终端，运行 "wine regedit"并将string值添加到此注册表项：

HKEY_CURRENT_USER\Software\ Sparx Systems \EA400\EA\Options

在哪里：

- 键名："LINUX"
- 关键值：路径

*path*是 shell 程序 "/bin/bash"的 Linux 路径，例如。

权限

在 Linux 下，您必须检查服务程序是否具有适当的权限。这些程序位于Enterprise Architect安装文件夹下的子目录 "EA/x86/linux"中。选择此目录中的每个程 都具有所有者的执行权限。

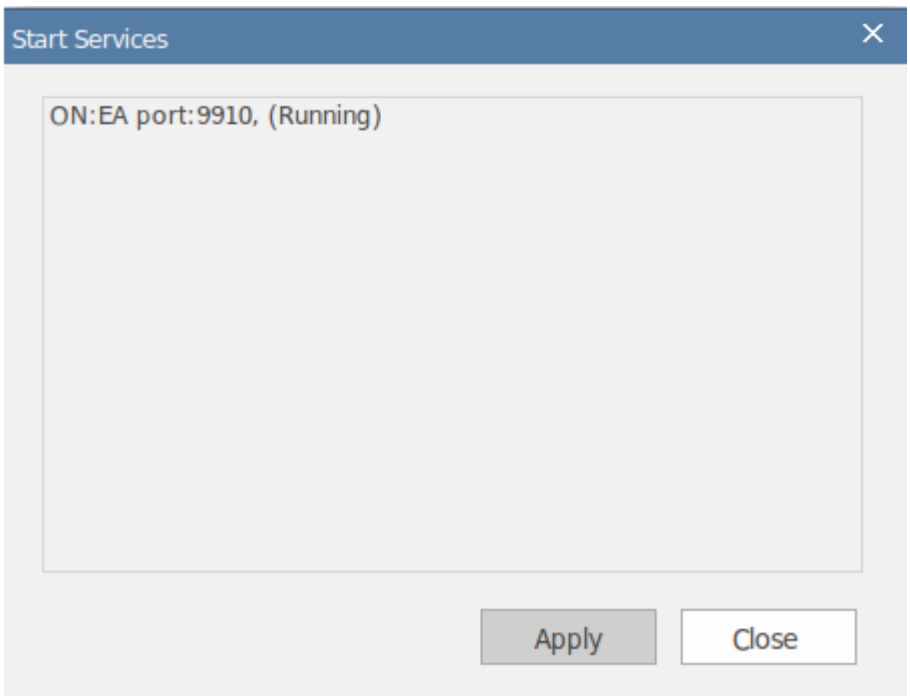
注记

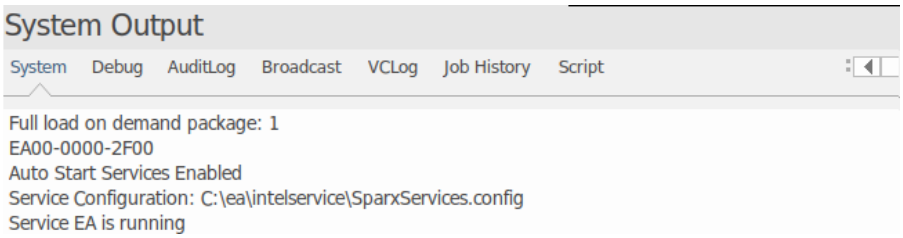
- Enterprise Architect统一版和终极版中启用了卫星服务

代理人服务

代理人服务负责管理Enterprise Architect的 GDB 调试器的调试会话。该服务允许Enterprise Architect用户调试 Linux 程序。可以从功能区管理服务。它也可以运行于终端运行。

服务菜单

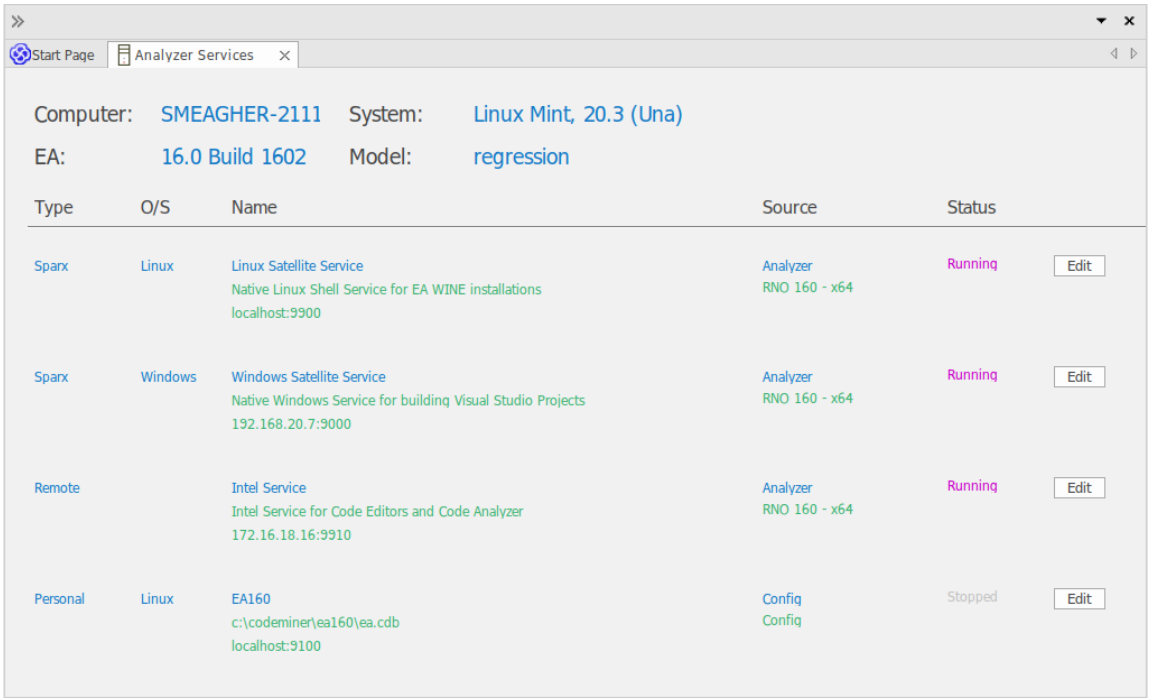
选项	描述
所有服务的视图状态	显示一个视图，列出配置文件中状态的每个Enterprise Architect服务的状态及其状态。
卫星服务	
开始	启动服务。该服务侦听任何分析器配置的卫星端口脚本Page。
停止	停止服务。
测试	测试卫星服务的状态，是否运行。
代理人服务	
开始	启动服务。该服务代理人端口配置的分析器脚本页面。
停止	停止服务。
测试	测试状态代理人的状态，是否正在运行。
代码矿工服务	
开始	该选项读取当前的Service配置文件，启动配置为运行的服务，停止运行未配置为运行的运行。 A以下情况下配置服务： 1. 它在配置文件中命名。 2. 它具有属性状态：ON。 
停止所有	此选项停止当前正在运行的所有服务。

编辑配置文件	此选项提示要使用的服务配置文件，然后在Enterprise Architect文本编辑器中打开该文件。系统会记住文件所在的位置。 31 # <number> - digits 32 # ---------- 33 # 34 { 35 name=project1, 36 status=ON, 37 lazyload=true, 38 port=9910, 39 allow=localhost, 40 network=local, 41 autoupdate=true, 42 show=true, 43 logoutput=c:\My Documents\project1.txt, 44 loglevel=information|warning|error, 45 database=c:\My Documents\project1\project1.cdb 46 } 47
自动开始使用 EA	此选项会在模型打开时自动启动具有 状态：ON"属性的服务。  当模型打开时，此处记录到系统输出窗口的消息表明该服务已经在运行。
关闭时自动停止	此选项在Enterprise Architect关闭时自动停止运行服务。

分析器服务窗口

分析器服务窗口显示以下各项的状态：

- 由主动分析器脚本的服务（通过'执行>工具>分析器'功能区选项维护）
- 使用服务配置文件单独管理的任何本地服务（可从 执行 > 工具 > 服务 >代码矿工服务 > 编辑配置文件”功能区选项中获得）



例如，该窗口让您可以一目了然地看到您正在使用的英特尔服务，或者您用于在Enterprise Architect中工作的窗口服务正在运行。

窗口中的所有数据都是只读的，但您可以通过单击 编辑”按钮来编辑服务。这将显示分析器脚本编辑器窗口的 私人选项 - 服务”页面，您可以根据需要对其进行更新。

访问

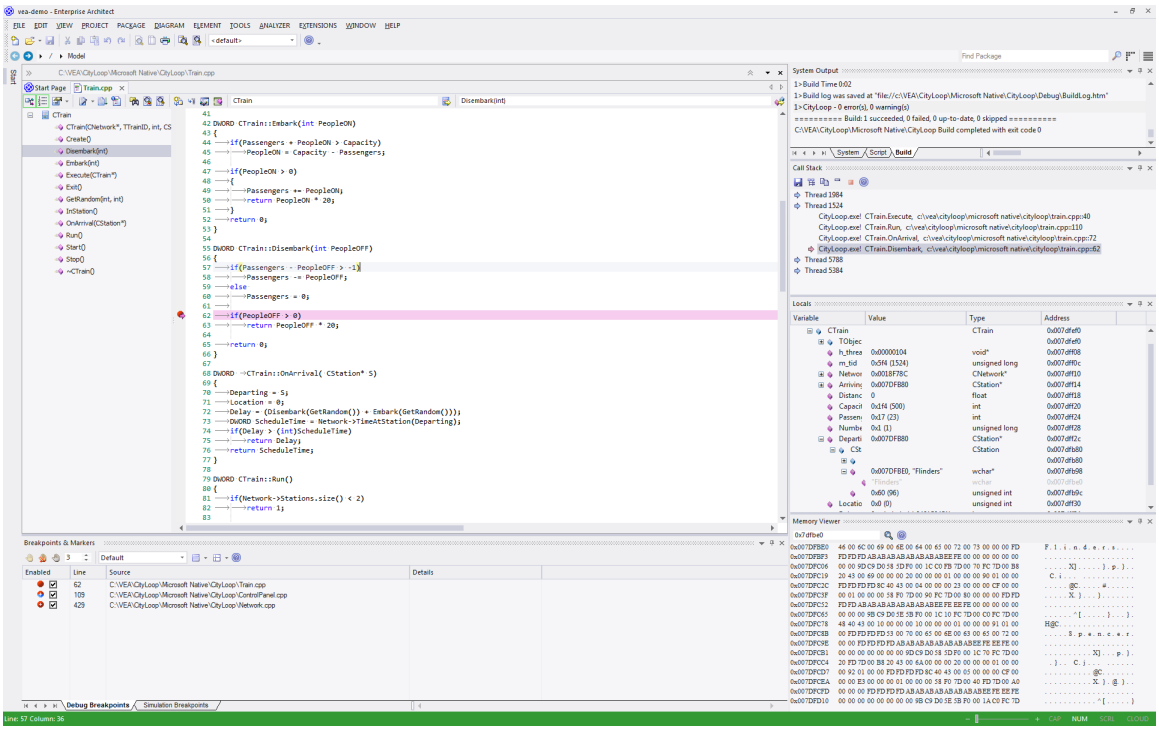
功能区	执行 > 工具 > 服务 >视图所有服务的状态 >开始> 所有窗口> 执行 > 分析 |分析器服务
键盘快捷键	Alt+4 |分析|分析器服务

分析器服务窗口字段

字段	描述
电脑	运行服务的工作站。
系统	工作站运行的操作系统。

EA	您正在使用的Enterprise Architect版本的发行号和内部版本号。
模型	您当前正在使用的模型的名称。
类型	服务类型。列为 "Sparx"类型的服务是Enterprise Architect Services。
O / S	运行服务的操作系统。
名称	服务的名称和描述。 • 当应用程序在WINE下的 Linux 上运行时，Linux卫星服务由Enterprise Architect自动管理 • 窗口卫星服务是一项可选服务，通常用于WINE安装中，以帮助在远程窗口机器（如 VM（虚拟机））上构建 Visual Studio 项目；该服务仅在窗口窗口本地进行管理
源	定义服务的脚本组或配置文件。
状态	服务的状态。 当您第一次打开窗口时，每个服务最初都有 状态测试"作为系统评估服务。状态然后更改为适当的值，例如 正在运行"或 已停止"。
编辑	单击此按钮可查看或编辑服务定义。

调试



Enterprise Architect不仅仅是一个绘图工具——它还提供了您在 IDE 中可能期望的每个特征。提供了适用于许多主要平台的综合调试环境和工具。在建模工具中集成调试功能允许其作者开发、构建和管理代码。在集成模型中工作和协作使行动变得重要，并且每一个行动都以使用其他工具链无法实现的方式负责。

特征

速度

Enterprise Architect中的调试器很快！逐步完成程序不会花费一整天的时间。

记录程序无需手动执行即可执行。

支持

- C++、C 和 Visual Basic
- 微软.NET , ASP.NET WCF
- Java ，使用套接字传输（JDWP）或内存模型（JVMIT）
- 模拟器或设备上的 Android
- JavaScript , VBScript 和 JScript
- Apache Web 服务器上的 PHP 脚本
- 在窗口中使用Enterprise Architect的远程 Linux GDB 进程
- 仿真- UML和 BPMN 中的调试仿真
- 可执行状态机——调试执行状态机

隔离

调试器在Enterprise Architect的进程外运行，将其与副作用隔离开来。

效率

启动和停止调试器既快速又轻松。它不会阻止你。设计为响应式 UI，主 UI 线程与不属于其职责的职责隔离。

生产率

从建模切换到需求，从提出变更请求到跟踪整个组织共享的模型中的代码变更，再到分析最近的代码变更。一站式工具。

注记

- 可视化执行分析器特征调试和记录特性不适用于 Oracle 的Java服务器平台 Weblogic”

运行调试器

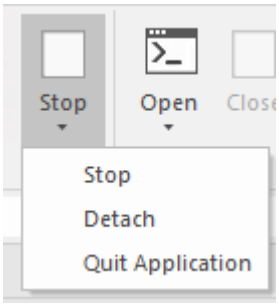
Enterprise Architect提供了许多方法来启动和控制调试会话。有主调试窗口，以及调试工具栏和 执行”功能区中的 运行”面板。每次运行调试会话时最好显示调试窗口，因为这是捕获所有调试输出的地方。

访问

功能区	执行 >运行>开始 执行 > 工具 >调试器>开始调试
键盘快捷键	Alt+8（显示调试窗口） F6（开始执行被调试的应用程序）
工具栏	探索 >门户> 显示工具栏 >调试

使用调试窗口

行动	细节
开始调试器	当分析器脚本已配置为支持调试时，您可以通过以下方式启动调试器： • 从功能区中，选择 执行 >运行>开始>运行” • 从功能区中，选择 执行 > 工具 >调试器>开始调试” • 在 调试”工具栏上，单击  按钮，或 • 按 F6 您也可以通过 分析器脚本窗口”中的上下文菜单启动任何脚本的调试器，或按 Shift+F12 如果你没有分析器脚本仍然可以通过直接附加到该进程来调试正在运行的应用程序： • 从功能区中，选择 执行 > 工具 >调试器> 附加到进程”，或者 • 在 调试”工具栏上，点击  （附加）按钮，手动选择调试平台
暂停/恢复调试	您可以通过以下方式暂停调试会话，或在暂停后恢复会话： • 从功能区中，选择 执行 >运行> 暂停” • 在 调试”工具栏上，单击  按钮
停止调试器	调试器通常在当前调试进程终止时结束；但是，某些应用程序和服务（例如Java虚拟机）可能需要手动停止调试器。要停止调试，请执行以下任一操作： • 在 调试”工具栏上，单击  （停止）按钮 • 按 Ctrl+Alt+F6 • 选择 执行 >运行> 停止”功能区选项上的下拉箭头

	功能区选项显示一个简短菜单，提供三种终止调试应用程序的方法。  • 停止 - 停止调试器并停止正在调试的进程（单击功能区图标时的默认设置） • 分离 - 停止调试器但让进程继续运行 • 退出应用程序 - 停止调试器并将 WM_QUIT 消息发布到进程的主窗口（如果有的话）
节过代码行	跳过下一行代码： • 从功能区中，选择 执行 >运行>节结束”，或 • 在 调试”工具栏上，单击 （节过）按钮，或 • 按 Alt+F6
节函数调用	进入函数调用： • 从功能区中，选择 执行>运行>节In”，或 • 在 调试”工具栏上，单击 （节入）按钮，或 • 按 Shift+F6 如果目标函数没有可用的源，则调试器立即返回给调用者。
节输出函数	退出函数： • 从功能区中，选择 执行>运行>节输出” • 在 调试”工具栏上，单击 （节输出）按钮，或 • 按 Ctrl+F6 如果调试器跳出一个没有源代码的函数，它将继续跳出，直到找到一个源代码的点。
显示执行点	在调试器暂停的同时，返回调试器即将执行的源文件和代码行： • 从功能区中，选择 执行>运行>开始>显示执行点” • 在 调试”工具栏上，点击 （显示执行点）按钮。 相应的行将突出显示，屏幕左边缘有一个粉红色箭头。
输出	在调试会话期间，调试窗口中显示的消息详细说明： • 启动的启动 • 会话终止 • 例外 • 错误 • 跟踪消息，例如使用Java系统.out 或.NET系统的输出。诊断。调试 如果双击调试消息，则：

	• 弹出窗口A更多完成消息文本，或 • 如果存在内存泄漏，则文件将显示在错误发生的位置
保存输出（并清除输出）	您可以将调试输出的全部内容保存到外部 .txt 文件，也可以将输出中的选定行保存到Enterprise Architect剪贴板。 要将所有输出保存到文件，请单击  （将输出保存到文件）按钮。 要将所选行保存到剪贴板，请右键单击所选内容并选择 将所选行复制到剪贴板”选项。 当您保存输出或不想再显示它时，右键单击当前输出并选择 清除结果”选项。
切换到探查器	如果您正在代码上运行调试会话，您可以停止调试会话并立即切换到分析会话。 从调试器切换到 Profiler： • 从功能区中，选择 执行 > 工具 >调试器> 切换到 Profiler” • 在调试窗口中，单击  |切换到 Profiler 选项，或 • 在调试工具栏上，单击  |切换到 Profiler 选项 Profiler 附加到当前运行的进程。 此功能不适用于Java调试器。

断点和标记管理

Enterprise Architect中的断点工作与任何其他调试器中的工作方式相同。标记类似于断点，但在Enterprise Architect中它们具有特殊的功能。简而言之，标记执行断点不执行的操作——例如记录执行和分析。断点的作用始终是停止程序。

您可以在源代码编辑器中设置任何标记或断点，它们在左边距中可见。单击此边距将在该行添加一个断点。断点和标记是可互换的 - 您可以使用其 属性"对话框将断点更改为标记，反之亦然。您可以使用 Ctrl 并单击编辑器边缘或断点和标记窗口中的图标，快速查看和编辑断点或标记的属性。

断点保持成套。每个模型都有一个默认设置，每个断点通常都驻留在那里，但您可以将当前断点配置保存为命名集，创建新集并在它们之间切换。断点集是共享的；也就是说，它们可供模型社区使用。例外是默认集，它是分配给任何模型的每个用户的私有和个人集。

访问

功能区	执行>窗口>断点 仿真>动态仿真仿真>断点

断点和标记选项

选项	细节
删除断点或标记	要删除特定断点： • 如果断点开启，点击源代码编辑器左边空白处的红色断点圆圈，或者 • 右键单击源代码编辑器、断点文件夹或断点和标记窗口中的断点或标记，然后选择 删除"选项，或 • 在 调试断点"选项卡中选择断点，然后按删除键
删除所有断点	单击删除所有断点按钮 ()。
断点属性	在断点窗口或代码编辑器中，使用标记的上下文菜单调出属性。您可以在此处更改标记类型、添加或修改约束以及输入跟踪语句。 (有用的快捷键：按住 Ctrl 键的同时单击标记，快速显示其属性。)
禁用断点	取消选中断点或标记对应的复选框。
启用断点或标记	选中断点或标记对应的复选框。
禁用所有断点	单击  按钮
启用所有断点	单击启用所有断点按钮 ()。
修改内存地址时中断	单击数据断点按钮 ()。

识别或更改标记集	选择断点和事件窗口工具栏中的断点 Default 字段。 如有必要，单击下拉箭头并选择不同的标记集。 默认设置通常用于调试，并且是您的用户 ID 个人的；其他标记集在模型内的所有用户之间共享。
更改断点和标记如何在断点和事件窗口上分组	断点和标记可以按类或代码文件分组。要对项目进行分组，请单击工具栏中 图标上的向下箭头，然后单击相应的选项。如果您不想对项目进行分组，请单击已选择的选项以取消选择它；然后按行号列出断点和标记。

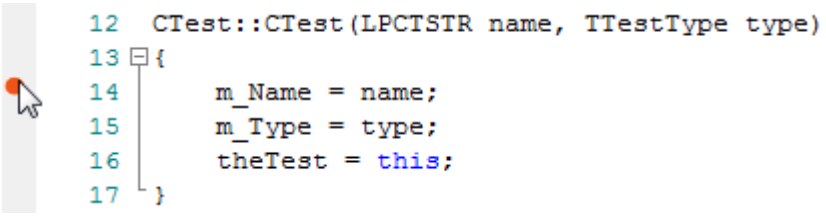
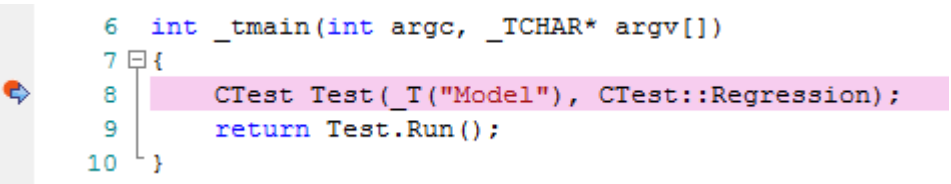
断点状态

状态	评论
	调试*Running:* Bound 调试未运行：已启用
	调试运行：禁用 调试未运行：禁用
	调试运行：未绑定 - 这通常意味着模块尚未加载。此外，dll 有时会被卸载。 调试未运行： N /a
	调试运行：失败 - 这意味着调试器无法将这行代码与任何已加载模块中的指令相匹配。可能源来自另一个项目或项目配置已过期。注记，如果模块日期早于断点的源代码日期，您将在调试器窗口中看到通知。文字是红色的，所以它们会脱颖而出。这清楚地表明该项目需要建设。 调试未运行： N /a

设置代码断点

正常断点通常设置在一行源代码上。当调试器在正常执行过程中到达指定行时，调试器停止执行并显示局部变量、调用堆栈、线程和其他运行时信息。

在一行代码上设置断点

节	行动
1	在集成源代码编辑器中打开源代码进行调试。
2	找到相应的代码行并单击左边距列 - 边距中的实心红色圆圈表示已在该位置设置断点。  如果代码当前在断点处暂停，则该点由标记旁边的蓝色箭头指示。  或者，您可以通过右键单击所需行的左边距来设置断点标记（或其他标记），以显示断点/标记上下文菜单；选择适当的标记类型。

跟踪声明

跟踪语句是在调试会话执行期间输出A消息。可以在Enterprise Architect中定义跟踪语句，而无需对应用程序源代码进行任何更改。

跟踪点标记在代码编辑器中设置。像断点一样，它们被放置在一行代码中。当该行代码执行时，调试器评估该语句，其结果记录到调试窗口（或者如果被分析分析器器脚本覆盖，则记录到文件）。

访问

任何现有的跟踪声明都可以在断点和标记窗口中查看和管理。可以使用此处概述的任何一种方法显示断点和标记窗口。

功能区	执行>窗口>断点

添加跟踪点标记

节	行动
1	在源代码编辑器中打开源代码进行调试。
2	找到适当的代码行，右键单击左边距并选择 添加跟踪点标记”选项。 如果标记已经存在，请按 Ctrl 并单击以显示断点属性窗口。
3	确保选中 跟踪状态”复选框。
4	在 跟踪语句”复选框下的文本字段中，键入所需的跟踪语句。
5	点击确定按钮。 A Marker 显示在代码编辑器的左边距中。

```
55 DWORD CTrain::Disembark(int PeopleOFF)
56 {
57     if(Passengers - PeopleOFF > -1)
58         Passengers -= PeopleOFF;
59     else
60         Passengers = 0;
61
62     if(PeopleOFF > 0)
63         return PeopleOFF * 20;
64
65     return 0;
66 }
```

指定跟踪

跟踪语句可以是A自由格式的文本。当前范围内的任何变量的值也可以通过在变量名称前加上特殊标记来包含

在跟踪语句中。

可用的令牌是：

- $ - 当变量被解释为string时
- @ - 当变量是原始类型时（ int 、double、char）

使用图像中的示例，我们可以使用以下语句输出下火车的人数：

在@PeopleOFF 在$Arriving 下车之前有@Passengers。名称Station

除了跟踪代码中的变量值之外，您还可以在跟踪语句中使用 $stack 和 $frame 关键字来打印当前堆栈跟踪；利用：

- $stack - 打印所有帧，或
- $frame[start](count) - 从给定帧开始从堆栈中打印特定数量的帧；例如，$frame[0](5) 将打印当前帧和 4 个祖先

注记

- 跟踪语句可以包含在任何类型的断点或标记上。

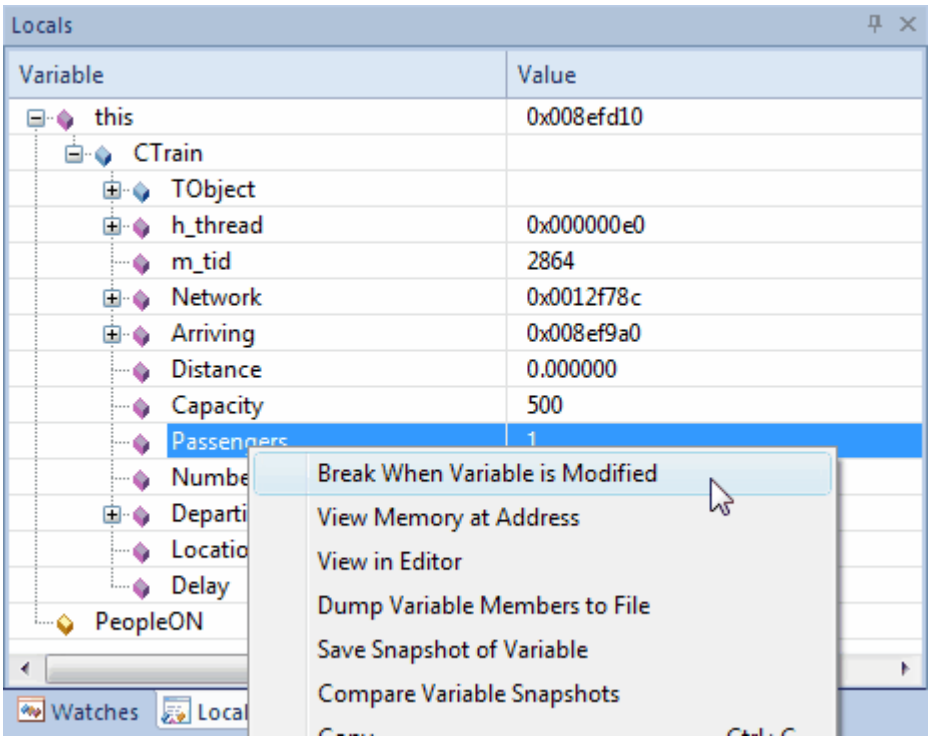
变量修改值时中断

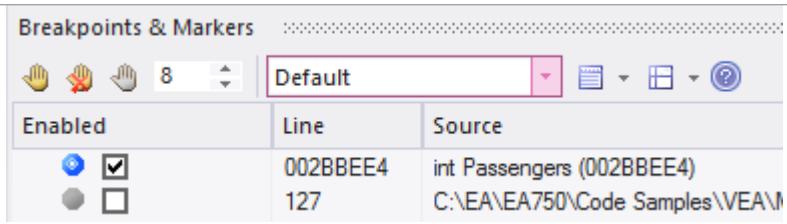
可以在预先确定的内存变量上设置数据断点，以使调试器在刚刚导致变量值更改的代码行处停止执行。这在尝试跟踪程序执行期间变量被修改的点时很有用，尤其是在不清楚程序执行如何影响特定object状态的情况下。

访问

功能区	执行 >窗口> 局部变量：右键单击变量 > 修改变量时中断或 执行 >窗口> 监视：右键单击变量 > 修改变量时中断
其它	在代码编辑器窗口中：右键单击感兴趣的变量 |修改项目时中断

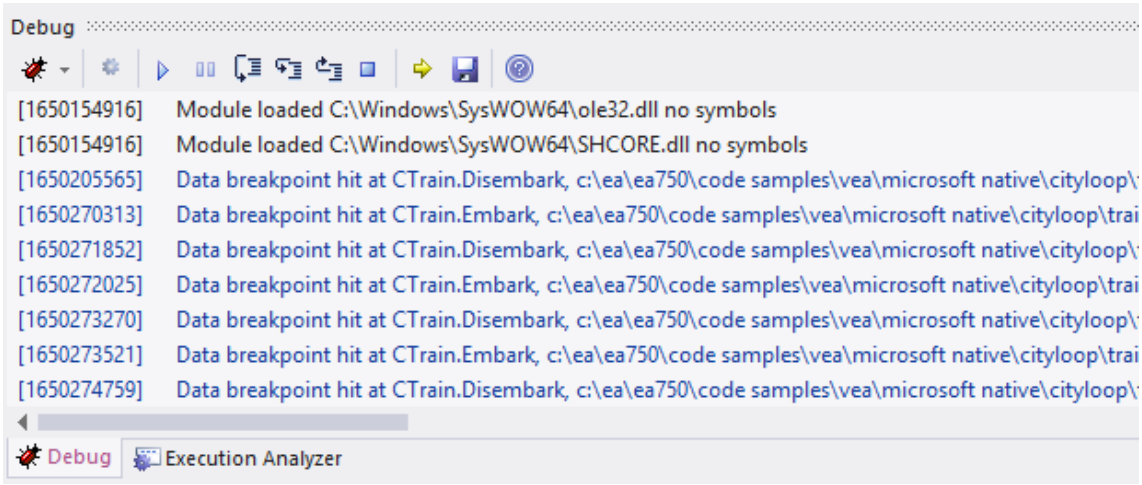
使用数据断点捕获对变量的更改

脚步	细节
1	在代码中设置一个正常的断点，以便您可以选择一个变量。然后运行调试器 (F6)。
2	当程序有断点时，选择感兴趣的变量，然后从其上下文菜单中，选择 "Break When Variable is Modified"选项。 
3	代码中没有断点指示符，但是断点和事件窗口中的数据断点很容易识别，是一个带有白色菱形的蓝色图标。 Enterprise Architect显示变量的名称及其地址而不是行号。



4 设置数据断点后，您可以禁用您可能拥有的任何其他断点。程序将在任何更改此变量值的代码行处停止。现在运行你的程序。

5 修改此变量时，调试器会暂停并在编辑器中显示当前代码行。这不是导致中断的行，而是事件之后的代码行。该事件被记录到调试窗口。

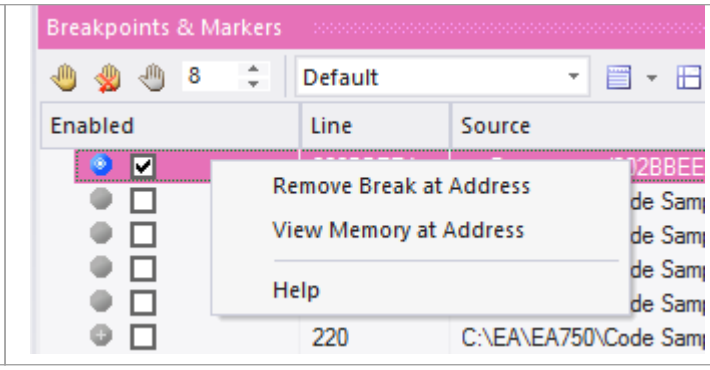


现在我们知道这个值（它的状态）如何以及在何处发生了变化。例如，第 58 行的语句刚刚更新了乘客的数量。

```
55 DWORD CTrain::Disembark(int PeopleOFF)
56 {
57     if(Passengers - PeopleOFF > -1)
58         Passengers -= PeopleOFF;
59     else
60         Passengers = 0;
61
62     if(PeopleOFF > 0)
63         return PeopleOFF * 20;
64
65     return 0;
66 }
```

6 发现此值和其他更改此值的位置后，请务必在继续之前删除通知。您可以通过在断点窗口中选择数据断点并按删除键来快速删除数据断点。

您也可以使用右键单击上下文菜单来执行此操作。



注记

- Microsoft .NET平台目前不支持此特征

跟踪变量修改值时

当您的代码执行时，它可能会更改变量的值。可以在调试窗口中捕获此类更改和变量的新值。然后，您可以双击更改记录以在代码编辑器中显示导致更改的代码行。

访问

功能区	执行>窗口>局部变量：右键单击变量>跟踪变量被修改或 执行 >窗口>跟踪: 右键单击变量 > 当变量被修改时
其它	在代码编辑器中 |右击变量跟踪当变量被修改时

设置跟踪

您要跟踪的变量必须在范围内，因此要识别和选择它，请在您知道该变量将存在的代码行上设置一个普通断点。当调试器到达此断点时，找到变量并使用其上下文菜单启用跟踪。

定位变量：

- 如果您在源代码中看到变量，请将鼠标悬停在其上，右键单击并选择 显示变量”选项； Enterprise Architect将找到它
- 如果变量在范围内（本地，或 this”或 this”的成员），请在本地窗口窗口中查找它（ 执行>窗口>局部变量”）
- 如果变量是全局变量（C、C++），则显示 Watches窗口（'Execute > Window> Watches'）并按名称搜索
- 如果变量是类静态成员，则显示 Watches 窗口（ 执行 >窗口> Watches”）并输入其完全限定名

启用跟踪后，您可以禁用所有其他断点并让程序运行。每次变量更改值时，都会记录到调试器的 输出”选项卡中。选择值的更改并双击该行以在代码编辑器中显示代码。

注记

- 发生更改事件时调试器不会停止，它只记录更改
- 此功能在 Microsoft Native 和Java平台上可用
- Microsoft .NET不支持值断点

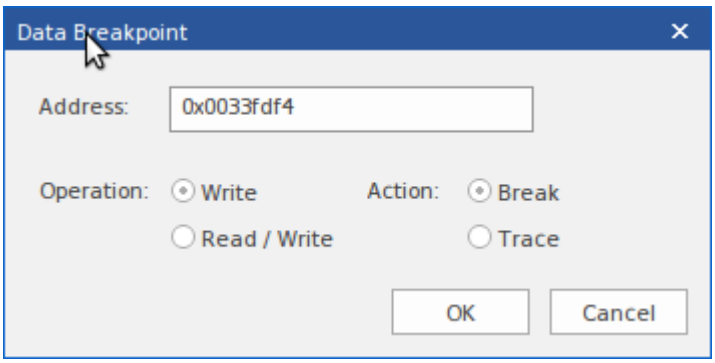
检测内存地址操作

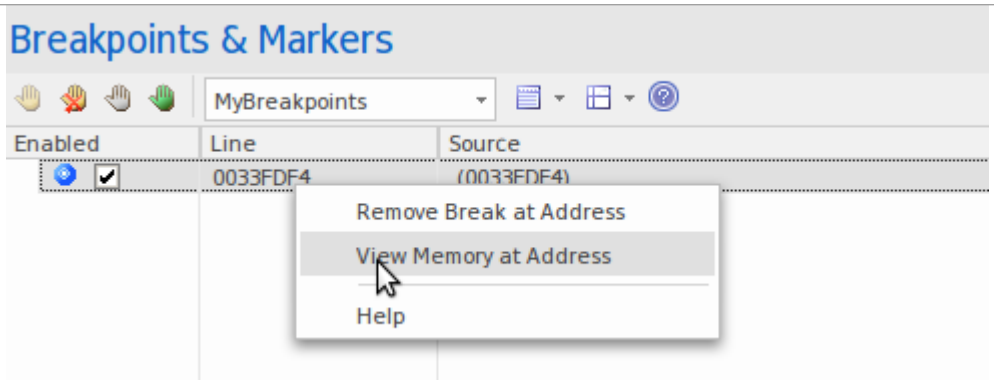
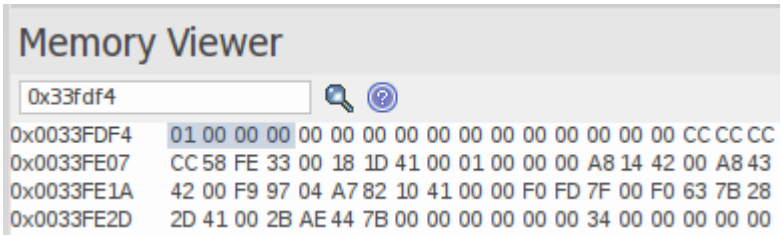
能够检测到内存区域被读取或写入的位置和时间对于调查人员来说是一个很大的帮助，即使代码库已经被很好地理解了。如果没有这个工具，C++ 开发人员可能会面临一项艰巨的任务，即跟踪访问全局变量的位置和时间，并调试这些函数。数据断点允许 C++ 程序员跟踪何时读取或写入变量/内存位置。当检测到该操作时，调试器将停止执行，并且该操作之后的代码行将显示在代码编辑器中。

访问

功能区	执行>窗口>断点

检测对内存地址的操作

节	行动
1	单击按钮。
2	输入要观看的内存地址。您可以从本地窗口（局部变量）窗口复制地址。 
3	选择要检测的操作。如果选择 写入”，则在写入地址时调试器将中断。如果您选择 读/写”，调试器将在读取或写入地址时通知您。
4	选择要执行的操作。如果您选择 Break”，调试器将停止程序并且代码行将显示在编辑器中。如果您选择 跟踪”，调试器将不会停止执行，但会在地址上log任何操作。此输出显示在调试器窗口中。
5	数据断点被添加到断点和标记窗口。

	
6	您可以使用数据上下文上的时间点菜单来检查内存地址处的值。 
7	要删除数据断点，请在断点和标记窗口中选择它，然后按删除键。或者，取消选中它旁边的复选框。数据断点在禁用时被删除；它们不像其他断点那样持续存在。

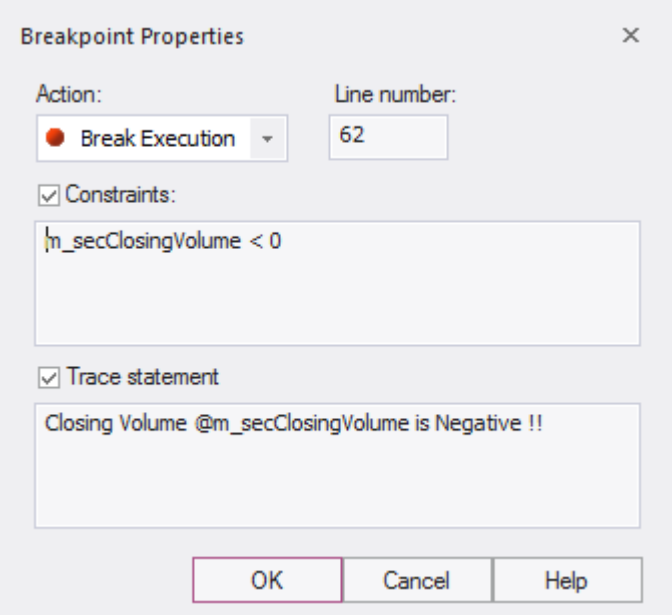
系统需求

C/C++ 本机调试器支持内存地址断点。

断点属性

断点有许多附加属性，这些属性确定在执行断点适用的代码行时会发生什么。
这些属性定义：

- 要执行的动作
- 断点适用的代码行
- 确定断点时是否执行动作的约束
- 断点时输出的跟踪信息



访问

有几种方法可以显示 断点属性”对话框：

代码编辑器	• 右键单击断点标记 |属性或 • Ctrl+单击断点标记或 • 右键单击具有断点标记的代码 |断点 |属性
断点和标记窗口	• 断点右键|属性

选项

字段	细节
行动	命中断点时的行为。
线	此断点适用的源代码行。
	对于堆栈捕获标记，要记录的调用者帧数。要记录整个堆栈，请将值设置为

堆叠高度	0。
约束	定义将执行断点操作的条件。对于正常断点，这将是停止执行的条件。在此示例中，对于正常断点，当条件评估为True时，执行将在此行停止。每次执行代码行时都会评估约束。 (this.m_FirstName="Joe") 和 (this.m_LastName="Smith")
跟踪声明	当断点被命中时，向调试窗口输出A消息。当前在作用域内的变量可以包含在跟踪语句输出中，方法是在变量名称前加上 $ 标记（用于string变量）或 @ 标记（用于原始类型（例如int或 long））。例如： 账户$pAccount->m_sName 余额为@pAccount->m_fBalance

绑定断点失败

A绑定断点出现问题，则会发生断点失败。断点失败最常见的原因是源文件被更改而没有重新构建应用程序。断点有时可以绑定到不同的行，导致它们被移动。如果断点不能在这一行或后面的三行绑定到二进制文件，则会显示一个问号。

断点和事件窗口的 详细信息”列中会显示A警告消息，指出问题的类型：

- 断点的源文件与用于构建应用程序映像的源文件不匹配
- 文件上的时间戳大于图像的时间戳

警告信息也会输出到调试窗口A

调试一个正在运行的应用程序

您可能希望调试已经在系统上运行的应用程序（进程），而不是从Enterprise Architect中显式启动进程。

在这种情况下，您可以使用调试功能附加到已经运行的进程。如果您将适当的调试信息写入正在运行的进程和/或关联的调试文件（例如 .PDB 文件），调试器就会绑定到该进程并启动调试会话。

您也可以在完成检查后从流程中 分离"并让运行正常运行。

访问

功能区	执行>运行>开始>附加到进程
调试器窗口	调试器窗口工具栏有一个附加按钮

阶段

阶段	描述
显示流程	当您选择调试另一个进程时，将显示 附加到进程"对话框。 您可以使用对话框顶部的单选按钮限制显示的进程；要查找 Apache Tomcat 或 ASP.NET 等服务，请选择系统单选按钮。
选择调试器	当您选择一个进程时，您可能必须从调试器下拉列表中选择调试器；但是，如果选择的包已经在分析器脚本中进行了配置，那么脚本中列出的调试器会预设在对话框中。
进程选择	双击包含调试信息的进程后， Enterprise Architect将附加到该进程： ● 调试器检测到遇到的任何断点 ● 遇到断点时停止进程，并且 ● 该信息在调试窗口中可用
脱离进程	要从进程中分离，请单击  （调试停止）按钮。

视图局部变量

本地窗口窗口显示执行系统的变量。无论你是录制C#，调试Java 、C++或VBScript，调试一个可执行状态机，还是运行一个模拟，这个窗口都是系统变量所在的地方。当前值仅在程序停止时显示。当在调试期间遇到断点时，当您跨过一行代码或当您在模拟中的状态之间切换时，就会发生这种情况。

访问

功能区	执行 >窗口> 局部变量 仿真>动态仿真> 局部变量
上下文菜单	在代码编辑器中 |右键单击任何变量标识符 > 显示变量

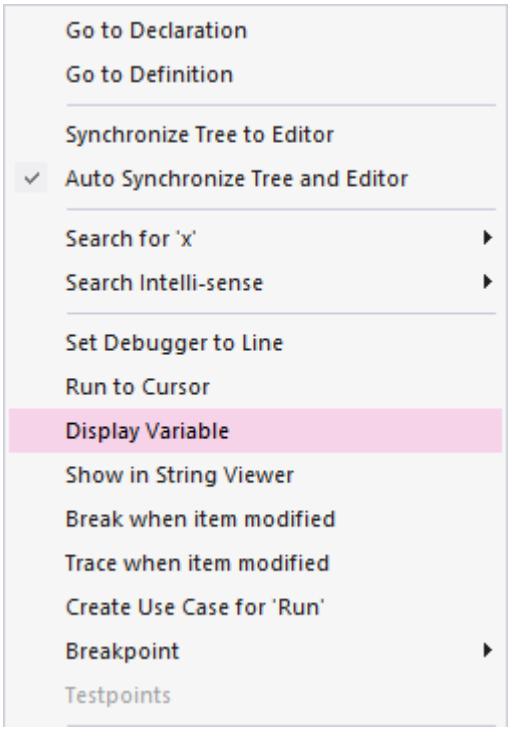
图标

任何范围内变量的值和类型都显示在树中；每个变量都有一个标识变量类型的彩色框图标：

- 蓝色-物件with members
- 绿色 - 数组
- 粉红色 - 元素类型
- 黄色 - 参数
- 红色 - 工作台实例

查找变量

查找变量的最简单方法是首先在代码编辑器中找到它，然后使用变量上的右键单击上下文菜单，选择 显示变量 ″。 Enterprise Architect将发现并揭示范围内的任何变量，包括深度嵌套的成员。如果在不同的范围（全局、文件、模块、静态）中找到变量，它将显示在 Watches 窗口中（请参阅其它范围视图的视图变量）。

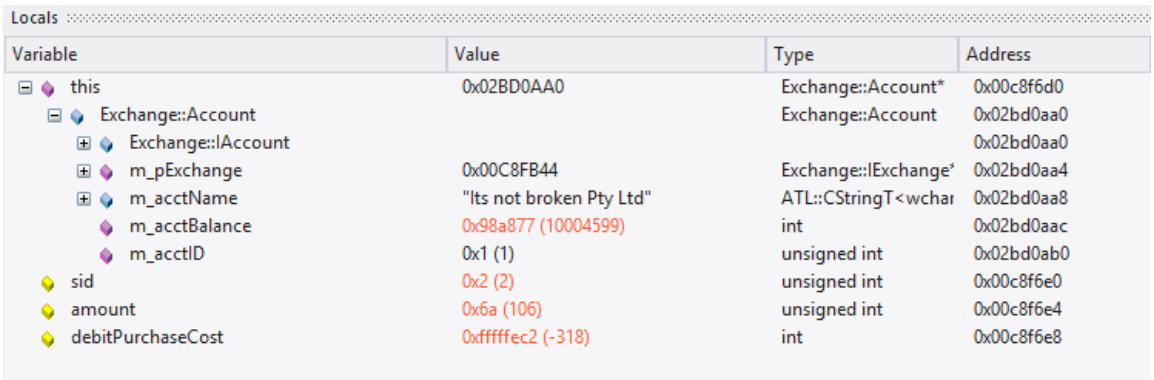


持久视图

变量的检查通常涉及在树中挖掘以显示感兴趣的值。在经历了这些麻烦之后，可能会很烦人，然后上下文下一行代码，只是由于时间的变化，这些变量又被隐藏起来了。本地窗口窗口有一个持久视图，在运行或 step 命令后会停留一段时间。当您在Enterprise Architect中单步执行函数时，变量结构会逐行保留。这使得逐步完成一个函数变得又快又容易。

发生了什么变化

作为持久视图的一部分，本地窗口窗口跟踪值的变化并突出显示它们。



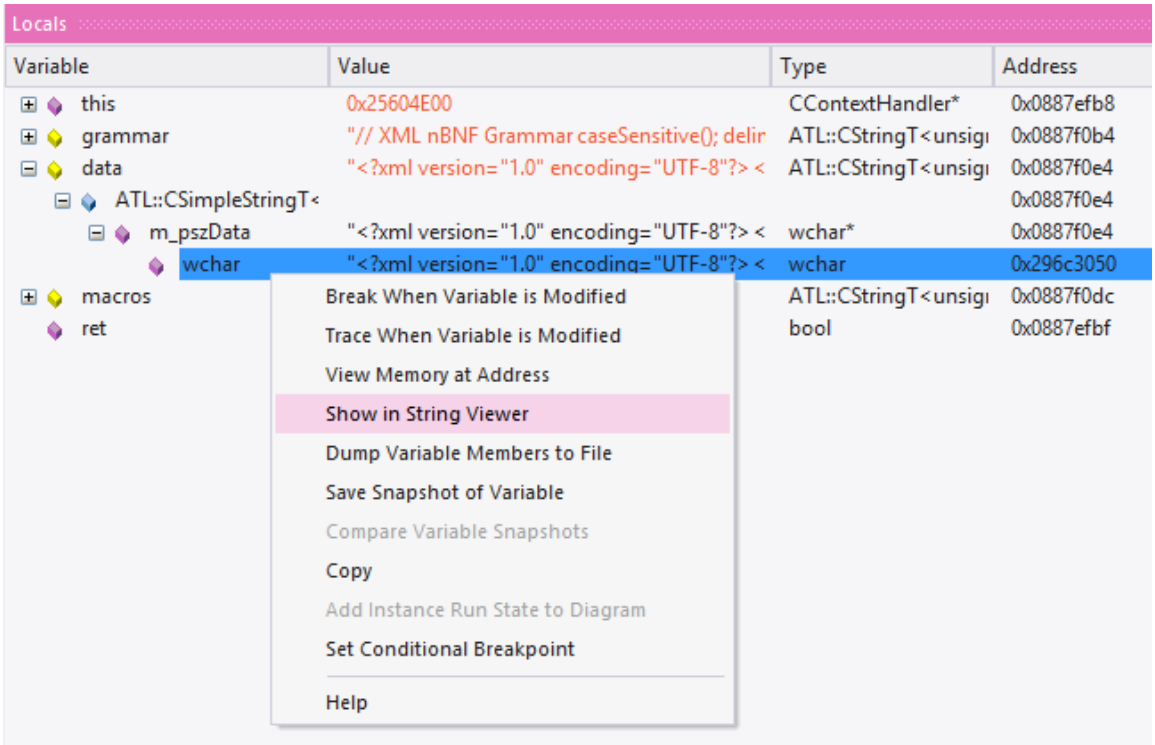
上下文菜单

功能	细节

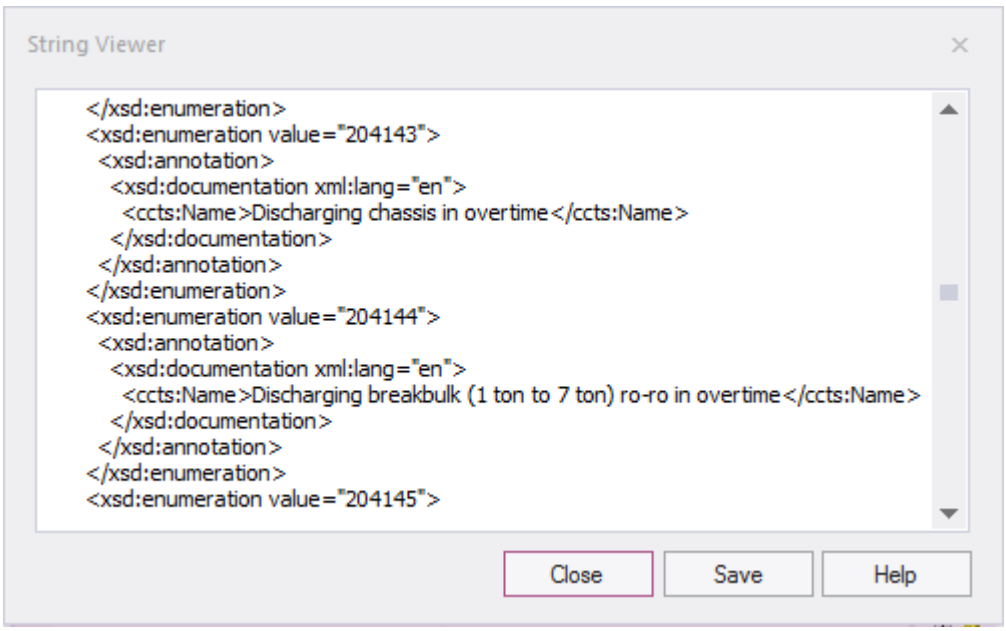
修改变量时中断	在选定的内存变量上设置数据断点，以在刚刚导致变量值更改的代码行处停止调试器的执行。
视图地址的内存	以十六进制和 ASCII 格式显示内存中所选地址的原始值。
在字符串查看器中显示	在 "字符串查看器"对话框中显示变量string 。
将变量成员转储到文件	捕获所选变量并将其存储到单独的位置；将显示一个浏览器以选择适当的 .txt 文件名和文件路径。
保存变量快照	在变量生命周期的特定时间点捕获变量的值。
比较变量快照	比较该变量生命周期中不同时间点的变量值。
复制	将所选变量复制到Enterprise Architect剪贴板。
将实例运行状态添加到图表	如果您打开了包含正在调试源代码的类的物件的模型图，此选项将使用变量值表示的运行状态更新该物件。
设置条件断点	在当前执行位置添加一个断点，并为此变量匹配其当前值。

视图长字符串的内容

为了提高效率，本地窗口窗口只显示部分字符串。但是，您可以使用 "字符串查看器"显示string变量的全部内容。



此示例显示了保存 XML 模式文件内容的变量的值。

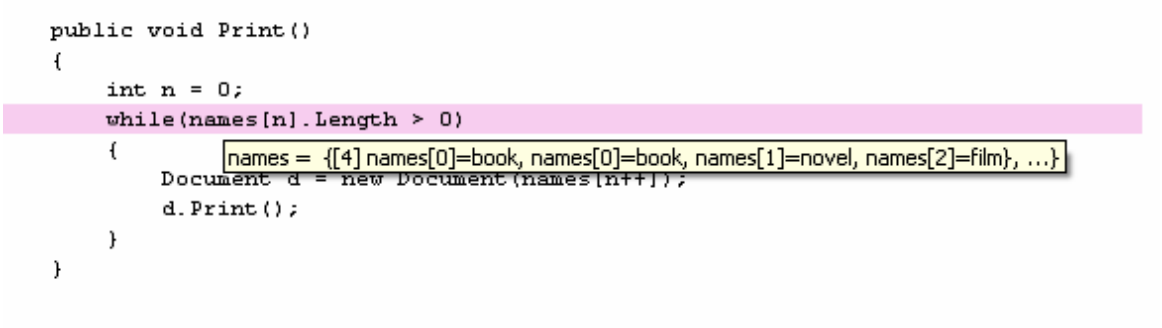


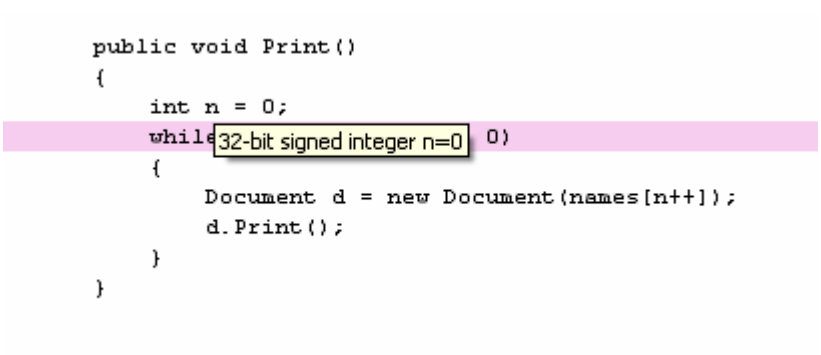
访问

从代码编辑器或本地窗口窗口	右键单击string变量 |在字符串查看器中显示

视图代码调试代码编辑器中的变量

发生断点时，您将在该窗口中看到所有局部变量。您还可以通过将鼠标悬停在引用上来检查源代码编辑器中的变量。这里有些例子。



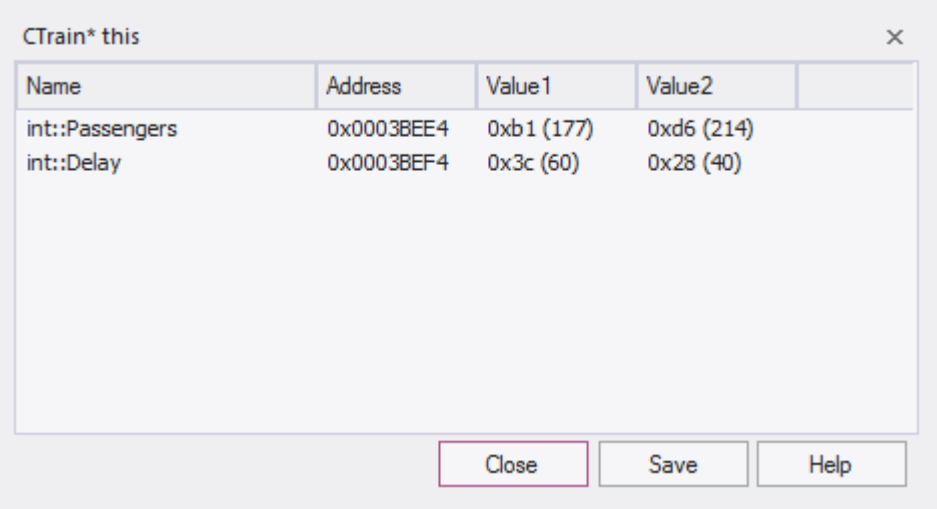


注记：变量不必是局部变量之一。它可以具有文件或模块范围。

可变快照

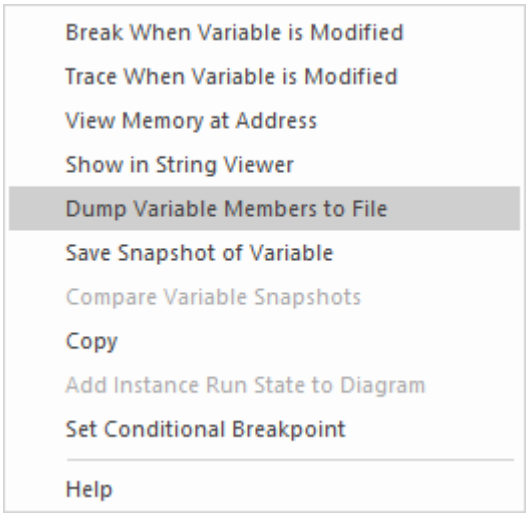
当您的程序遇到断点时，可以对变量进行 快照”，并使用此快照来查看变量的值在其生命周期的不同点是如何变化的。调试器不会只复制所选变量的值；对于复杂变量，它会复制所选变量的值及其每个成员层次结构的值，直到它不再为某个成员找到更多调试信息或找不到更多成员。

捕获变量快照

节	行动
1	在代码编辑器中，设置两个断点：一个在函数的开头，另一个在函数的结尾。
2	在开始断点处，右键单击本地窗口窗口中的变量并选择 保存变量快照”菜单选项。
3	运行应用程序。
4	当到达结束断点时，右键单击本地窗口窗口中的变量并选择 比较变量快照”选项。 将显示A对话框，其中显示了第一个快照的原始值和第二个快照的当前值，如从 EA 获取的此图中所示。示例模型。 

将变量快照保存到文件

您可以使用其右键单击上下文菜单将变量的状态保存到文件中。



这是文件内容的摘录。

```
73 00000006|0x00731F00|name|TObjectType::Type  |value|TypeIsStation|
74 00000005|0x00731F08|name|wchar::Name    |value|"Treasury"|
75 00000005|0x00731F0C|name|unsigned::Location |value|0x40 (64)|
76 00000003|0x0003BED8|name|float::Distance    |value|0|
77 00000003|0x0003BEE0|name|int::Capacity  |value|0x1f4 (500)|
78 00000003|0x0003BEE4|name|int::Passengers    |value|0xd6 (214)|
79 00000003|0x0003BEE8|name|unsigned::Number   |value|0x3 (3)|
80 00000003|0x0003BEF0|name|unsigned::Location |value|0x0 (0)|
81 00000003|0x0003BEF4|name|int::Delay |value|0x28 (40)|
```

行动点

行动点是可以执行动作的断点。遇到断点时，调试器会调用运行脚本，然后进程继续运行。行动点是复杂的调试工具，可为专业开发人员提供额外的命令套件。有了它们，开发人员可以改变函数的行为，捕捉行为改变的点，并修改/检测object的状态。为了支持这些特征，行动点可以改变原始局部变量和成员变量的值，可以定义自己的 "用户定义变量"并改变程序执行。

行动点和断点中的用户定义变量

用户定义的变量 (UDV)：

- 提供在 Actionpoint 语句中设置 UDV 原语或string的方法
- 可用于多个标记/断点的条件语句
- 可以在同一个局部变量窗口中轻松查看
- 调试结束时会记录所有 UDV 的最终值。

在 UDV 语法中，UDV 名称：

- 必须以 # (哈希) 字符开头
- 不区分大小写

行动点声明

Actionpoint 语句可以包含 set 命令、goto 命令和 jmp 命令。

设置命令

设置变量值。一个 Actionpoint 语句可以包含多个 "set"命令，所有这些命令都应该在任何 "goto"命令之前。

'set' 命令语法是：

设置 *LHS* = *RHS*

在哪里：

- **LHS** = 变量的名称为：
  - 用户定义变量 (UDV)，例如 #myval
  - 局部变量或成员变量，例如 strName 或 this.m_strName
- **RHS** = 要分配的值：
  - 作为文字或局部变量
  - 如果是文字，则为以下之一：整数、布尔值、浮点数、字符或string

set 命令 - 变量示例

UDV 示例	局部变量示例
设置 #mychar = 'a'	设置 this.m_nCount=0
设置#mystr = "一个string "	设置 bSuccess=false

设置#myint = 10	
设置#myfloat = 0.5	
设置#mytrue = true	

转到命令

此命令将执行切换到函数中的不同行号。 Actionpoint 语句只能包含一个 goto 命令，作为语句中的最后一个命令。

goto 命令语法为：

转到*L*

其中**L**是当前函数中的行号。

goto命令使用断点来实现其目标，这会导致代码执行稍有延迟。这在非常频繁执行的代码区域中可能很明显，因此您可能更喜欢在此类代码中使用 jmp 命令，以实现相同的执行转移但延迟更少。

jmp 命令

jmp命令实际上与goto命令相同。

跳转125

转到125

这两个命令都会导致执行更改为第 125 行。

然而， jmp语句在内部使用检测来指示程序移动执行，而goto语句使用断点来执行此操作，这会导致处理延迟。因此，区别在于jmp语句的卓越性能，尤其是在代码区域执行非常频繁的情况下。

整数符

如果存在用户定义变量 (UDV) 并且它的类型为int ，则可以使用 ++ 和 -- 运算符对其进行递增和递减。例如：

1. 创建一个 UDV 并将其值和类型设置为本地整数变量。
   AP1：设置#myint = nTotalSoFar
2. 增加 UDV。
   AP2：#myint++
3. 减少 UDV。
   AP3：#myint--

定时器操作

行动点可以报告两点之间经过的时间。只有一个定时器可用，可以通过 startTimer 命令重置或启动。然后可以使用 printTimer 命令打印当前经过的时间。最后，打印总经过时间并使用 endTimer 命令结束计时器。

示例动作点条件

使用文字和常量：

- (#mychar='a')
- (#mystr <> "")
- (#myint > 10)
- (#myfloat > 0.0)

使用局部变量：

- （#myval == this.m_strValue）
- (#myint <> this->m_nCount)
- (#myint != this->m_nCount)

指令记录

指令记录可用于检测已知行为的变化；执行点 (B) 与先前的执行 (s) (A) 不同。命令是：

- recStart - 开始录制或开始比较之前的录制是否存在
- recStop - 结束录制
- recPause - 暂停录音
- recResume - 恢复录制

recStart命令开始记录指令。然后存储执行的指令。当遇到**recStop**命令时，记录会被保存。两个行动点之间的任一时间只能保存一个录音。当遇到**recStart**并且存在先前的记录时，调试器将开始将每个后续指令与其记录进行比较。它可以执行许多比较。如果并且当检测到差异时，调试器将中断并且行为改变的代码行将显示在代码编辑器中。比较的迭代也被打印出来。

录音默认存储在内存中，但也可以使用命令语法将其存储到文件中：

recStart 文件规范

例如：

recStart c:\mylogs\onclickbutton.dat

当遇到指定文件的**recStart**命令并且该文件存在时，它将被加载到内存中并且调试器将立即进入比较模式。

表达式

Breakpoint、Actionpoint 和测试点条件表达式中没有隐含的优先级。在复杂表达式中，必须使用括号。请参阅以下示例：

类型	示例
Actionpoint UDV 示例	(#myint=1) 与 (#mystr="德国")
局部变量示例	(this.m_nCount > 10) 或 (nCount%1) (this.m_nCount > 10) 或 (bForce)
条件表达式中的等式运算符	<> - 不等于 != - 不等于 == - 相等 = - 相等
Actionpoint 中的赋值运算	= - 将 RHS 分配给 LHS

符	
条件表达式中的算法运算符	/ - 分配 + - 加号 - - 减 * - 乘法 % - 模量
条件表达式中的逻辑运算符	AND - 两者都必须为真 或 - 一个必须为真 && - 两者都必须为真 || - 一个必须是真的 ^ - 异或（只有一个必须为真）

视图的其它变量

访问

功能区	执行 >窗口> 手表
其它	执行分析器window工具栏：  |手表

视图

视图	描述
手表	Watches 窗口对于本机代码（C、C++、VB）最有用，它可用于评估不能作为局部变量使用的数据项 - 具有模块或文件范围的数据项和静态类成员项。 您还可以使用该窗口评估Java和.NET中的静态类成员项 要添加监视，请在工具栏中键入要监视的变量的名称，然后按 Enter 键。 要检查 C++、 Java或 Microsoft .NET中的静态类成员变量，请输入其完全限定名称： CMyClass::MyStaticVar 要检查具有模块或文件范围的 C++ 数据符号，只需输入其名称。 通过查看当前范围来评估变量；即当前栈帧所在的模块（可以在断点处双击帧中的帧调用堆栈范围）。 如果全局变量存在于不同的模块中，您可以通过在变量前面加上模块名称来检查变量 模块名！变量名 错误输入数据项名称很容易，因此如果您出错或发现错误，请突出显示string，按 F2 并重新输入文本。这还通过在发现匹配项时中断搜索来加快调试器中命名项的解析。
历史	输入项目的历史记录被保留。可以使用工具栏文本框中的向上箭头键和向下箭头键再次选择以前输入的名称或表达式。历史也将在同一台机器上的Enterprise Architect或模型的任何实例中为用户保留。

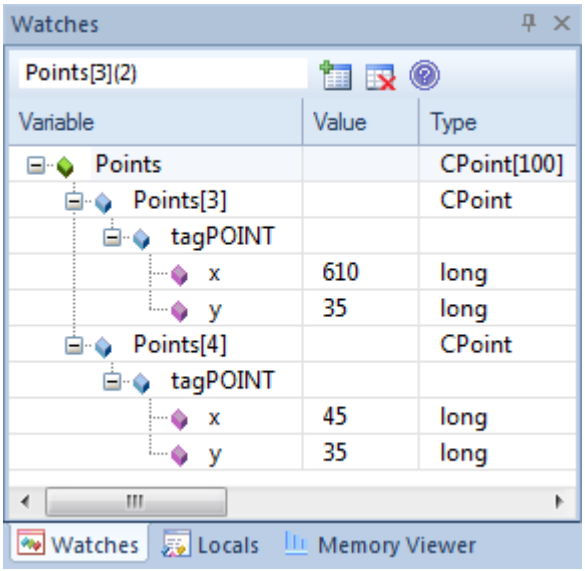
视图元素of Array

您可以使用 Watches 窗口检查数组的一个或多个特定元素。

在 Watches 窗口工具栏左侧的字段中，键入数组的变量名称，后跟开始元素和要显示的元素数。起始元素用方括号括起来，元素个数用括号括起来；那是：

变量[start_element](count_of_elements)

例如，Points[3](2) 显示 Points 数组的第四个和第五个元素，如图所示。



如果您输入 Points[3]，Watches 窗口将仅显示第三个数组元素。

访问

功能区	执行 >窗口> 手表
其它	执行分析器window工具栏： |手表

查看调用堆栈

调用堆栈窗口用于显示一个进程中当前正在运行的所有线程。它可用于在程序故障发生之前立即识别哪个线程正在运行。

当仿真处于活动状态时，调用堆栈会显示当前运行上下文的执行时间。这将包括每个并发模拟 线程"的单独上下文堆栈。

每当线程被挂起、通过其中A步骤操作或遇到断点时，都会显示堆栈跟踪。调用堆栈窗口可以记录堆栈变化的历史，并且可以根据这个历史生成序列图。

访问

功能区	执行 >窗口>调用堆栈
其它	执行分析器调用堆栈工具栏：  |执行分析器

使用到

- 视图堆栈历史以了解进程的执行
- 视图线程
- 保存调用堆栈以备后用
- 记录序列图生成的调用堆栈更改
- 从调用堆栈生成序列图
- 视图源代码编辑器中的相关代码行

功能

功能	描述
指标	• A红色箭头突出显示当前堆栈帧 • 蓝色箭头表示正在运行A线程 • 红色箭头表示正在记录堆栈跟踪历史记录A线程
将调用堆栈保存为 .TXT文件	目前不可用。
在调试会话中记录线程	要记录线程的执行并将记录指向Record &调用堆栈窗口，右键单击资源中的线程并选择适当的上下文选项： • 'Record' - 在调试会话期间手动记录当前线程 与调试器的 步骤"按钮一起使用；由于 step 命令而调用的每个函数都会记录到 Record & Analyze 窗口 • 'Auto-Record' - 在调试会话期间执行自动记录 当您选择此图标时，分析器开始记录并且在程序结束、停止调试器或单

	击 停止”图标之前不会停止
停止记录	如果您已经开始手动或自动记录线程，您可以在完成之前将其停止；选择线程（由红色箭头指示），然后： • 单击工具栏中的  （停止记录）按钮或 • 右键单击并选择 停止”选项
生成调用堆栈序列图表	要从调用堆栈轨迹生成序列图，可以： • 单击  （生成堆栈序列图表）按钮，或 • 右键单击并选择 生成序列图表”选项
将堆栈复制到记录历史	要立即将堆栈详细信息添加到 记录和分析”窗口（用于以后生成序列图），请执行以下任一操作： • 单击  按钮，或 • 右键单击并选择 将堆栈复制到记录历史记录”选项
切换堆栈深度	要在显示完整堆栈和仅显示带有源的帧之间切换，请单击  （切换堆栈深度）按钮。
在源代码编辑器中显示相关代码	双击一个线程/帧，在源代码编辑器中显示相关的代码行；局部变量也会为选定的帧刷新。

创造图表的序列调用堆栈

调用堆栈窗口记录了堆栈变化的历史，您可以从中生成序列图。

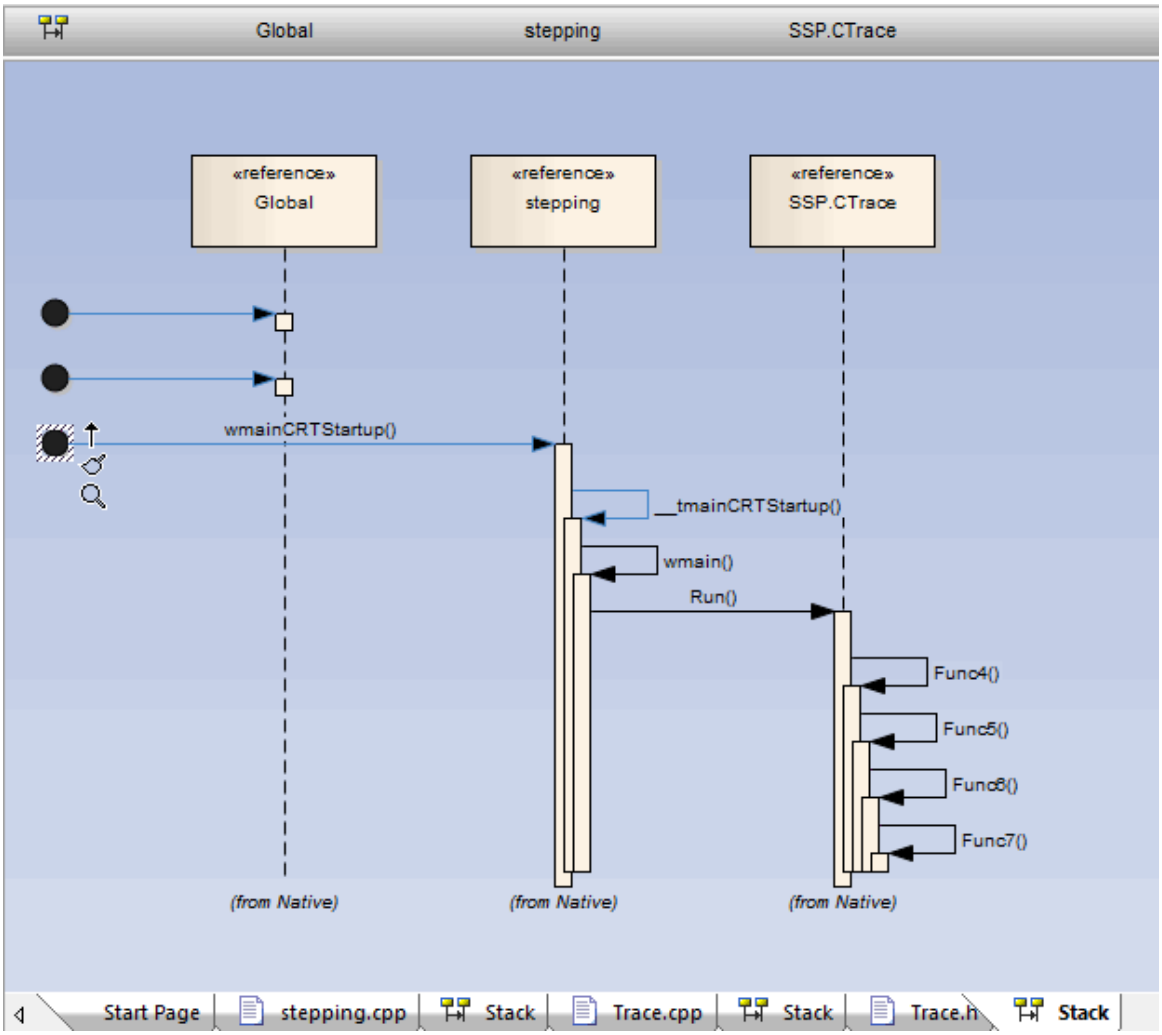
访问

功能区	执行 >窗口>调用堆栈
其它	执行分析器调用堆栈工具栏：  |执行分析器

使用到

- 记录调用堆栈图生成的序列
- 生成来自二调用堆栈的序列图

要生成序列图堆栈，请单击当前工具栏上的调用堆栈堆栈的  (生成序列图表) 按钮。
这会立即在图形图表视图中生成一个序列图。


Global
stepping
SSP.CTrace
«reference»
Global
«reference»
stepping
«reference»
SSP.CTrace
wmainCRTStartup()
__tmainCRTStartup()
wmain()
Run()
Func4()
Func5()
Func6()
Func7()
(from Native)
(from Native)
(from Native)
Start Page
stepping.cpp
Stack
Trace.cpp
Stack
Trace.h
Stack

检查进程内存

使用内存查看器，您可以以十六进制和 ASCII 格式显示内存的原始值。您可以在 地址”字段（右上角）中手动定义内存地址，或者右键单击本地窗口窗口或监视窗口中的变量并选择 视图地址处的内存”选项。

访问

功能区	执行 >窗口> 内存查看器
其它	执行分析器window工具栏：  |内存查看器 从本地窗口窗口或监视窗口：右键单击变量|视图地址处的内存

注记

- 内存查看器可用于调试在WINE窗口运行的 Microsoft 本机代码应用程序（C、C++、VB）

显示加载的模块

对于.NET和本机窗口应用程序，您可以使用 模块”窗口列出被调试进程加载的 DLL。此列表还可以包括调试器使用的关联符号文件（PDB 文件）。

访问

功能区	执行 >窗口> 模块

模块窗口显示

柱子	描述
小路	显示加载模块的文件路径。
加载地址	显示加载模块的基内存地址。
修改日期	显示本地文件日期和修改模块的时间。
调试符号	显示： ● 调试符号类型 ● 模块中是否存在调试信息，以及 ● 模块是否存在线路信息（调试时需要）
符号文件匹配	表示符号文件的有效性；如果值为false ，则符号文件已过期。
符号路径	显示符号文件的文件路径，它必须存在才能进行调试。
修改日期	显示创建符号文件的本地文件日期和时间。

处理首次异常

访问

功能区	执行 > 工具 >调试器>处理首次异常
其它	调试窗口工具栏：  |处理首次异常

加工元素

元素	描述
调试进程	当应用程序正在被调试并且调试器被通知异常时，应用程序被暂停并且调试器以它配置的方式响应；它要么： • 恢复应用程序并将异常留给应用程序管理，或 • 保持应用程序暂停并将异常传递给适当的例程以进行自动解决或手动干预
第二次机会例外	Enterprise Architect调试器默认为第一个列出的行为。 如果应用程序可以处理异常，则继续处理；如果它不能处理异常，调试器会再次收到通知，这一次它必须暂停应用程序并解决异常情况。 在这种行为中，因为调试器已经遇到过两次异常，所以称为第二次机会异常；在这种情况下，如果异常没有停止执行，它会被忽略，您可以避免将时间花在不影响处理整体结果的条件上。 您可能会在大型或复杂系统上以这种方式工作，这些系统总是在处理路径中的某处涉及异常条件。
第一次机会例外	但是，如果您想检查发生的每个异常，您可以将调试器设置为采用第二种行为。 因为调试器在第一次接触时响应异常，所以它被称为第一次机会异常。 您可能会以这种方式处理必须干净或根本不工作的单个函数或例程。
选择	选择 处理首次异常”选项以在第一次联系时调试异常。 取消选择仅当应用程序发生异常时才处理异常的选项。

即时调试器

您可以将Enterprise Architect调试器注册为操作系统即时调试器，以便在系统上的Enterprise Architect外部运行的应用程序遇到异常或崩溃时调用。当您这样做时，应用程序崩溃将导致Enterprise Architect被打开，并显示崩溃的源和原因。

访问

功能区	执行>工具>调试器>设置为JIT调试器

编译申请

本主题说明如何在Enterprise Architect中对您的应用程序执行编译脚本。

访问

功能区	执行>源>编译>编译
键盘快捷键	Ctrl+Shift+F12
其它	'编译'工具栏 >  执行分析器窗口 | 

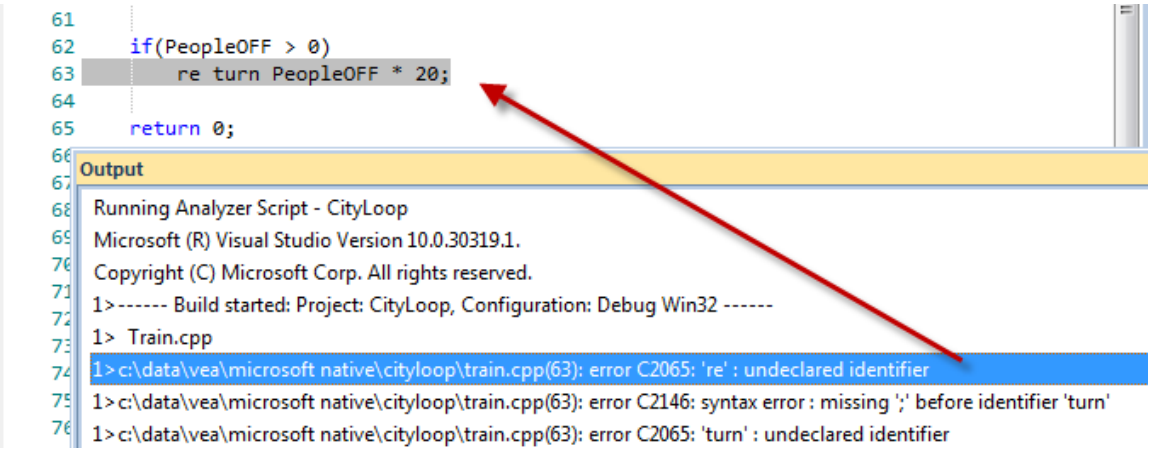
行动

当您选择 编译”选项时，它会在执行分析器窗口中选择的脚本中执行 编译”命令。构建操作的进度和结果显示在系统输出窗口的 编译”选项卡中。

您可以通过双击错误快速访问出现的任何编译错误的代码行。

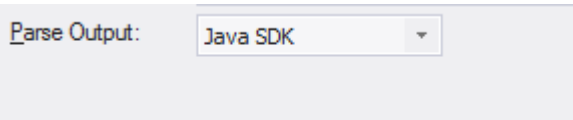
在代码中定位编译器错误

当您使用分析器脚本应用程序时，编译器输出会记录在系统输出窗口中。您可以双击此处出现的任何错误消息并转到源代码。当您这样做时，光标将位于包含错误的行上。



小费

如果缺少输出，请检查分析器分析器中是否提到了语言脚本（Shift+F12）。

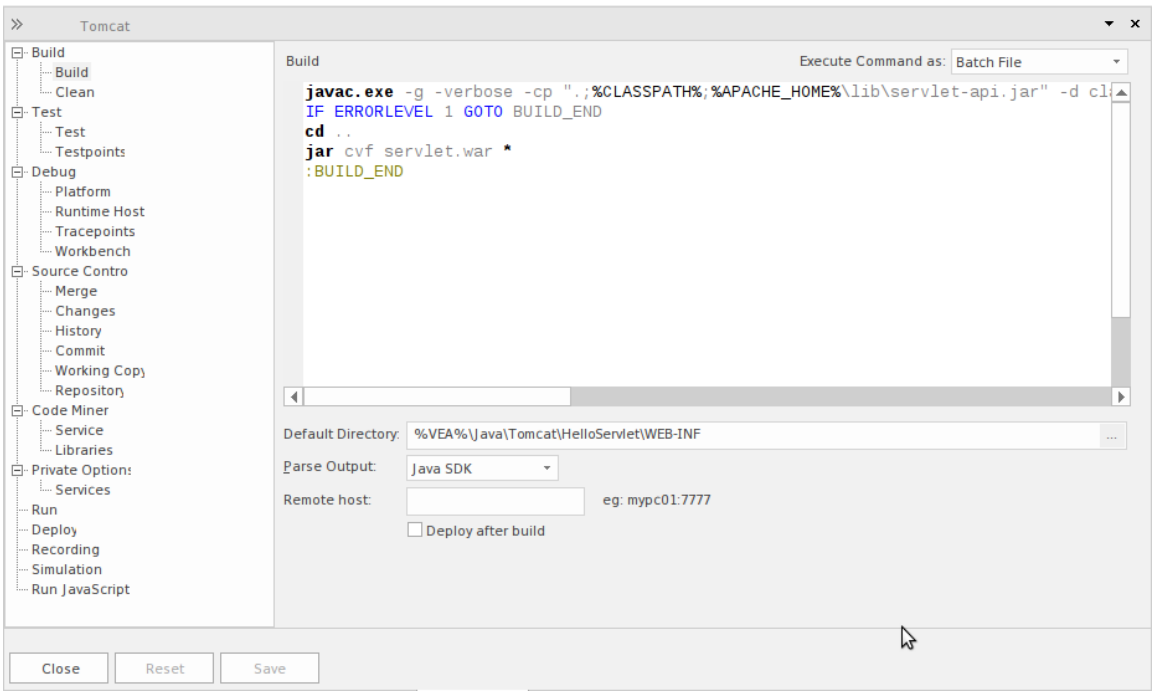


访问

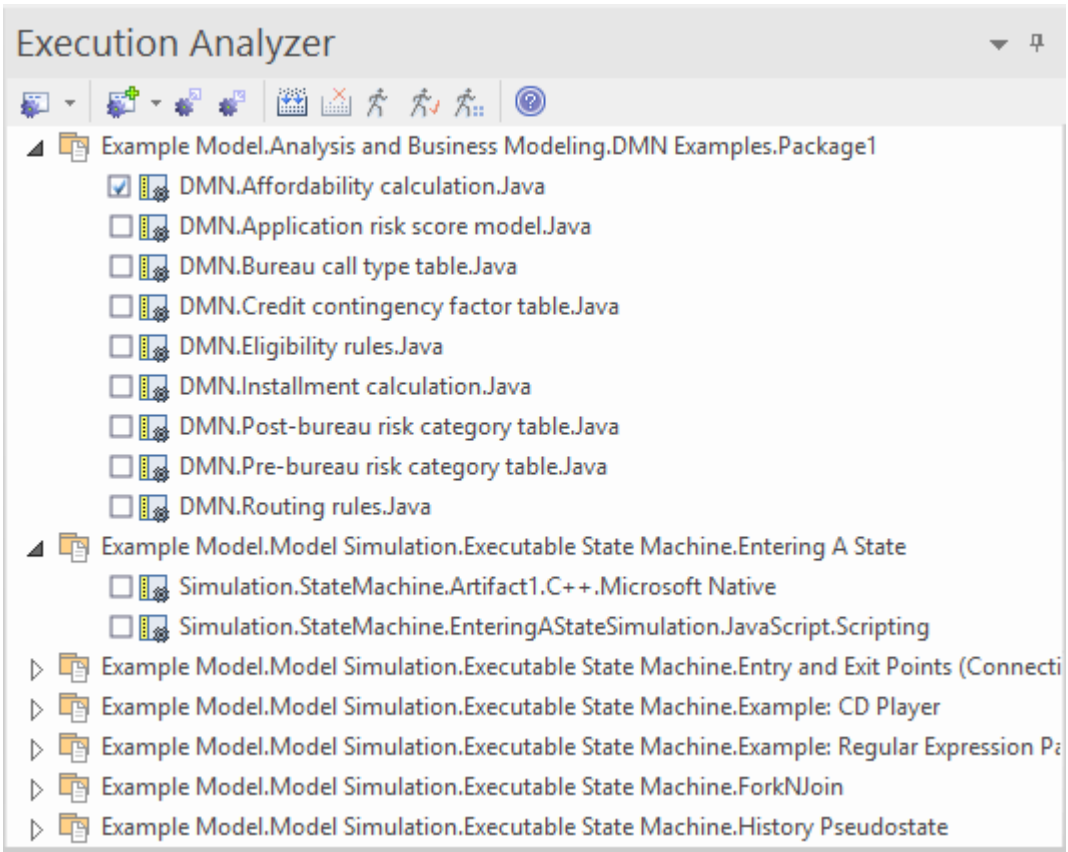
功能区	>开始> 所有窗口>设计> 探索 >系统
键盘快捷键	Ctrl+Shift+8

分析器脚本

分析器脚本由执行分析器使用。您无需担心创建这些。它们与JavaScript或 PHP 不是同一类型的脚本，而是使用熟悉的用户界面（树形视图）进行管理，并且您可以快速定位要更改的特征。分析器脚本可以由社区模型的用户共享并且是轻松导入和导出为 XML 文件。



A项目可以有多个配置，并且可以在分析器窗口中找到这些配置。



每个分析器脚本为一个包定义的，所以项目可以很愉快地共存。在许多组织中，管理系统的过程是分布式的，并且因人而异，因人而异。分析器Enterprise Architect模型中的脚本可以通过信任单个、共享和负责的过程为这些组织提供一些安心用于构建和部署任何种类的配置。脚本的所有方面都是可选的。例如，您可以在没有人的情况下进行调试；但是，只需几行代码，它们就可以启用这些有用的特征：

- 建造
- 测试
- 调试
- 记录
- 执行
- 部署
- 仿真

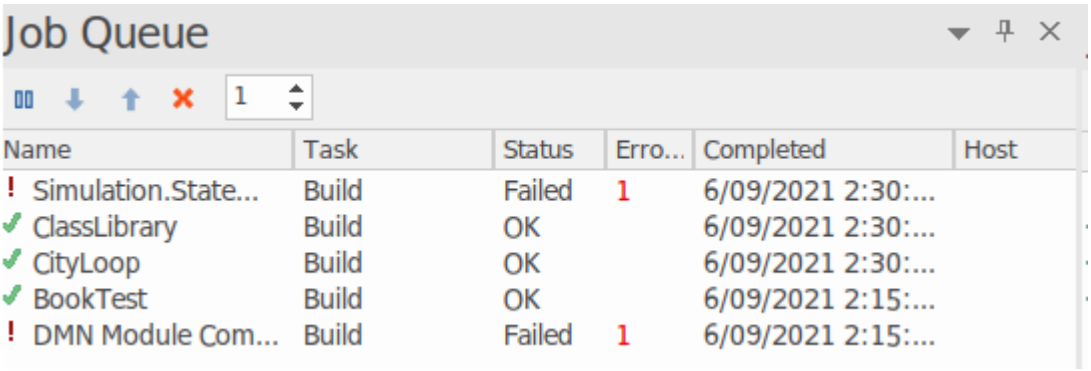
远程脚本执行

编译、运行等各种分析器脚本，提供了一个'Remote Host'字段。此字段用于描述脚本应在其上运行的运行。为了使用这个特征，Sparx卫星服务必须在机器上运行。该字段的格式是*hostname:port*，其中*hostname*是Linux机器的IP地址或网络名称， *port*是卫星服务正在监听的端口窗口号。此特征的主要目标是允许在 Linux 上运行的Enterprise Architect用户执行 Linux 原生的命令。

作业队列窗口

Job Queue 窗口简化了使用分析器脚本的过程，这些脚本最初是单独处理的，如果没有其他脚本正在执行。在 Enterprise Architect脚本以后的版本中，当一个分析器执行上下文菜单选项被执行时（例如，'编译'），它被放置在一个作业队列中；可以将多个作业排入队列，并在处理作业时执行其他工作。

以分析器的脚本作为作业名称。一个分析器脚本有多个部分，例如编译、测试、运行和部署，每个部分都被分配为作业的一个任务。



从作业队列窗口执行的每个作业的输出被捕获到系统输出窗口中的 作业历史”选项卡。

访问

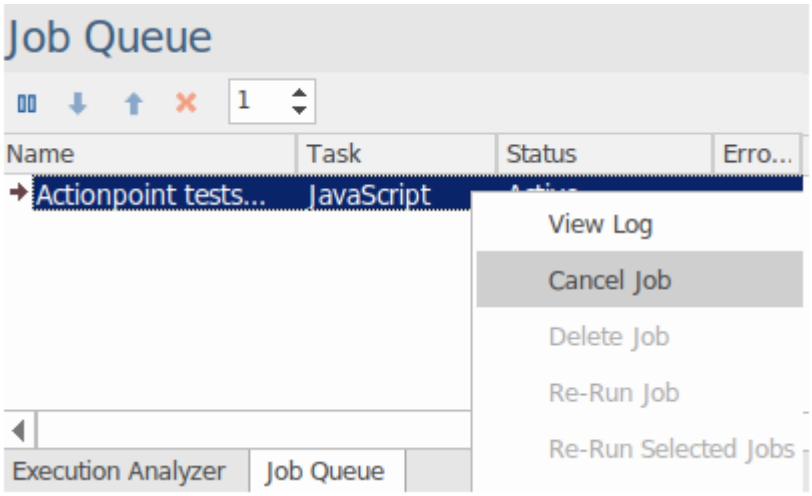
功能区	执行 > 工具 >分析器>视图作业队列 >开始> 所有窗口>设计> 探索系统工作经历

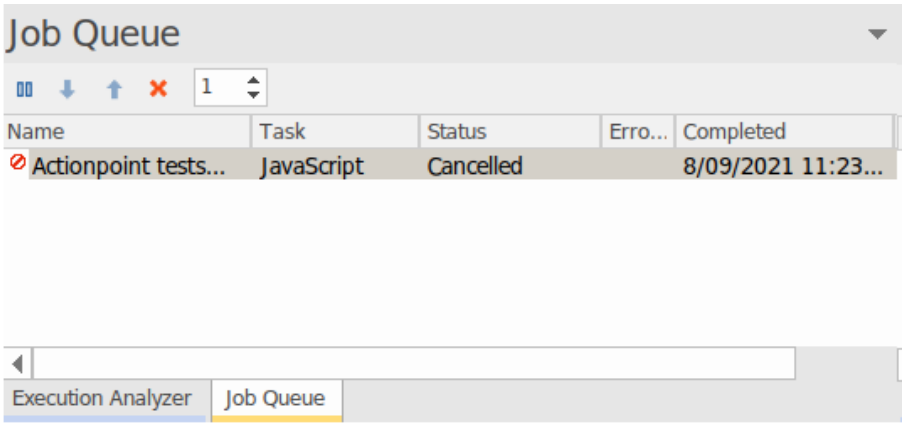
作业队列列

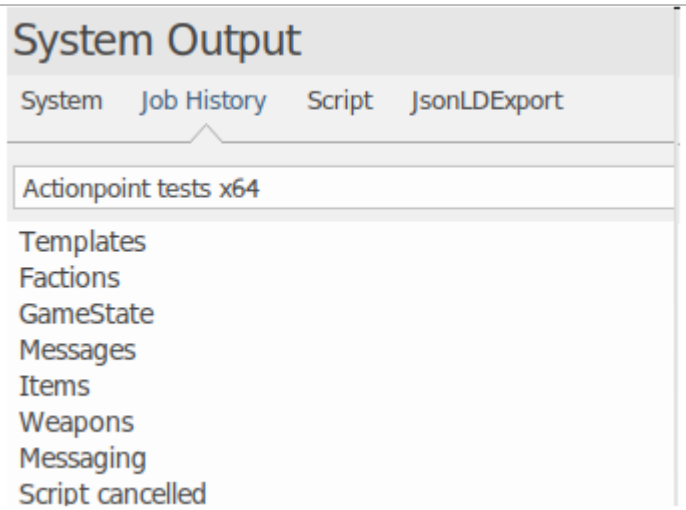
柱子	描述
名称	已将作业添加到作业脚本中的分析器名称。如果作业已执行，则名称前面将有一个勾号（表示成功完成）或感叹号（表示失败）。
任务	作业正在执行的分析器脚本- 例如，编译或部署。
状态	作业的完成状态- 作业是成功完成（'确定'）还是失败。
错误	如果作业已执行并失败，则出现的错误数。
完全的	作业完成的日期和时间。
主持人	如果作业正在远程运行，运行远程机器的 IP 地址或主机名。
接收	如果作业正在运行，则来自主机的任何返回消息。

上下文菜单选项

右键单击作业名称或窗口背景，以显示 作业队列"上下文菜单。

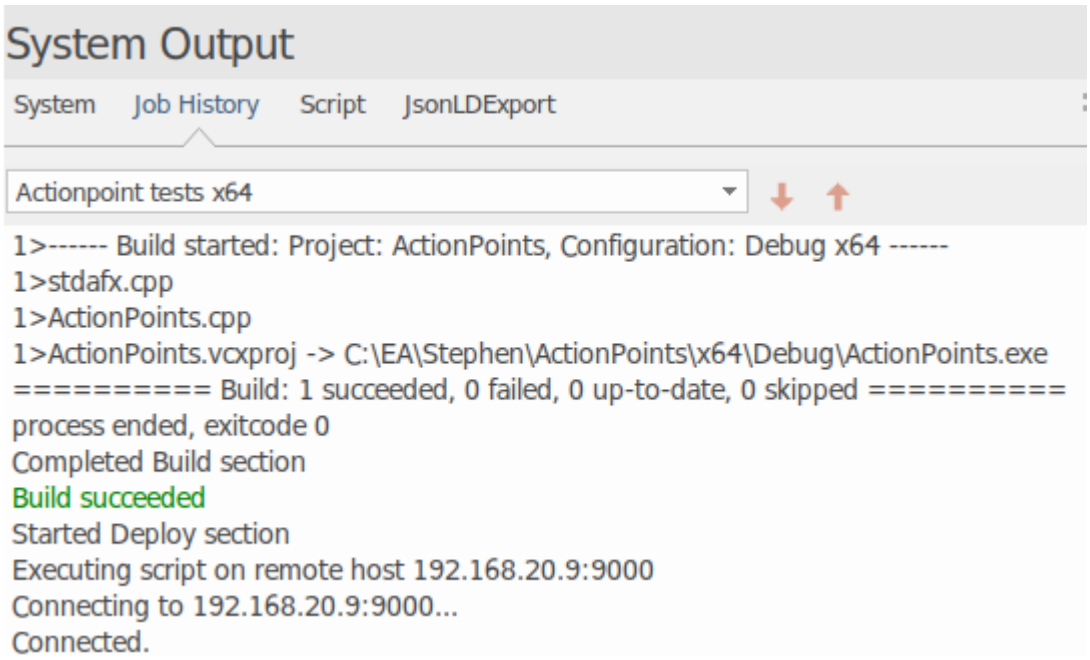


选项	描述
视图日志	选择此选项以显示系统输出窗口的 作业历史"选项卡，以显示已执行的作业。
取消作业	右键单击作业名称，然后根据需要随时单击此选项以取消选定的JavaScript任务。 取消在作业队列窗口中由作业名称旁边的 无条目"图标指示。  在系统输出窗口的 作业历史"选项卡中，输出以 脚本已取消"消息终止。

	
删除工作	单击尚未开始的作业并选择此选项以将其从作业队列中删除。
重新运行作业	选择此选项可再次执行选定的已完成作业。
重新运行选定的作业	（按住 Ctrl 并单击多个所需的作业。）选择此选项可再次执行所有选定的已完成作业。
重新运行已完成的作业	选择此选项可再次执行当前列表中的所有已完成作业。

作业历史选项卡

从作业队列窗口执行的每个作业的输出被捕获到系统输出窗口中的 作业历史”选项卡。从那里，您可以自行决定查看任何作业log ，方法是从工具栏的下拉列表中选择它。如果作业失败并且有错误消息，您可以使用红色箭头图标从一条消息跳到另一条消息。如果没有活动的错误消息，这些图标将被禁用。



作业队列工具栏选项

选项	描述
	单击作业名称并单击此图标可暂停或恢复所选作业。
	单击作业名称和这两个箭头之一以在作业队列中向上或向下移动作业，使其按处理顺序更早或更晚。
	单击尚未开始的作业的作业名称，然后单击此图标可从 作业队列"窗口中删除该作业。
	单击向上或向下箭头可设置可运行运行的作业数，最多为 8 个。 计数默认为1 ，以便 Job Queue 一次处理一个作业，First In First输出。

代码矿工脚本

代码矿工系统使用一组数据库来提供对从现有源代码派生的信息的快速和全面的访问。 Enterprise Architect的代码编辑器的智能感知特征及其搜索工具可以利用从这些数据库中挖掘的信息。

通过代码矿工脚本页面，您可以指定用于特定项目的代码矿工数据库，您可以创建、更新和添加新数据库到代码矿工库。 服务"页面允许您指定本地代码矿工库，或者您希望通过 Sparx 英特尔服务访问可用的库。

每个分析器可以指定不同的代码矿工脚本，所以使用的代码矿工库由激活的分析器脚本。

访问

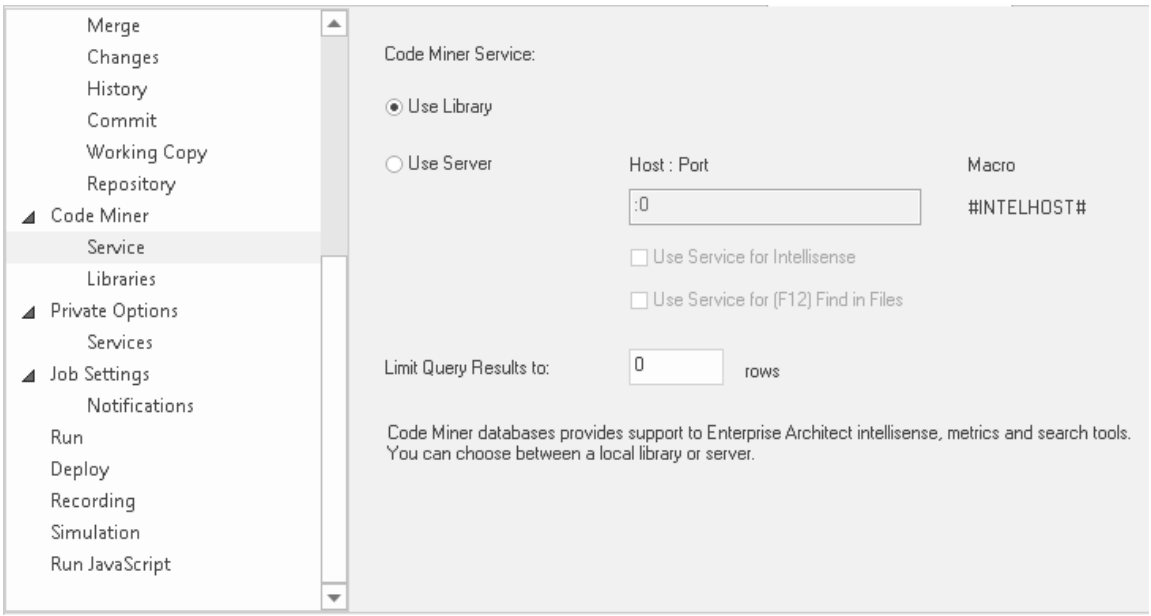
在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 代码矿工"页面或
- 点击窗口工具栏中的 ，选择要新建脚本的包，选择 代码矿工"页面

功能区	执行>工具>分析器>视图分析器脚本>双击脚本名称>代码矿工>服务 开发>源代码>执行分析器>编辑分析器脚本>双击脚本名称>代码矿工>服务
键盘快捷键	Shift+F12

Sparx 英特尔服务

代码矿工A可以在本地使用，也可以部署到可以为多个客户端提供服务的服务器位置。您在分析器脚本的 代码矿工脚本"页面上选择要使用的分析器。



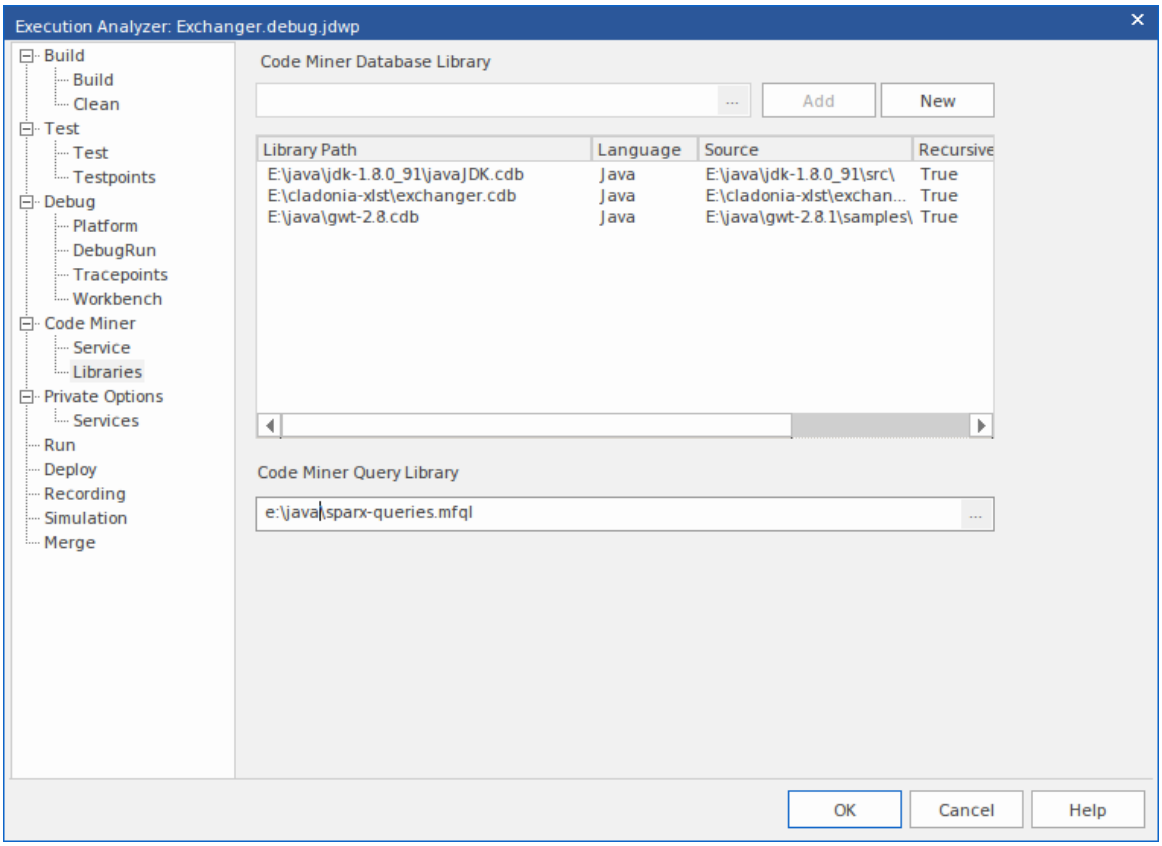
代码矿工库

代码矿工数据库库是包含从源代码派生的信息的数据库集合。一起，这些代码矿工库由Enterprise Architect特征

智能感知数据库使用。通常，将为每个框架或项目创建一个库。代码矿工库页面允许创建新数据库，以及从库中添加、更新或删除现有数据库。

代码矿工查询库是一组函数，用代码矿工的 mFQL 语言编写，捆绑到一个源文件中。

给定分析器的数据库代码矿工库和查询库在 代码矿工脚本”中指定。脚本编辑的图书馆页面。



服务脚本

分析器的 服务”页面描述了各种脚本可视化执行分析器（导入项目、生成可执行状态机）创建脚本时使用的默认端口。您可以在此页面上更新任何端口规范。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 私人选项 |服务页面或
- 单击窗口工具栏中的  ，选择要在其中创建新脚本的包，然后选择 Private Options |服务页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Shift+F12

合并脚本

分析器脚本中A分析器脚本为用户提供了执行某些操作的附加命令。合并操作取决于您的要求。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 "合并"页面或
- 点击窗口工具栏中的  ，选择要创建新脚本的包，然后选择'Merge'页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行>源>合并
键盘快捷键	Ctrl+Alt+M

记录脚本

录音的美妙之处并不在于我们总能看到更大的画面，而是有机会看到一个有一些真相的小画面。我们都见过不太有用的序列图。（同一条消息在图表上连续出现 *100* 次确实可以告诉我们一些信息，但不多。）幸运的是，Enterprise Architect通过使用片段来处理第一点。重复行为被标识为模式，并在序列图上以片段的形式表示一次。该片段根据迭代次数进行标记。当然，记录历史总是显示整个历史。我们还需要工具来帮助我们聚焦特定感兴趣区域的记录并减少其他人的噪音。我们可以使用过滤器来做到这一点。使用过滤器，您可以从任何记录中排除任何类、函数甚至模块。您可以创建多组过滤器并将它们与标记集一起使用以针对不同的使用案例。

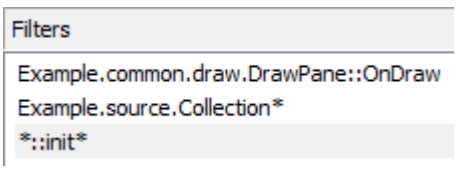
访问

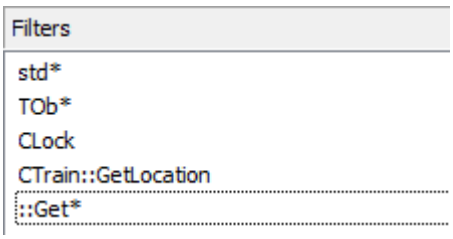
在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 记录”页面或
- 点击窗口工具栏中的  ，选择要新建脚本的包，然后选择 记录”页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Shift+F12

过滤器Strings

元素	讨论
过滤	如果在执行分析器脚本器的 记录”页面上选中了 启用过滤器”复选框，则调试器将从记录中排除对匹配方法的调用。比较区分大小写。 要添加值，请单击 排除过滤器”框右上角的 新建”（ 插入”）图标，然后输入比较string ；每个过滤器string采用以下形式： class_name_token::method_name_token class_name_token 排除了对名称与令牌匹配的类或类的所有方法的调用；string可以包含通配符 *（星号）。 method_name_token 不包括对名称与令牌匹配的方法的调用；同样， string可以包含通配符 *。 两个令牌都是可选的；如果没有类标记，则过滤器仅应用于全局或公共函数（即不属于任何类的方法）。
示例	在此Java示例中，调试器将排除： • 调用类示例的 OnDraw 方法 • 调用任何名称以示例.源.Collection 开头的类的任何方法 • 调用任何类的任何构造函数（例如 <clint> 和 <init>） 

	在本机代码示例中，调试器将排除： • 对标准模板库命名空间的调用 • 对任何以类开头的类的调用 • 调用类CLock的任何方法 • 调用类CTrain的GetLocation方法 • 调用名称以 Get 开头的任何全局或公共函数 

过滤器

使用过滤器Entry	到过滤器
：：得到*	录制会话中名称以 "Get" 开头的所有公共函数（例如，窗口中的 GetClientRect ）。
：：得到	任何类中以 "Get" 开头的所有方法。
C类::获取*	CClass类的所有以 Get 开头的方法。
C类::*	CClass类的所有方法。
ATL* 标准*	属于标准模板和活动模板库的类的所有方法。
CClass::GetName	CClass类的特定方法 GetName。

部署脚本

这些部分解释了如何创建用于部署当前包的命令脚本。可以通过选择 执行>源>编译>部署”功能区选项或按 Ctrl+Shift+Alt+F12 来执行脚本。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 部署”页面或
- 点击窗口工具栏中的  ，选择要在其中创建新脚本的包，然后选择 部署”页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Ctrl+Shift+Alt+F12

行动

行动	细节
执行命令为：	**Process** If the deployment is handled externally, enter the path to the program or batch file to run, followed by any parameters; the program is launched in a separate process. Example: C:\apache-ant-1.7.1\bin\ant.cmd myproject deploy **Batch File** When using this option, you can enter multiple commands that are then executed as a single script in a command console; you have access to any environment variables available in a standard command console. Example: @echo on IF NOT EXIST "%1%" GOTO DEPLOY_NOWAR IF "%APACHE_HOME%" == "" GOTO DEPLOY_NOAPACHE xcopy /L "%1%" "%APACHE_HOME%\webapps" GOTO DEPLOY_END rem rem NO WAR FILE rem :DEPLOY_NOWAR echo "%1% WAR file not found" GOTO DEPLOY_END rem

	rem NO APACHE ENVIRONMENT VARIABLE rem :DEPLOY_NOAPACHE echo "APACHE_HOME environment variable not found" :DEPLOY_END pause
解析输出	从列表中选择解析器会导致部署脚本的输出被捕获；根据从列表中选择的语法解析输出。 要显示系统输出窗口，请选择 开始>所有窗口>设计>探索>系统"功能区选项。

运行脚本

本节介绍如何创建用于运行可执行代码的命令。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 运行”页面或
- 点击窗口工具栏中的  ，选择要新建脚本的包，选择 运行”页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Ctrl+Alt+N

脚本元素

元素	描述
命令	这是当您选择 执行>运行>开始”功能区选项时执行的命令；在最简单的情况下，脚本将包含要运行的文件的位置和名称。
例子	这两个示例显示了配置为在Enterprise Architect中运行和Java应用程序的脚本。 。网： C:\benchmark\cpp\example_net_1\release\example.exe Java ： 顾客 此字段中列出的命令就像来自命令提示一样执行；因此，如果可执行路径或任何参数包含空格，它们必须用引号括起来。

注记

- Enterprise Architect提供了正常启动应用程序或从同一脚本进行调试的能力； '分析器' 菜单有单独的启动正常运行和调试运行的选项

调试脚本

配置分析器的调试部分的过程脚本是一次性的，很少需要重新访问。因此，一旦您的脚本正常工作，您可能就不必再考虑它了。您提供的细节并不复杂，但定义脚本提供了许多好处，例如：

- 调试
- 序列图记录
- 可执行状态机执行与模拟
- 测试域创作和记录
- 各种运行时进程的行为分析

您需要做的就是选择合适的平台并输入一些基本细节。您可以使用的调试器平台包括：

- Java
- Java调试Wire Protocol (JDWP)
- 微软.NET调试器
- Microsoft Native Code调试器(C++, C, VB)
- 单核细胞增多症
- PHP调试器
- GNU调试器(GDB)

访问

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	F6

注记

- 调试Enterprise Architect模型脚本（例如JavaScript或 VBScript）不需要分析器脚本

运营系统具体需求

Enterprise Architect调试器能够在许多不同的平台上运行。此表描述了在每个平台上进行调试的个别要求。

平台

平台	细节
微软.NET	• Microsoft™ .NET框架 4.0、3.5 和 2.0 • 语言支持：C、C#、C++、J#、VB.NET
Java	• Oracle™ 的Java SE 开发工具包（最低版本 5.0）（32 位或 64 位 JDK） Java平台调试器架构(JPDA) 是在Java SE 5.0 版中引入的。 JPDA 提供了两种调试协议； Java虚拟机工具 (接口) 和Java接口调试连线协议 (JDWP)。 Enterprise Architect的调试器支持这两种协议。
GNU调试器(GDB)	Enterprise Architect支持使用 GNU调试器进行调试，它使您能够在 Linux 下本地或远程调试您的应用程序。 需要 GDB 7.0 或更高版本。 源代码文件路径不能包含空格。
原生应用程序窗口	Enterprise Architect支持调试使用 Microsoft™ 编译器编译的本地代码（C、C++ 和 Visual Basic），其中关联的 PDB 文件可用。
PHP	Enterprise Architect使您能够在 Web 服务器中执行 PHP 脚本的本地和远程调试。 需要将 Web 服务器配置为支持 PHP。 需要将 PHP 配置为支持 XDebug PHP（第 3 方 PHP 扩展）。

注记

- Enterprise Architect的所有版本都提供调试功能

支持 UAC 的操作系统

微软系统窗口7提供用户账户控件（UAC）来管理应用程序的安全性。

Enterprise Architect可视化执行分析器是 UAC 兼容的，启用 UAC 的系统的用户可以在只有用户组成员的帐户下使用可视化执行分析器和相关功能执行操作。

但是，当附加到在启用 UAC 的操作系统上作为服务运行的进程时，可能需要以管理员身份log 。

以管理员身份登录

节	行动
1	在运行Enterprise Architect之前，右键单击桌面上的Enterprise Architect图标并选择 以运行身份运行”选项。

或者

编辑或创建指向Enterprise Architect的链接，并配置指向以运行身份运行的链接。

节	行动
1	右键单击Enterprise Architect图标并选择 属性”选项。 显示Enterprise Architect的 属性”对话框。
2	单击高级按钮。 显示 高级属性”对话框。
3	选中 以运行身份运行”复选框。
4	单击确定按钮，然后再次单击 Enterprise Architect属性”对话框。

WINE 调试

配置在WINE下调试Enterprise Architect

节	行动
1	在命令行中，运行$运行。
2	选择 应用程序”选项卡。从Enterprise Architect安装文件夹中添加Enterprise Architect可执行文件“EA.exe”。然 从 VEA 子目录添加这些程序： • SSampler32.exe • SSampler64.exe • SSProfiler32.exe • SSProfiler64.exe
3	依次选择每个程序，然后切换到 库”选项卡。确保以（本机、内置）优先级列出这些值： • 数据库帮助 • msxml4 • msxml6
4	将应用程序源代码和可执行文件复制到您的瓶子中。 路径必须与编译后的版本一致；那是： 如果窗口源= C:\源\SampleApp，Crossover 下必须是C:\源\SampleApp。
5	复制应用程序使用的任何 Side-By-Side 程序集。

权限

Enterprise Architect的安装包含一些本地 Linux 程序，这些程序在Wine下为Enterprise Architect提供构建和调试服务。这些程序需要使用 Linux 文件系统或 shell 进行检查，以确保它们具有适当的 执行”权限设置。这些程序位于Enterprise Architect安装的 ѰEA/x86/linux”子目录中。

访问违规异常

由于WINE处理直接绘制和访问 DIB 数据的方式，在调试窗口工具栏的下拉菜单中提供了一个附加选项，用于忽略或处理程序直接访问 DIB 数据时引发的访问冲突异常。

选择此选项可捕获真正的（意外）访问违规；取消选择它以忽略预期的违规行为。

由于调试器无法区分预期和意外违规，您可能必须使用试错法来捕获和检查真正的程序崩溃。

注记

- 如果WINE崩溃，后面的痕迹可能不正确

- 如果您使用 MFC，请记住将调试并行程序集复制到 C:\window\winsxs 目录
- 要将 windows 路径添加到WINE ，请修改注册表项：
  HKEY_LOCAL_MACHINE\系统\CurrentControlSet\控件\会话管理器\环境

Java

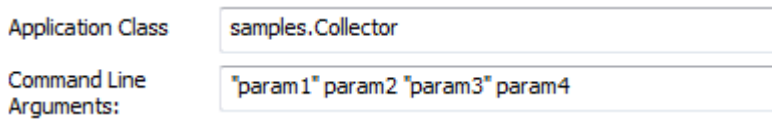
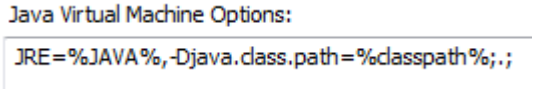
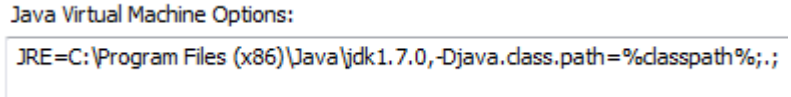
本节介绍如何设置Enterprise Architect以调试Java应用程序和网络服务器。

Java的常规设置

调试Java应用程序的一般设置支持两个选项：

- 调试应用
- 附加到正在运行的应用程序

选项1 -调试应用程序

字段	行动
调试器	选择Java 。
x64	如果您正在调试 64 位应用程序，请选中此复选框。 如果您正在调试 32 位应用程序，请取消选中该复选框。
模式	选择运行。
默认目录	创建Java虚拟机时，此路径会添加到类路径属性中。
应用类	确定要调试的全限定类名；类必须有一个用这个签名声明的方法： public static void main(字符串()); 
命令行参数	指定要传递给 Application类的 main 方法的任何参数。 包含空格的参数应该用双引号括起来。
Java虚拟机选项	指定用于创建虚拟机的命令行选项。 您还必须为Java运行时环境(JRE) 提供一个参数作为搜索 jvm.dll 的路径；这是 Sun Microsystems ™作为运行时环境或 JDK 的一部分提供的 DLL。 JRE 参数可以是： • 企业架构师定义的本地路径 • 用于调试的Java JDK 安装文件夹的绝对文件路径（不带双引号） JRE 参数必须指向Java JDK 的安装文件夹。要成功调试，必须A JDK。 JRE 不应指向公共Java运行时环境（如果已安装）的安装。在指定 VM 启动选项时可以使用环境变量，例如类路径。 例如，使用： • Enterprise Architect Local Path JAVA 和环境变量类路径：  • 或者 JDK 安装目录的绝对路径和环境变量类路径： 

	在这两个示例中，调试器将使用位于 JRE 参数值的 JDK 创建一个虚拟机。 如果未指定类路径，则调试器始终创建虚拟机，其类路径属性等于环境变量中包含的任何路径加上在此脚本的默认工作目录中输入的路径。 如果源文件和 .class 文件位于不同的目录树下，classpath属性必须包括源文件的根路径和二进制类文件的根路径。

选项 2 - 附加到虚拟机

附加到 VM 时几乎不需要指定；但是，VM 必须加载Sparx Systems调试代理。

字段	行动
调试器	选择Java
模式	选择附加到虚拟机

高级技术

除了标准的Java调试技术，您还可以：

- Attach to Virtual Machine
- Internet Browser Java Applets

附加到虚拟机

您可以通过附加到托管Java虚拟机的进程来调试Java应用程序；您可能希望这样做以附加到 Tomcat 或 JBOSS 等网络服务器。

Sun Microsystems 的Java Virtual Machine Tools接口是Enterprise Architect使用的 API；它允许在创建 JVM 时指定调试代理。

要从Enterprise Architect调试正在运行的 JVM，必须在启动时将Sparx Systems的调试代理指定为 JVM 的启动选项； Tomcat 和 JBOSS 等产品如何实现这一点应由该产品自己的文档提供。

对于 java.exe，加载Enterprise Architect调试代理的命令行选项可能是（取决于您的环境）：

- -agentpath:"c:\程序文件\sparx 系统\ea\VEA\x86\SSJavaProfiler32"
- -agentpath:"c:\程序文件 (x86)\sparx 系统\ea\VEA\x86\SSJavaProfiler32"
- -agentpath:"c:\程序文件 (x86)\sparx 系统\ea\VEA\x64\SSJavaProfiler64"

适当的选项取决于您的操作系统以及您是在使用 32 位应用程序还是 64 位应用程序。

或者，如果将适当的 VEA 目录添加到 PATH 环境变量中，则可以选择使用：

- -agentlib:SSJavaProfiler32
- -agentlib:SSJavaProfiler64

附加到虚拟机时无需配置分析器脚本您可以只使用分析器工具栏之一上的附加按钮。

如果你配置一个分析器脚本那么只有两件事是必须选择的：

- 选择 "Java "作为调试平台
- 选择 附加到虚拟机"选项

互联网浏览器Java Applets

本主题描述了从Enterprise Architect调试在浏览器中运行的Java Applet 的配置要求和过程。

从Enterprise Architect附加到托管Java虚拟机 (JVM) 的浏览器进程

节	行动
1	确保要调试的小程序代码的二进制文件已使用调试信息构建。
2	使用Java控件面板配置JVM。
3	在 "Java Applet 运行时设置"面板中，单击视图按钮。
4	在要使用的已安装版本上，在 "运行时参数"字段中包含以下选项之一，具体取决于您的环境以及您使用的是 32 位应用程序还是 64 位应用程序： -agentpath:"c:\程序文件\sparx 系统\ea\VEA\x86\SSJavaProfiler32" -agentpath:"c:\程序文件 (x86)\sparx 系统\ea\VEA\x86\SSJavaProfiler32" -agentpath:"c:\程序文件 (x86)\sparx 系统\ea\VEA\x64\SSJavaProfiler64"
5	在此字段中添加所需的类路径。 这些路径中至少有一个应包含用于调试的源文件的根路径。
6	设置断点。
7	启动浏览器。
8	从Enterprise Architect附加到浏览器进程。

使用Java网络服务器

如果您在Enterprise Architect中调试 JBOSS 和 Apache Tomcat 等Java Web 服务器（包括服务器配置和窗口服务配置），请应用这些配置要求和过程。

注记：Oracle 的Java服务器平台 "Weblogic"不支持可视化执行分析器的调试和记录特征。

从Enterprise Architect附加到托管Java虚拟机的进程

节	行动
1	用于调试 Web 服务器代码的编译二进制文件，带有调试信息。
2	使用 "虚拟机启动"选项启动服务器，如服务器配置中所述。
3	导入源代码导入Enterprise Architect模型，或同步现有代码。
4	设置断点。
5	启动客户端。
6	从Enterprise Architect附加到流程。

服务器配置

Web 服务器与Enterprise Architect交互所需的配置必须解决这两个基本点：

- 任何要由服务器调试、创建或托管的 VM 都必须指定Sparx Systems代理人命令行选项或在 VM 启动选项中指定（即：
  -agentlib:SSJavaProfiler32 或 -agentlib:SSJavaProfiler64)
- CLASSPATH，不管是传给VM的，都必须指定包源文件的根路径

Enterprise Architect调试器使用被调试VM 中的类属性，定位执行过程中发生的断点对应的源文件；例如，要调试的类称为：

abC

这位于物理目录中：

C:\源\a\b

因此，要成功调试，CLASSPATH 必须包含根路径：

c:\源

分析器脚本配置

使用 "编译"脚本的 "调试"选项卡，为您导入的代码创建一个脚本，然后：

- 选择 "附加到进程"单选按钮，然后在其下方的字段中输入 "附加"
- 在 "使用调试器"字段中，单击下拉箭头并选择 "Java"

所有其他字段都不重要； "目录"字段通常在没有任何类路径属性的情况下使用。

运行调试器

断点可能会显示一个问号。在这种情况下，类可能尚未被 VM 加载。如果即使在您确定包含断点的类已加载后问号仍然存在，那么：

- 服务器正在执行的二进制文件不是基于源代码的
- 调试器无法将断点协调到源文件（检查类路径），或者
- JVM 尚未加载Sparx Systems代理

节	行动
1	运行服务器并检查服务器进程是否已加载Sparx Systems代理人： DLL SSJavaProfiler32.DLL 或 SSJavaProfiler64 使用'使用'或类似的工具来证明服务端进程已经加载了代理。
2	在Enterprise Architect中，打开源代码并设置一些断点。
3	点击Enterprise Architect中的运行调试按钮。 将显示 附加到进程”对话框。
4	选择托管应用程序的服务器进程。
5	点击确定按钮。 调试窗口中会显示A确认消息，说明该进程已附加。

JBOSS服务器

在这个 JBoss 示例中，对于 32 位应用程序，简单 servlet 的源代码位于目录位置：

C:\Benchmark\ Java \JBOSS\Inventory

JBOSS 执行的二进制文件位于以下位置的 JAW.EAR 文件中：

C:\JBOSS\03b-dao\build\distribution

Enterprise Architect调试器必须能够在调试期间定位源文件；为此，它还使用 CLASSPATH，在任何列出的路径中搜索匹配的 JAVA源文件，因此 CLASSPATH 必须包含包根的路径，以便Enterprise Architect在调试期间找到源。

这是执行 JBOSS 服务器的命令文件的摘录；要调试的类在：

com/库存/dto/carDTO

因此，该路径的根被包含在JBOSS_CLASSPATH 中。

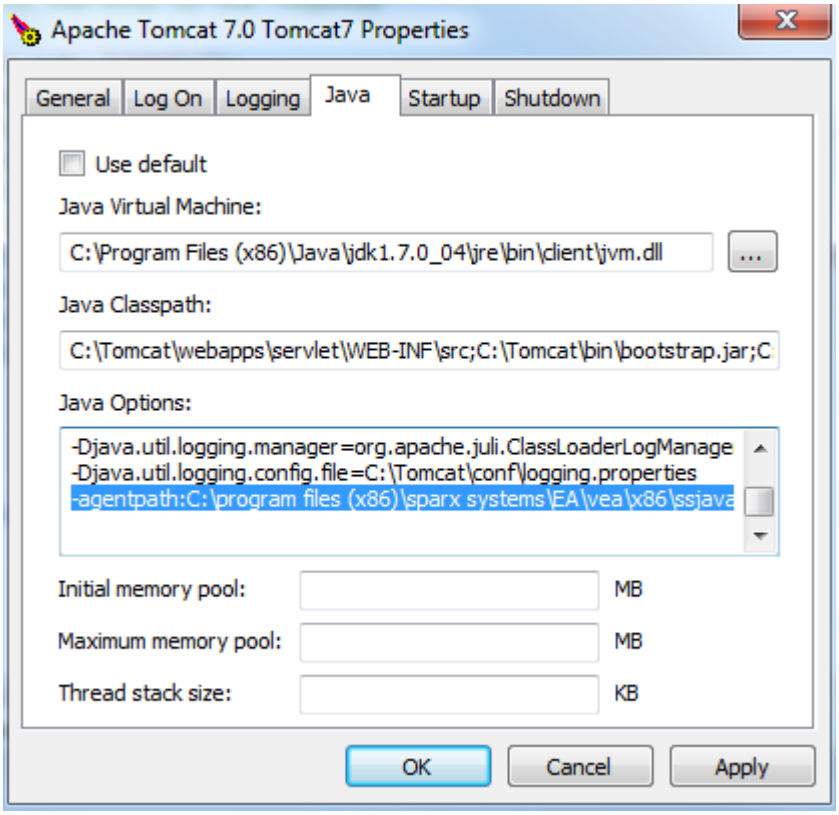
示例代码

RUN.BAT

```
set SOURCE=C:\Benchmark\Java\JBOSS\Inventory
set JAVAC_JAR=%JAVA_HOME%\lib\tools.jar
if "%JBOSS_CLASSPATH%" == ""
(
set JBOSS_CLASSPATH=%SOURCE%;%JAVAC_JAR%;%RUNJAR%;
)
else
(
set JBOSS_CLASSPATH=%SOURCE%;%JBOSS_CLASSPATH%;%JAVAC_JAR%;%RUNJAR%;
)
set JAVA_OPTS=%JAVA_OPTS% -agentpath:"c:\program files\sparx systems\vea\x86\ssjavaprofiler32"
```

Apache Tomcat服务器

Apache Tomcat服务器可以配置为使用Enterprise Architect中的Java调试器进行调试。此示例显示了运行窗口的 PC 上 Apache Tomcat 7.0 的配置对话框。



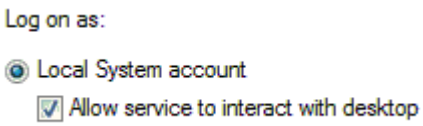
这三点很重要：

- “Java虚拟机”指定安装Java JDK 的运行时
- 将要调试的任何 servlet 的源路径添加到Java Classpath；在这种情况下，我们将路径添加到 Tomcat servlet：c:\tomcat\webapps\servlet\WEB-INF\src
- “Java选项”包括Sparx Systems调试代理的路径：
  -agentpath:c:\程序文件 (x86)\sparx 系统\vea\x86\ssjavaprofiler32

Apache Tomcat窗口服务

配置

对于将 Apache Tomcat作为窗口服务运行的用户，配置该服务以启用与桌面的交互非常重要；不这样做会导致在Enterprise Architect中调试失败。



选中 允许服务与桌面交互”复选框。

.NET

本节介绍如何配置Enterprise Architect以调试.NET应用程序。这包括：

- General Setup for .NET
- Debugging an Unmanaged Application
- Debug COM Interop
- Debug ASP .NET

.NET的常规设置

这是调试 Microsoft .NET应用程序的一般设置。调试时有两种选择：

- 调试一个应用程序
- 附加到正在运行的应用程序

选项1 -调试应用程序

字段	行动
调试器	选择 Microsoft .NET作为调试平台。
x64	如果您正在调试 64 位应用程序，请选中此复选框。 如果您正在调试 32 位应用程序，请取消选中该复选框。
模式	选择运行单选按钮。
默认目录	这被设置为正在调试的进程的默认目录。
申请途径	选择并输入应用程序可执行文件的完整或相对路径。 • 如果路径包含空格，则指定完整路径；不要使用相对路径 • 如果路径包含空格，则路径必须用引号引起来
命令行参数	在启动时传递给应用程序的参数。
显示控制台	为调试器创建一个控制台窗口；不适用于附加到进程。
符号搜索路径	指定任何其他路径来定位调试器的调试符号；用分号分隔路径。

选项 2 - 附加到正在运行的应用程序

字段	行动
调试器	选择 Microsoft .NET作为调试平台。
x64	如果您正在调试 64 位应用程序，请选中此复选框。 如果您正在调试 32 位应用程序，请取消选中该复选框。
模式	选择附加到进程单选按钮。

调试非托管应用程序

如果使用非托管应用程序调试托管代码，调试器可能无法检测到要加载的公共语言运行时 (CLR) 的正确版本。
如果您还没有在脚本的调试命令中指定的调试应用程序的配置文件，您应该指定一个配置文件。
配置文件应与您的应用程序位于同一目录中，并采用以下格式：
名称.exe.config
其中 "名称"是您的应用程序的名称。
您指定的 CLR 版本应与由调试对象调用的托管代码加载的版本相匹配。

在下面的示例代码中，"clr_version"表示您的插件或 COM 代码所针对的 CLR 的版本。

示例配置文件

```
<configuration>
    <startup>
        <requiredRuntime version="clr_version"/>
    </startup>
</configuration>
```

调试COM互操作

Enterprise Architect使您能够在本地或进程内服务器中调试使用 COM 执行的.NET托管代码。
此特征对于调试插件和 ActiveX 组件很有用。

调试使用 COM 执行的.NET托管代码

节	行动
1	在Enterprise Architect中创建一个包并导入代码进行调试。
2	确保 COM 组件是使用调试信息构建的。
3	为包创建脚本。
4	在'调试|平台”页面，您可以选择附加到非托管进程或指定非托管应用程序的路径以调用您的托管代码。
5	在源代码中添加断点进行调试。

附加到非托管进程

如果您正在使用：

- 进程内 COM 服务器，附加到客户端进程
- 本地 COM服务器A附加到服务器进程

单击调试窗口运行按钮（或按 F6）以显示可供选择的进程列表。

注记

- 从您一直在调试的 COM 互操作进程中分离会终止该进程；这是 Microsoft .NET Framework 的一个已知问题，有关它的信息可以在许多 MSDN .NET博客上找到

调试ASP .NET

对 ASP 等 Web 服务的调试要求Enterprise Architect调试器能够附加到正在运行的服务。

首先确保包含 ASP .NET服务项目的目录已导入Enterprise Architect ，如果需要，包含客户端网页的 web 文件夹。

如果您的 web 项目目录位于网站托管目录下，您可以从根目录导入，同时包含 ASP 代码和网页。

必须先启动客户端，因为 ASP .NET服务进程可能尚未运行；使用浏览器加载客户端 - 这可确保 Web 服务器正在运行。

然后在调试设置中选择 附加”单选按钮。选择此选项后，调试器每次都会提示您要调试的进程。

点击调试窗口运行按钮启动调试器；将显示 附加到进程”对话框。

如*ASP .NET SDK*中所述，进程的名称因 Microsoft 操作系统而异；例如，在窗口上，进程的名称类似于 aspnet_wp.exe，尽管该名称可以反映它所支持的.NET框架的版本。

XP下可以有多个ASP.NET进程运行；您必须确保附加到正确的版本，这将是托管您的应用程序运行的.NET框架版本的版本；检查 Web 服务的 web.config 文件以验证它所绑定的.NET框架的版本。

调试窗口停止按钮应该被启用并且任何断点应该是红色的，表明它们已经被绑定。

您可以随时在 Web 服务器代码中设置断点。如果您导入它们，您还可以在 ASP 网页中设置断点。

注记

一些断点可能没有成功绑定，但如果根本没有绑定（用带问号的深红色表示），则某些断点不同步；尝试重建并重新导入源代码

Mono调试器

Mono 是一个由.NET基金会赞助的软件平台，用于促进跨平台开发。它因其丰富的游戏、基于 API 和特征的特点而受到游戏开发者的欢迎。

Enterprise Architect通过为建模和开发软件提供现代环境来为 Mono 社区提供支持。现有项目可以在 Linux 和窗口上本地导入、构建和调试。

概述

Mono下的调试涉及到三个进程的配合。 Mono 运行时管理应用程序并使用套接字协议与Enterprise Architect调试器通信，后者又与作为前端的Enterprise Architect通信。当您需要指示它以支持调试时，您可以使用命令行指令来实现，在该指令中您命名主机并启动 Mono 应该监听的端口号。主机可以省略，在这种情况下，Mono 将接受来自任何 IP 地址的连接。主机可以具有值 ' localhost ' 来限制与同一台机器的连接。端口号码是您选择的号码。

主机和端口号是重要的信息，在配置分析器脚本使用。

用于需求窗口

- Enterprise Architect （最低版本 14 ）
- 用于窗口的 Mono（最低版本 5.4 ）

Linux需求

- Enterprise Architect （最低版本 14 ）
- 适用于 Linux 的 Mono（最低版本 5.4 ）
- 适用于 Linux 的Wine

运行时主机页面

此页面是可选的，仅在 Mono 和Enterprise Architect将在同一台机器上运行时才有用。它提供了在Enterprise Architect调试器启动之前首先使用所需的调试指令运行的能力。调试器连接后，它会恢复 Mono 运行时，该运行时已以挂起状态启动。如果应用程序在与您正在使用的Enterprise Architect不同的机器上运行，您应该清除此部分。

调试配置Linux

调试器配置

本节介绍Linux下调试脚本的分析器的调试部分。此处未列出的字段不是必需的。

调试器	选择 单声道”。
默认目录	这是应用程序所在的 Unix 格式的完全限定的本机 Linux 路径。
联系	• port：调试端口 • host：运行 Mono 的机器的名称或 IP 地址（如果机器相同，则为 ' localhost ' ） • localpath：窗口格式源代码的根路径；这是用于在Enterprise Architect的代码编辑器中设置断点的源文件的路径 • remotepath：Unix格式源代码的根路径，这是Linux下构建程序的源文件的路径 这些路径在调试事件期间返回，然后映射到本地路径，以便Enterprise Architect可以在断点或步骤期间显示源文件 - 两个参数可以指定相同的物理源文件根，但必须使用窗口或 Unix每个字段的格式 • 关机：（ true or false ）；当调试器停止时，VM 被终止 • timeout：套接字调用的超时时间（以毫秒为单位） • 输出：要写入的log文件的Wine /窗口路径 • 日志记录：（ true or false ）；如果为 true，则在调试窗口中记录额外的消息，并将套接字消息记录到指定的输出文件中

自动启动单声道

您可以将Enterprise Architect配置为在您启动调试器时为您启动 Mono。你可以通过配置你的分析器的'Runtime脚本'页面来做到这一点。命令的格式如下所述：

cd 程序路径

/usr/bin/mono --debug --debugger-agent=transport=dt_socket,address= *host:port* ,server=y,suspend=y程序

在哪里：

- *path-to-program*是程序所在的目录路径

- 主机是其中之一：

- localhost
- IP 地址
- 联网机器名称

- 端口是端口的端口

- *program*是应用程序的名称（例如 MonoProgram.exe）

使用命令行手动启动 Mono

您可以从控制台手动启动 Mono。在文件资源管理器中找到该程序，然后在该位置打开控制台。命令行的格式如下所述：

/usr/bin/mono --debug --debugger-agent=transport=dt_socket,address= *host:port* ,server=y,suspend=y程序

其中主机是其中之一：

- localhost
- IP 地址
- 联网机器名称

*port*是套接字的端口， *program*是应用程序的名称（例如端口）。

调试配置窗口

调试器配置

本节介绍一个分析器脚本关于在窗口下调试 Mono 的调试部分。此处未列出的字段不是必需的。

字段	描述
调试器	选择 单声道”。
x64	选择要调试的程序是否为 64 位可执行文件。
运行或附加	选择 迩行”来命名要启动的程序。如果您将始终附加到正在运行的进程，请选择 附加”。
默认目录	程序运行时将采用的默认目录。
申请途径	Mono 应用程序的完整路径。
命令行参数	要传递给程序的任何参数。如果参数包含空格，请用双引号 (") 将它们括起来

自动启动单声道

您可以将Enterprise Architect配置为在您启动调试器时为您启动 Mono。你可以通过配置你的分析器的'Runtime脚本'页面来做到这一点。命令的格式如下所述：

cd程序路径

mono --debug --debugger-agent=transport=dt_socket,address= *host:port* ,server=y,suspend=y程序

在哪里：

- *path-to-program*是程序所在的目录路径

- 主机是其中之一：

- localhost
- IP 地址
- 联网机器名称

- 端口是端口的端口

- *program*是应用程序的名称（例如 MonoProgram.exe）

使用命令行手动启动 Mono

您可以从控制台手动启动 Mono。在文件资源管理器中找到该程序，然后在该位置打开控制台。命令行的格式如下所述：

mono --debug --debugger-agent=transport=dt_socket,address= *host:port* ,server=y,suspend=y程序

其中主机是其中之一：

- localhost
- IP 地址

- 联网机器名称

*port*是套接字的端口， *program*是应用程序的名称（例如端口）。

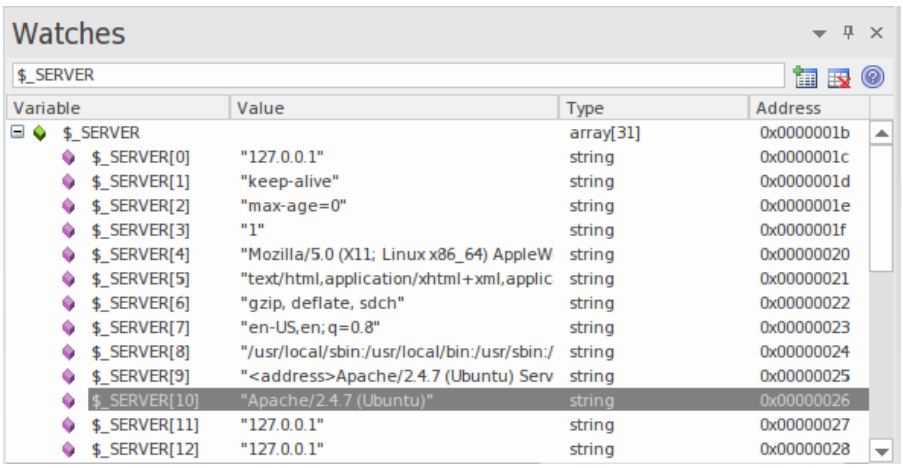
PHP调试器

Enterprise Architect PHP调试器器使您能够调试 PHP.exe 脚本。本节讨论基本设置和常见的各种调试场景；这些场景涉及文件路径的映射，这对于远程调试会话的成功至关重要。

- 脚本Setup
- 本地窗口机器（Apache服务器）
- 本地窗口机 (PHP.exe)
- 远程 Linux 机器（Apache服务器）
- 远程 Linux 机器 (PHP.exe)

设置和场景

设想	细节
脚本Setup	分析器脚本Enterprise Architect中调试的基本要求；您使用执行分析器的工具栏创建一个脚本。 选择PHP.XDebug作为调试平台；当您选择此平台时，属性页面会显示以下连接设置： • host - localhost - Enterprise Architect监听来自 PHP 的传入连接的适配器 • localpath - %LOCAL% - 指定要映射到远程文件路径的本地文件路径；这是一个远程调试设置 - 对于本地调试，清除该值，该值是一个占位符，您应该对其进行编辑以适合您的特定场景 • remotepath - %REMOTE% - 指定本地文件路径要映射到的远程文件路径；这是一个远程调试设置 - 对于本地调试，清除该值，该值是一个占位符，您应该对其进行编辑以适合您的特定场景 • logging - 输入true or false以启用来自 XDebug 服务器的通信记录 • 输出 - 命名远程机器上要与日志记录选项一起使用的文件路径；此文件将始终被覆盖
本地机器 Apache服务器	在这种情况下，请考虑以下配置： • O ： S • 网络计算机名称：MyPC • 网络共享 MyShare 映射到 c:\myshare • Enterprise Architect中的源文件已从 c:\myshare\apache\myapp\scripts 导入 • Apache 文档根设置为 //MyPC/MyShare/apache 在这种情况下，连接参数的分析器脚本可能配置为： • 主机： localhost • 端口：9000 • 本地路径：c:\myshare\apache\ • 远程路径：MyPC/MyShare/apache/
本地机器 PHP.EXE	在这种情况下，连接参数的分析器脚本可能会被配置为如图所示，因为文件路径总是映射到相同的物理路径： • 主机： localhost • 端口：9000 • 本地路径：

	• 远程路径：
远程 Linux 机器 Apache服务器	在这种情况下，请考虑以下配置： 本地机器： • O ： S • Enterprise Architect中的源文件已从 c:\myshare\apache\myapp\scripts 导入 远程机器： • O ： S • Apache文档根设置为家/apache/htdocs • Apache 中的源文件位于家/apache/htdocs/myapp/scripts 在这种情况下，连接参数的分析器脚本可能配置为： • 主机： localhost • 端口：9000 • 本地路径：c:\myshare\apache\ • 远程路径：家/apache/htdocs/
远程 Linux 机器 PHP.exe	在这种情况下，请考虑以下配置： • 本地机器 • O ： S • Enterprise Architect中的源文件已从 c:\myshare\apache\myapp\scripts 导入 • 远程机器 • O ： S • Apache 中的源文件位于家/myapp/scripts 在这种情况下，连接参数的分析器脚本可能配置为： • 主机： localhost • 端口：9000 • 本地路径：c:\myshare\apache\ • 远程路径：家/
PHP 全局变量	当您处于断点时，您可以使用分析器Watches 窗口检查 PHP 全局变量的值。要列出每个全局变量，请在字段中输入 "globals"或 "superglobals"。要显示单个项目，请输入其名称。此图像显示正在显示的 PHP 环境变量 $_SERVER 的值。 

PHP调试器-系统需求

本主题确定Enterprise Architect PHP 调试器的系统要求和操作系统。

系统需求：

- Enterprise Architect版本 9
- PHP 5.3 或以上版本
- PHP zend 扩展 XDebug 1或以上
- 对于 Apache 等 Web 服务器，支持 PHP 版本的服务器版本

支持的操作系统：

- 客户（ Enterprise Architect ）
- Microsoft窗口及以上版本
- Linux 运行 Crossover Office
- 服务器(PHP)
- Microsoft窗口及以上版本
- Linux

PHP调试器检查清单

本主题提供了在Enterprise Architect中调试 PHP 脚本的故障排除指南。

选择积分

选择点	细节
系统需求	• Apache HTTP网络服务器版本 2.2 • PHP 5.3 或以上版本 • XDebug 版本 2. 1 . 1
Enterprise Architect	• 该模型有一个分析器配置为使用PHP脚本平台 • PHP源代码已导入模型（用于记录和测试点） • 当从 "分析器"对话框中选择 PHP脚本平台时，默认运行时设置会在 "连接"字段中列出： 本地路径：%LOCAL% 远程路径：%REMOTE% 为这些默认变量定义本地路径或编辑脚本以提供实际路径。 例如：本地源、远程源 本地路径：c:\code samples\vea\php\sample 远程路径：网络服务器/示例 • 'webserver' 是网络或本地共享 • 'sample' 是共享下面的文件夹
PHP	为了在Enterprise Architect中调试 PHP 脚本，需要正确配置 PHP 以加载 XDebug 扩展。 应使用与这些类似的设置： • [xdebug] • xdebug.extended_info=1 • xdebug.idekey=ea • xdebug.remote_enable=1 • xdebug.remote_handler=dbgp • xdebug.remote_autostart=1 • xdebug.remote_host=XXXX • xdebug.remote_port=9000 • xdebug.show_local_vars=1 IP 地址 XXXX 指并应与模型分析器脚本指定的主机相匹配。 IP 地址是 XDebug 连接的地址，也是Enterprise Architect PHP 代理监听的地址。
阿帕奇	对于使用 Apache 进行调试，这些行应该存在于 Apache 配置文件 httpd.conf 中： LoadModule php5_module "php_home/php5apache2_2.dll" AddHandler 应用程序/x-httpd-php .php

	PHPIniDir "php_home" 值“php_home”是PHP安装路径（php.ini和apache dll存在的路径）。
故障排除	为了防止调试会话期间 PHP 和 Apache 超时，这些设置可能需要修改。 在Enterprise Architect中开发 PHP调试代理时使用了这些设置。
PHP	文件：php.ini ; Enterprise Architect在调试 PHP 扩展时防止 PHP 超时 max_execution_time = 0 ; Enterprise Architect在调试 PHP 扩展时防止 Web 服务器超时 最大输入时间 = -1 ; Enterprise Architect记录错误 display_errors = 开 ; Enterprise Architect显示启动错误 display_startup_errors = 开
阿帕奇	文件：httpd.conf ; Enterprise Architect在调试 php 扩展时防止超时 超时 60000

GNU调试器(GDB)

在调试您的应用程序时，您可以使用 GNU调试器(GDB)，它是可移植的，可在 Linux 等类 Unix 系统以及窗口上运行。 GDB 适用于多种编程语言，包括 Ada、 Java 、C、C++ 和 Objective-C。使用 GDB，您可以在本地或远程调试您的应用程序。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 调试>平台"页面或
- 点击窗口工具栏中的  ，选择要新建脚本的包，选择 调试>平台"页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
上下文菜单	浏览器窗口 |右键单击包|执行分析器
键盘快捷键	Shift+F12

设置 GNU调试器

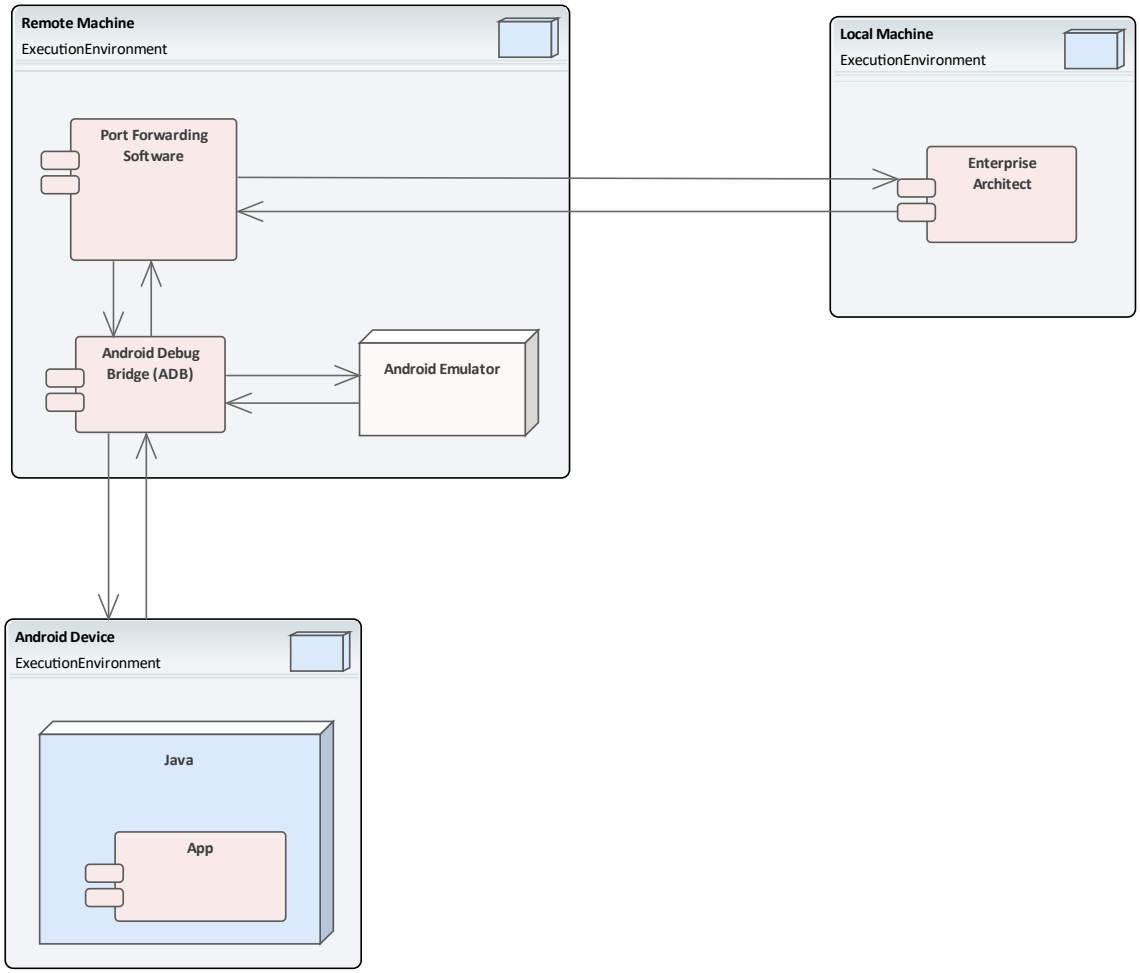
任务	细节
设置脚本	分析器脚本Enterprise Architect中调试的基本要求；您使用执行分析器工具栏创建一个脚本。 在执行分析器脚本器的 平台"页面，在 调试器"字段中点击下拉箭头并选择 "GDB"。
定义连接设置	属性面板显示您为其提供值的许多连接设置。 • path - <path> - GDB 可执行文件的完成文件路径；只有在系统路径中找不到 GDB 时才指定此项 • 源- <path>, <path> - 调试器搜索源文件的路径，如果它们不驻留在可执行目录中 • remote - F - 设置为远程调试；否则留空 • port - <端口> - 远程服务器上要连接的端口 • host - localhost - 要连接的主机名 • fetch - T - 设置从远程系统检索二进制文件 • dumpgdb - <path> - 将 GDB 输出写入到的文件名 • initpath - <path> -完成文件的完成文件路径

注记

- GDB 的A要求是你的源代码文件路径不能包含空格；调试器将无法运行运行文件路径中的空格

Android调试器

如果您正在开发在 Android 设备或模拟器上运行的Java应用程序，您还可以调试它们。本地和远程机器可以在 32 位平台或 64 位平台上。



系统需求

在远程机器上，需要此软件：

- Android SDK，包括android调试桥，ADB（你需要熟悉SDK及其工具）
- Java JDK（32 位和 64 位支持）
- 端口转发软件（第3方）

在本地计算机上，需要此软件：

- Enterprise Architect 10 或更高版本

分析器脚本

字段/按钮	行动
调试器	单击下拉箭头并选择Java (JDWP)。

运行	单击此单选按钮。
默认目录	不适用 - 留空。
申请途径	不适用 - 留空。
命令行参数	不适用 - 留空。
先编译	不适用 - 留空。
显示控制台	不适用 - 留空。
显示诊断消息	不适用 - 留空。
联系	不适用 - 留空。
端口	这是应用程序端口，使用 adb 或其他方式前向分配， Enterprise Architect和 Android 虚拟机 (VM) 可以通过该端口进行通信。
主持人	主机（默认为localhost ） 如果 Android 在连接到联网计算机的设备上的模拟器上运行，请在此处输入网络名称。 默认情况下，调试将尝试连接到您在本地机器上指定的端口。
源	这是Java中类路径设置的源等价物。 应该列出每个源树的根。如果指定了多个，则应以分号分隔；那是： c:\myapp\src;c:\myserver\src 您必须至少指定一个根路径。 当断点发生时，调试器会在此处列出的每个源树中搜索 java源。
日志记录	允许记录来自调试器的附加信息 可能的值：真、 false 、 1 、是、否
输出	指定要写入的本地log文件的全名。 该文件夹必须存在，否则不会创建log 。 log文件通常包含调试器和 VM 之间发送的字节转储。
平台	如果要调试在任何 android 场景下运行的Java ，请选择 Android。 对于所有其他方案，请选择Java 。

配置端口用于调试-端口转发（本地）

调试器一次只能调试一个虚拟机；它使用单个端口与 VM 通信。可以使用 Android SDK 提供的 ADB 分配要调试的应用程序的端口。

在调试之前，在设备中启动一次应用程序。当应用程序启动时，发现它的进程标识符（pid）：

亚行 jdwp

列出的最后一个数字是上次启动的应用程序的 pid；注记pid 并使用它来允许调试器连接到 VM：

- adb forward tcp:port jdwp:pid
  - 端口 = 分析器脚本中列出的端口号
  - pid = 设备上应用程序的进程 ID

配置端口用于调试-端口转发（远程）

要进行远程调试，应该遵循与本地计算机相同的过程，但通信需要额外的转发，因为使用 adb forward 命令创建的套接字将仅在本地适配器上侦听。套接字绑定到localhost ，并尝试连接到此端口。将遇到 连接被拒绝”消息

为了实现远程调试，需要在远程机器上运行一个代理，该代理侦听所有传入连接并将所有流量转发到 adb；端口有许多软件产品可以做到这一点。

除非您使用Enterprise Architect配置了代理端口转发器，否则远程调试将不起作用。

Java JDWP调试器

Java提供了两种主要的调试技术：一种基于进程内代理的系统，称为Java虚拟机工具接口(JVMTI)，另一种是基于套接字的范例，称为Java调试连线协议 (JDWP)。 Java A机可以命名其中之一，但不能同时命名两者，并且必须在启动 JVM 时配置特征。

系统需求

1. Enterprise Architect JDWP 调试器将只能与使用 "DWP"选项启动的 JVM 通信。以下是命令行选项的示例：
   java -agentlib:jdwp=transport=dt_socket,address=localhost:9000,server=y,suspend=n -cp "c:\java\myapp;%classpath%" demo.myApp "param1" "param2"
2. 虚拟机当前不应附加到调试器。
3. Enterprise Architect和 Eclipse 不能同时调试 VM。

分析器脚本

字段/按钮	行动
调试器	单击下拉箭头并选择Java (JDWP)。
运行	单击此单选按钮可在执行脚本时运行调试器。
默认目录	不适用 - 留空。
申请途径	不适用 - 留空。
命令行参数	不适用 - 留空。
先编译	不适用 - 留空。
显示控制台	不适用 - 留空。
显示诊断消息	不适用 - 留空。
联系	不适用 - 留空。
端口	在Java命令行选项中设置启动期间分配给 VM 进程的应用程序端口转发。
主持人	设置主机（默认为localhost ） 如果 VM 在联网计算机上运行，请在此处输入网络名称或 url。 默认情况下，调试将尝试连接到您在本地机器上指定的端口。
源	这是Java中类路径设置的源等价物。 列出每个源树的根；指定至少一个根路径。如果您指定多个，请用分号分隔它们；例如： c:\myapp\src **;** c:\myserver\src 当断点发生时，调试器会在此处列出的每个源树中搜索Java源。

日志记录	启用或禁用来自调试器的附加信息的日志记录。 可能的值包括： • 真的 • false • 1 • 0 • 是的 • 不
输出	指定要写入的本地log文件的全名。如果文件夹不存在，则不会创建log 。 log文件通常包含调试器和 VM 之间发送的字节转储。
平台	选择Java 。

用于调试的配置端口

调试器一次只能调试一个虚拟机；它使用单个端口与 VM 通信。待调试应用程序的端口是在创建 VM 时分配的。

本地调试

如果Enterprise Architect和Java VM 在同一台机器上运行，您可以执行本地调试。必须在启用 JDWP 传输的情况下启动 VM - 有关命令行选项规范，请参阅 Oracle 的*Java*平台调试器架构*(JPDA)*文档。例如：

java -agentlib:jdwp=transport=dt_socket,address=localhost:9000,server=y,suspend=n -cp "c:\samples\java\myapp;%classpath%" samples.MyApp "param1" "param2"

在这个例子中，分析器脚本的值是'host: localhost '和'port:9000'。

远程调试

当Enterprise Architect在本地机器上运行而Java VM 在远程机器上运行时，您可以执行远程调试。有必要在启用 JDWP 传输的情况下启动 VM - 有关命令行选项规范，请参阅 Oracle 的 JPDA 文档。这是一个示例，其中远程计算机的网络名称为 testmachine1：

java -agentlib:jdwp=transport=dt_socket,address=9000,server=y,suspend=n -cp "c:\samples\java\myapp;%classpath%" samples.MyApp "param1" "param2"

注记地址中没有主机名。这意味着 VM 将侦听来自任何机器的连接。在此示例中，分析器脚本的值将是 "主机：分析器"和 "端口：9000"。

追踪点输出

分析器脚本的分析器页面使您能够指导任何跟踪语句的输出在调试会话期间的去向脚本

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 调试> Tracepoints"页面或
- 点击窗口工具栏中的  ，选择要新建脚本的包，选择 调试> Tracepoints"页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器> 选择并运行脚本
上下文菜单	浏览器窗口 |右键单击包|执行分析器
键盘快捷键	Shift+F12

跟踪点属性

字段	细节
输出	您可以从两个选项中进行选择： • '屏幕' (默认) - 输出被定向到调试窗口 • '文件' - 输出被定向到文件
文件夹	输入用于跟踪语句log文件的文件夹。
文件名	输入用于跟踪语句log文件的名称。
覆盖	如果选中，则每次启动调试会话时都会覆盖指定的文件。
自动编号	如果选中，则跟踪log文件由您指定的文件名和一个数字组成。 每次启动调试会话时，数字都会增加。
前缀函数的跟踪输出	如果选中，则在调试会话运行期间执行的任何跟踪状态都以当前函数调用为前缀。

工作台设置

本主题描述在Java和 Microsoft .NET上设置物件工作台的要求。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 调试> Workbench"页面或
- 点击窗口工具栏中的  ，选择要新建脚本的包，选择 调试> Workbench"页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Shift+F12

平台

平台	细节
支持的平台	Workbench 支持以下平台： • Microsoft .NET （2.0 版或更高版本） • Java （JDK 1 .4 或更高版本）
微软.NET工作台	.NET工作台需要一个程序集，用于创建工作台项。 您在分析器的 工作台"页面上指定程序集的脚本。 使用.NET工作台有两个限制： • 不支持在托管代码中定义为结构的成员 • 不支持定义为内部的类
Java工作台	Java工作台使用分析器脚本调试'页面中配置的虚拟机设置来创建JVM。

Microsoft C++ 和本机（C、VB）

只有当可执行文件有相应的 PDB 文件时，您才能调试本机代码。作为构建应用程序的结果，会创建A PDB 文件。

构建应包括完整的调试信息，并且不应设置优化。

脚本必须指定两件事来支持调试：

- 可执行文件的路径
- Microsoft Native 作为调试平台

常规设置

这是调试 Microsoft 本机应用程序（C++、C、Visual Basic）的一般设置。调试时有两种选择：

- 调试一个应用程序
- 附加到正在运行的应用程序

选项1 -调试应用程序

字段	行动
调试器	选择 Microsoft Native 作为调试平台。
x64	如果您正在调试 64 位应用程序，请选中此复选框。 如果您正在调试 32 位应用程序，请取消选中该复选框。
模式	选择运行单选按钮。
默认目录	这被设置为正在调试的进程的默认目录。
申请途径	选择并输入应用程序可执行文件的完整或相对路径。 • 如果路径包含空格，则指定完整路径；不要使用相对路径 • 如果路径包含空格，则路径必须用引号引起来
命令行参数	在启动时传递给应用程序的参数。
显示控制台	为调试器创建一个控制台窗口；不适用于附加到进程。
符号搜索路径	指定任何其他路径来定位调试器的调试符号；用分号分隔路径。

选项 2 - 附加到正在运行的应用程序

字段	行动
调试器	选择 Microsoft Native 作为调试平台。
x64	如果您正在调试 64 位应用程序，请选中此复选框。 如果您正在调试 32 位应用程序，请取消选中该复选框。
模式	选择附加到进程单选按钮。
符号搜索路径	指定任何其他路径来定位调试器的调试符号。 如果您愿意，可以在此处指定符号服务器；用分号或逗号分隔路径。

调试符号

对于使用 Microsoft Platform SDK 构建的应用程序，在构建应用程序时将调试符号写入应用程序 PDB 文件。

可视化执行调试器使用的 API窗口调试工具使用这些符号向执行分析器控件提供有意义的信息。

这些符号很容易过时并导致异常行为 - 调试器可能会在断点处突出显示编辑器中的错误代码行；因此，最好确保在任何调试或记录会话之前构建应用程序。

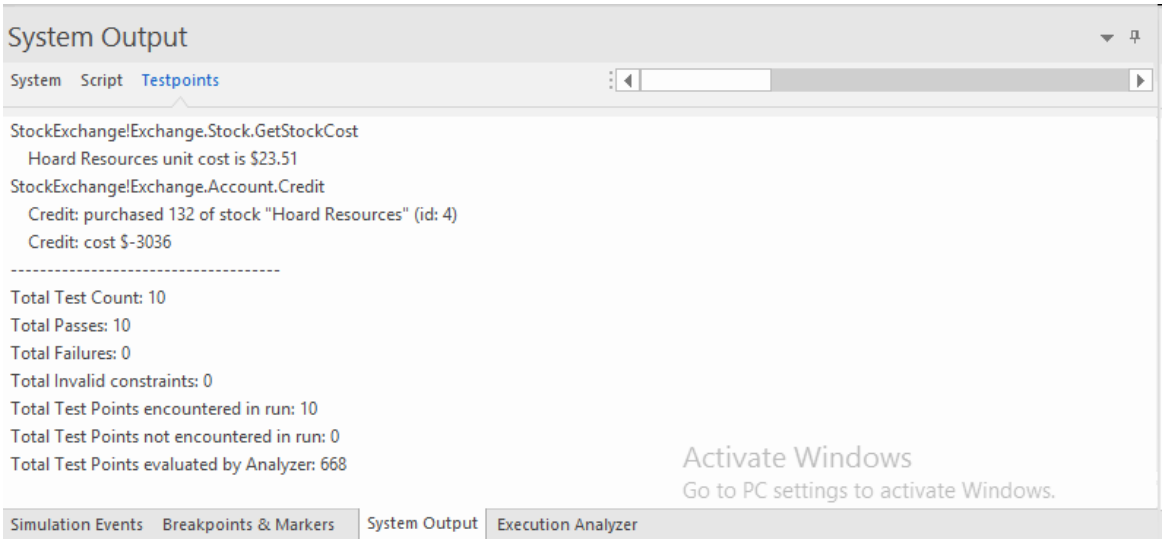
调试器必须通知 API 如何协调正在调试的映像中的地址；它通过指定一些 API 路径来告诉它在哪里寻找 PDB 文件来做到这一点。

对于没有找到调试符号的系统 DLL（kernel32、调用堆栈），二进制显示一些仅带有模块名称和地址的帧。

您可以通过向 API 传递额外的路径来补充翻译的符号；您在 调试"选项卡中以分号分隔的列表传递其他符号路径。

测试点输出

分析器脚本的分析器脚本帮助您配置测试点运行的输出。
默认情况下，输出记录到系统输出窗口，如本例所示。



访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 测试>测试点"页面或
- 点击窗口工具栏中的 ，选择要新建脚本的包，选择 测试> Testpoints"页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器>视图分析器脚本
键盘快捷键	Shift+F12

选项

选项	描述
输出	您可以从两个选项中进行选择： • '屏幕'（默认）- 输出被定向到系统输出窗口的'测试点'选项卡 • '文件' - 输出被定向到文件
文件夹	单击用于测试点log文件的文件夹。
文件名	输入用于测试点log文件的名称。
覆盖	选择此选项时，每次执行测试点运行时都会覆盖指定的文件。

自动编号	选择该选项时，测试点输出由您指定的文件名和测试运行次数组成；每次执行测试运行时，数字都会增加。
前缀函数的跟踪输出	选择此选项时，在测试点运行期间执行的任何跟踪语句都以当前函数调用为前缀。

测试脚本

这些部分解释了如何配置分析器的 测试”页面来对你的脚本进行单元测试。大多数用户会将其应用于 NUnit 和 JUnit 测试场景。 Enterprise Architect接受来自这些系统的输出，并且可以自动添加和管理每个单元测试用例历史。要查看案例历史记录，您可以选择测试案例类元素并按 Alt+2 >测试。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 测试>测试”页面或
- 点击窗口工具栏中的  ，选择要新建脚本的包，选择 测试>测试”页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
分析器上下文菜单	测试
键盘快捷键	Shift+F12

行动

测试 类型该字段中的测试命令或脚本。例如：

- NUnit - "C:\Program Files\NUnit\bin\nunit-console.exe" "bin\debug\Calculator.exe"
- JUnit - java junit.textui.Testrunner %N

此字段中列出的命令就像来自命令提示一样执行；因此，如果可执行路径或任何参数包含空格，它们必须用引号引起来。

如果您在测试脚本中包含string %N，则在执行脚本时，它将被当前所选类的完全命名空间限定名称替换。

执行命令为 单击下拉箭头并选择适当的选项：

- 批处理文件- 使用此选项创建一个在系统命令窗口中执行的 shell 脚本；可以通过此脚本中的命令访问环境变量
- 进程-使用此选项运行单个程序 - 命令应指定程序的路径，以及任何命令行参数；如果路径或参数包含空格，请用引号将路径括起来 - 例如：“c:\program files (x86)\java\bin\javac.exe”

默认目录 默认为为编译脚本输入的值。如果尚未为编译脚本设置值，请浏览或键入清理脚本进程的默认目录运行。

解析输出 选择解析器后，可以从模型中解析、保存和管理 nUnit 和 jUnit 测试的输出；（ Alt+2 >测试 ）。请注意，仅在选择解析器时才会捕获输出。

远程主机 远程主机系统的 ID 及其端口中的类型；例如，mypc01:7777。

如果将此属性设置为#属性#，则脚本在窗口运行时发送到窗口卫星服务，在Wine下运行时发送到Linux卫星服务。服务 ID 和端口在分析器脚本编辑器的

私人选项 - 服务”部分中定义。

编译第一 选择以确保每次运行测试时都会编译包。

清理脚本

增量构建是只构建那些以某种方式发生变化的资产的做法。然而，有时有理由从头开始重新构建一切。此命令用于这些情况，以删除与特定构建或配置关联的二进制文件和中间文件。然后可以重建该项目。当您在脚本上执行 重建"菜单选项时，将执行您在此字段中指定的命令，然后立即执行来自同一分析器脚本的 编译"命令。一些编译器可以为您执行此操作。例如，Visual Studio 有 /t:clean"命令行开关。

这是一个示例脚本：清理/ 收调试MyProject.sln

访问

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 编译>清理"页面或
- 点击窗口工具栏中的 ，选择要新建脚本的包，选择 编译>清理"页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行>源>编译>清理
分析器上下文菜单	清理
键盘快捷键	Shift+F12

方面

方面	细节
清理	输入从脚本上下文菜单中选择 清理"时要执行的命令。
执行命令为	单击下拉箭头并选择适当的选项： • 批处理文件- 使用此选项创建一个在系统命令窗口中执行的 shell 脚本；可以通过此脚本中的命令访问环境变量 • 进程-使用此选项运行单个程序 - 命令应指定程序的路径，以及任何命令行参数； - 如果路径或参数包含空格，请用引号将它们括起来 - 例如：“c:\program files (x86)\java\bin\javac.exe”
默认目录	默认为为编译脚本输入的值。如果尚未为编译脚本设置值，请浏览或键入清理脚本进程的默认目录运行。
解析输出	单击下拉箭头并选择自动解析编译器输出的方法。 如果您选择此选项，脚本的输出将记录在系统输出窗口中； Enterprise Architect根据您指定的语法解析输出。
远程主机	远程主机系统的 ID 及其端口中的类型；例如，mypc01:7777。 如果将此属性设置为#属性#，则脚本在窗口运行时发送到窗口卫星服务，在Wine下运行时发送到Linux卫星服务。服务 ID 和端口在分析器脚本编辑器的 私人选项 - 服务"部分中定义。

编译脚本

编译”页面使您可以输入命令来构建您的项目。您可以在编写命令行时使用Enterprise Architect本地路径和环境变量。您可以选择创建自己的构建脚本，输入各种 shell 命令。您还可以选择简单地运行外部程序或批处理文件，例如 Ant 脚本。

访问

在执行分析器窗口中：

- 找到并双击所需的脚本并选择 编译>编译”页面或
- 点击窗口工具栏中的 ，选择要新建脚本的包，选择 编译>编译”页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
分析器上下文菜单	编译
键盘快捷键	Ctrl+Shift+F12

编译

使用 windows shell 命令在此文本框中编写脚本；本节的格式和内容取决于您用于构建项目的实际编译器。如果路径或参数包含空格，请用引号将它们括起来；例如：“c:\program files (x86)\java\bin\javac.exe”。

这里有些例子：

视觉工作室：

"C:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\IDE\devenv.com" /重建调试RentalSystem.sln

使用本地路径的 Visual Studio：

"%VsCompPath%\devenv.exe" /build调试Subway.sln

Java ：

C:\Program Files (x86)\ Java \jdk1.6.0_22\bin\javac.exe" -g -cp "%classpath%;." %r*.java

Java使用本地路径：

"%JAVA%\bin\javac.exe" -g -cp "%classpath%;." %r*.java

通配符Java构建 (%r) - 子文件夹中的源文件可以使用 %r 令牌构建。如示例所示，该令牌具有对所有子文件夹中的任何文件递归执行相同命令的效果。

执行命令为

单击下拉箭头并选择执行脚本的模式：

- 批处理文件-使用此选项在系统命令窗口中执行 shell 脚本；可以通过此脚本中的命令访问环境变量
- 进程-使用此选项将命令作为单个程序执行；该命令应指定程序的路径，以及任何命令行参数

默认目录

浏览或输入构建脚本进程将在其中运行的默认目录运行。

解析输出

单击下拉箭头并选择自动解析编译器输出的方法。

如果您选择此选项，脚本的输出将记录在系统输出窗口中； Enterprise Architect根据您指定的语法解析输出。

远程主机

远程主机系统的 ID 及其端口中的类型；例如，mypc01:7777。

如果将此属性设置为#属性#，则脚本在窗口运行时发送到窗口卫星服务，在Wine下运行时发送到Linux卫星服务。服务 ID 和端口在分析器脚本编辑器的 私人选项 - 服务"部分中定义。

编译后部署

选择此框可在此编译脚本完成后立即执行 Deploy 脚本。

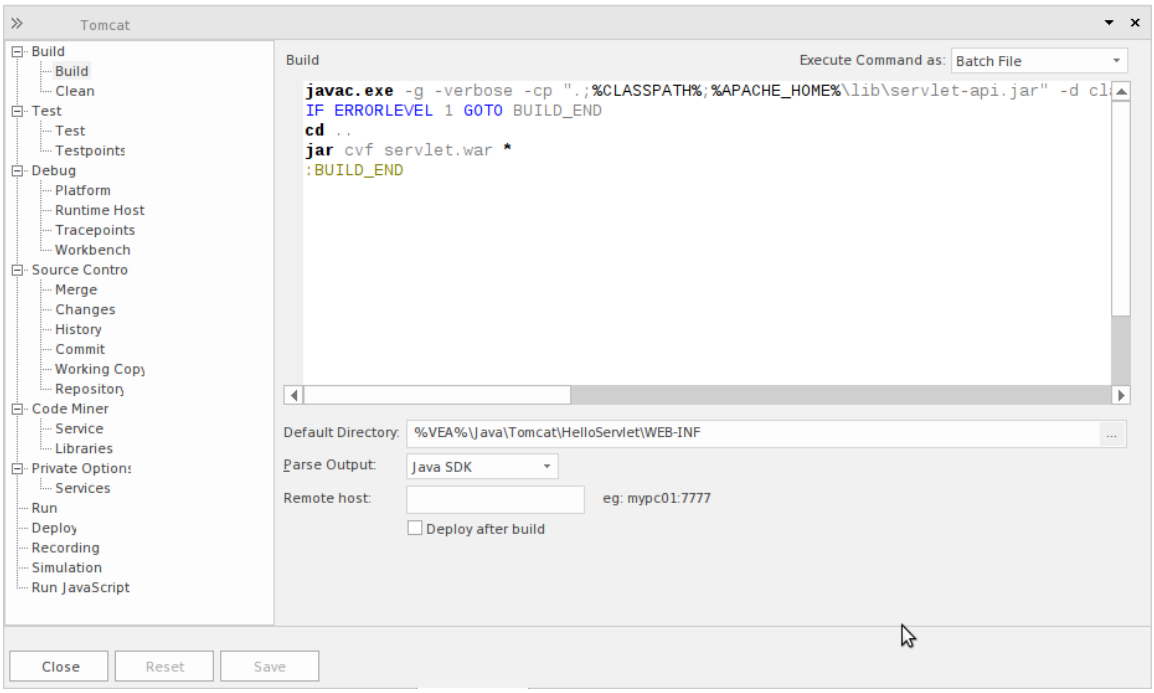
注记

要执行编译脚本，请单击浏览器窗口中的包，然后：

- 右键单击任何工具栏并选择 分析器工具栏|编译"，或
- 按 Ctrl+Shift+F12 或
- 选择 执行>源>编译>编译"功能区选项

分析器脚本

分析器脚本器有一个简单的用户界面，左侧是脚本的树形视图，可以轻松导航脚本组，右侧是内容视图，您可以在其中定义和配置脚本。



访问

在 执行分析器"窗口中，可以：

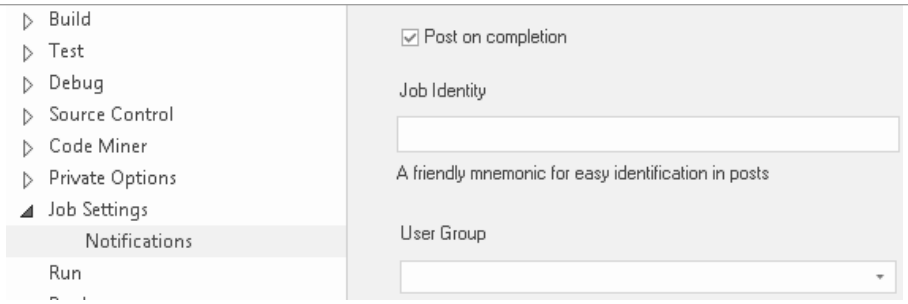
- 双击脚本进行编辑或
- 右键单击脚本并选择 编辑"选项

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Shift+F12

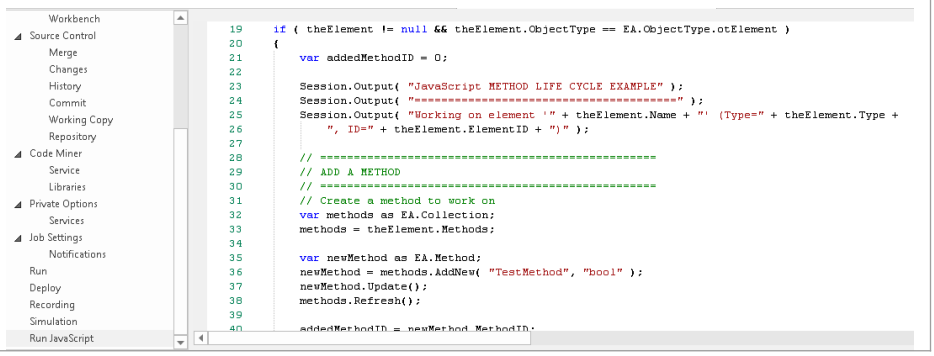
执行分析器脚本

任务- Page	行动
编译-编译	输入脚本或命令来构建应用程序。这可以是 Apache Ant 或 Visual Studio 命令，但也可以根据您的开发环境进行定制。注记：请记住选择解析器以在出现任何错误时直接获取源代码。解析器字段位于同一页面上，并提供对多种语言的支持。
编译-清理	输入脚本或命令来清理以前的构建。这是您构建系统时通常发出的命令行。这可以是 Apache Ant 或 Visual Studio 命令，具体取决于您的开发环境。

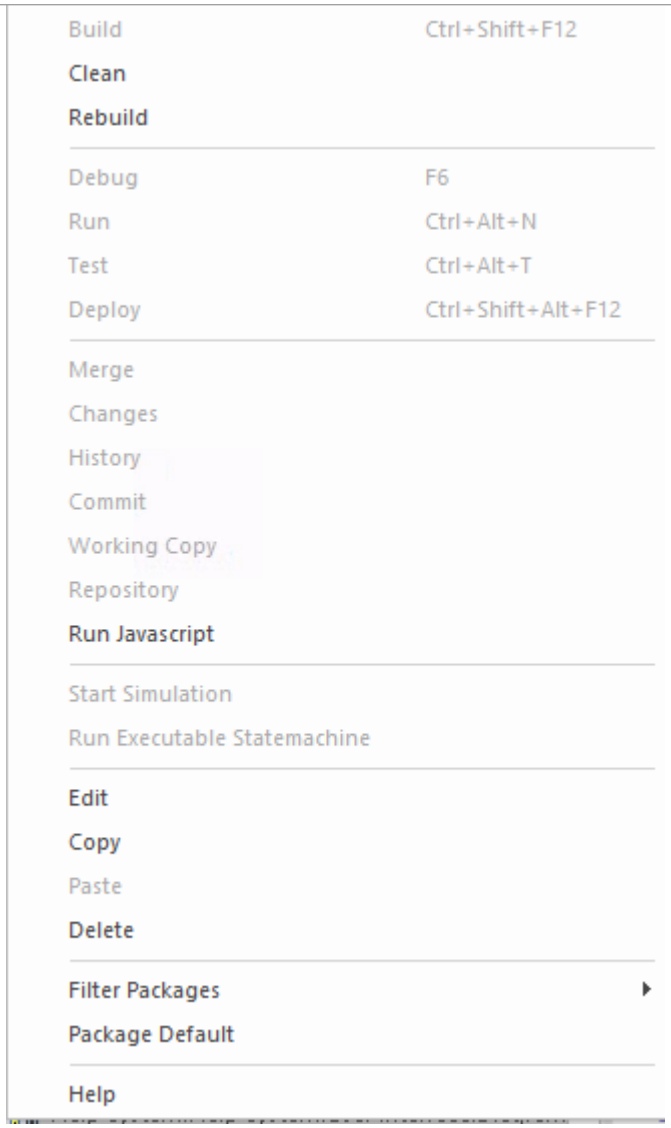
测试-测试	输入脚本或命令来测试应用程序。这通常是可以配置 nUnit 或 jUnit 调用的地方，但它也可以很容易地成为任何过程或程序。
测试- 测试点	指定测试点运行的输出被发送到哪里。
调试-平台	指定调试平台、要调试的应用程序和调试模式（附加到进程或运行）。
调试- 跟踪点	指定在调试会话期间遇到的跟踪点的输出发送到哪里。
调试-工作台	对于.NET项目，要加载的程序集。 Java不需要。
调试-运行时主机	允许Enterprise Architect使用命令行启动要调试的程序。这通常用于使用套接字传输进行调试的 Mono 或Java程序。该命令在调试器运行之前运行。此命令中的端口应与传递给调试中指定的 端口"选项的值相同。当调试器启动时，它将尝试连接到此端口上的运行时。如果成功，它将绑定任何断点并恢复程序，它假定程序已挂起。 Java和 Mono 在调试传输上都有命令行选项，以在调试器连接之前暂停进程。
源控制- Merge	这是从上下文脚本时间菜单中选择 合并"选项时执行的分析器。它提供了一个运行程序或 shell 脚本来检查源文件之间差异的地方。
源控制-修改	这是从上下文脚本时间菜单中选择 修改"选项时执行的分析器。它为运行源控制程序（例如 "svn"）提供了一个地方，该程序可能会列出对源控制存储库的当前更改。
源控制-历史	这是从上下文脚本时间菜单中选择 历史"选项时执行的分析器。它为运行源控制程序（例如 "svn"）提供了一个地方，该程序可能会列出对源控制存储库的更改历史记录。
源控制- Commit	这是从上下文脚本时间菜单中选择 提交"选项时执行的分析器。它为运行源控制程序（例如 'svn'）提供了一个地方，该程序可能会提交对源控制工作副本的更改。
源控制-工作副本	这是从上下文脚本时间菜单中选择 工作副本"选项时执行的分析器。它为运行源控制程序（例如 运行"）提供了一个地方，以对源存储库的当前工作副本执行操作。
源控制-存储库	这是从上下文脚本时间菜单中选择 存储库"选项时执行的分析器。它提供了一个运行源控制程序（例如 "svn"）对源存储库执行操作的地方。
代码矿工- 服务	在本部分中，您可以选择代码矿工服务的运行方式。您可以选择远程服务器或在本地使用库。
代码矿工- 库	本节为代码矿工库的管理提供了场所。在这里，您可以基于项目代码库或存储库创建库。在这里创建的代码矿工库可以使用 mFQL 查询进行搜索。在 mFQL 中组成的查询可用于在单个操作中搜索一个或多个库。
私人期权 - 服务	这是为 Linux 和窗口配置Enterprise Architect卫星服务的 IP 地址和端口的地方。这些服务为系统管理功能和远程调试方案提供企业范围的支持。
作业设置	分析器中包含的大部分命令都是作为 Job脚本中的作业执行的。每个脚本都可以配置为在指定作业完成时向指定用户组发布通知。 作业设置"组提供 通知"选项，该选项显示您输入作业身份的字段，作为文本的一部分显示给模型邮件用户组的成员，以及用户组名称。

<table>
<tr>
<td></td>
<td>


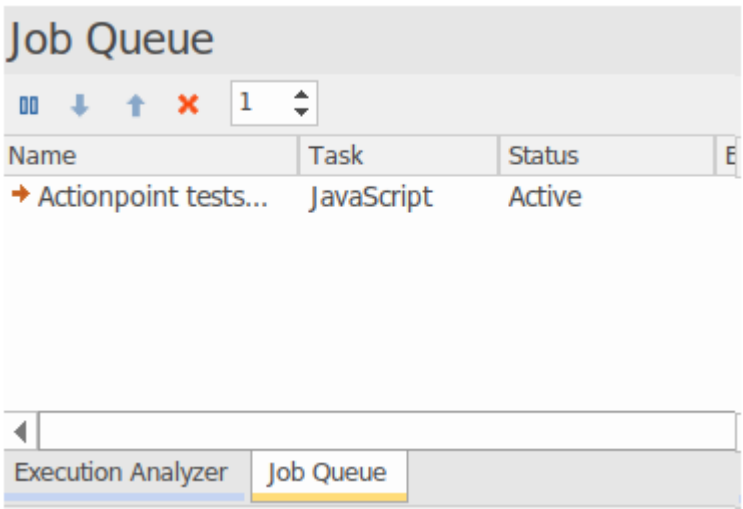
完成后发布 - 选中此复选框以启用其他两个字段
Job Identity - 要显示的文本中的类型，它也应该标识工作；例如： 安装程序已完成”
用户组 - 单击下拉箭头并选择适当的模型邮件用户组

</td>
</tr>
<tr>
<td>运行JavaScript</td>
<td>在本节中，您可以通过从上下文的分析器菜单中选择 迈行JavaScript ”选项来创建和存储可以执行的JavaScript脚本


</td>
</tr>
</table>



选择此选项后，将在 作业队列"窗口中创建一个作业。



运行	输入命令以运行应用程序。
部署	输入脚本或命令来部署项目。编译您的 jar 文件。部署到您的设备、模拟器或

	Tomcat 服务器。发布一个网站。由你决定。
记录	你的序列图像国家电网吗？使用过滤器减少混乱。过滤器在您的代码库中定义了排除区域，可以显着减少正在记录的任何 噪音”。即使是准确的噪音也不总是有用的。
仿真	完全控件仿真配置。

管理分析器脚本

执行分析器窗口使您能够管理模型中的所有分析器脚本。您可以使用窗口工具栏按钮或脚本上下文菜单选项来控制脚本任务。脚本按包列出；该列表仅显示具有针对它们定义的分析器脚本的包。每个用户可以设置自己的活动脚本，独立于同一模型的其他用户；一个用户激活脚本不会影响其他用户当前活动的脚本或影响他们可用的脚本。活动脚本控制执行分析器的行为；例如，从菜单中选择构建命令或单击工具栏上的调试按钮时。

窗口中可能有很多包含脚本的包。为了帮助您定位和隔离特定的包，请使用 "过滤器包"上下文菜单选项，如本主题的上下文菜单选项表中所述。

注记当脚本在执行分析器窗口中高亮显示时，窗口上下文菜单选项和执行分析器窗口工具栏按钮将对高亮的脚本进行操作。但是，任何功能区或浮动工具栏或调试器（不在窗口上）中的执行分析器选项将始终使用默认分析器脚本 - 在脚本名称旁边具有选中复选框的脚本。

访问

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
键盘快捷键	Shift+F12

工具栏选项

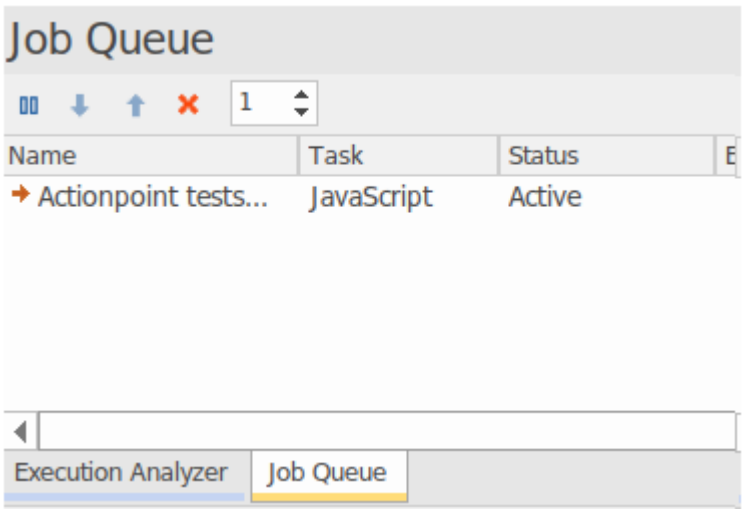
工具栏按钮	行动
	快速访问分析器核心窗口，如调用堆栈或局部变量，以及功率特征： • 剖析 • 记录 • 测试点 • 仿真
	在 Linux 或分析器下为所选包创建和编辑新的分析器脚本窗口
	导出脚本。 导出一个或多个分析器脚本导出为 XML 文件，可用于将脚本导入另一个模型。 将显示 "执行分析器：导出"对话框，您可以从中选择要导出的一个或多个脚本，然后显示目标文件名和位置的提示。
	导入脚本。 导入一个或多个分析器脚本从先前导出的 XML 文件导入当前模型。 将显示 "查找包"对话框，您可以在其中选择要导入脚本的包，然后显示源文件名和位置的提示。
	执行活动脚本的 "编译"命令。

	取消当前正在进行的 编译”命令。
	执行活动脚本的 运行”命令。
	执行活动脚本的 测试”命令。
	执行活动脚本的 部署”命令。

上下文菜单选项

右键单击所需的脚本或包以显示上下文菜单。

选项	行动
加新脚本	将新脚本添加到选定的包。 执行分析器窗口显示，显示 编译”页面。
粘贴脚本	粘贴将Enterprise Architect剪贴板中的脚本复制到选定的包中。 您可以多次粘贴复制的脚本；每个副本都有后缀 副本”。 要重命名复制的脚本，请按 F2 并改写脚本名称。
导出脚本	从所选包中导出脚本。 将显示 执行分析器：导出”对话框，从中选择要导出的一个或多个脚本，然后显示目标文件名和位置的提示。
导入脚本	将导入文件中的脚本导入到选定的包中。 A显示源文件名和位置的提示。
过滤器包	显示选项子菜单以： • 输入要过滤包列表的包路径 - 当您选择 过滤器包”选项时，将显示一个提示以接受当前选择的包路径，或者您可以删除路径的结尾元素以指定更大的套包；当您单击确定按钮时，列表中的第一个包被展开以列出它包含的脚本 • 在显示包的完整列表和隐藏包的完整列表之间切换以仅显示当前选择的包 • 删除当前活动的过滤器以显示包的完整列表
在项目中选择浏览器	在浏览器窗口中突出显示选定的包。 显示浏览器窗口，该窗口现在展开以显示突出显示的包。
编译	执行所选脚本的 编译”命令。
清理	执行选中脚本的 清理”命令
重建	执行所选脚本的 清理”和 编译”命令。
调试	执行所选脚本的 调试”命令。

运行	执行所选脚本的 运行”命令。
测试	执行所选脚本的 测试”命令。
部署	执行所选脚本的 部署”命令。
合并	执行所选脚本的'Merge'源控制命令。
修改	执行选中脚本的 修改”源控制命令。
历史	执行所选脚本的'History'源控制命令。
犯罪	执行所选脚本的'Commit'源控制命令。
工作副本	执行所选脚本的'Working Copy'源控制命令。
存储库	执行所选脚本的 存储库”源控制命令。
运行JavaScript	执行所选脚本的 运行JavaScript ”命令。选择此选项后，将在 作业队列”窗口中创建一个作业。 
开始仿真	开始 分析器脚本”页面仿真引用的模拟。
运行可执行状态机	选定的可执行状态机开始工件。
编辑	在 分析器脚本编辑器”中打开选定的脚本。
复制	将选定的脚本复制到Enterprise Architect剪贴板。
粘贴	粘贴最近复制的脚本到与所选脚本相同的包中。 您可以多次粘贴复制的脚本；每个副本都有后缀 副本”。 要重命名复制的脚本，请按 F2 并改写脚本名称。
删除	删除选定的脚本；没有提示确认。 要从执行分析器窗口中删除一个包，请从该包中删除脚本。删除最后一个脚本时，不再列出该包。

包默认	将所选脚本设置为包的默认脚本。 脚本左侧的图标改变颜色；任何以前的包默认恢复正常。