Server Automation

软件版本：10.50

开发人员指南

文档发布日期：2016 年 7 月
软件发布日期：2016 年 7 月

Hewlett Packard Enterprise

法律声明

担保

HPE 产品和服务的唯一担保已在此类产品和服务随附的明示担保声明中提出。此处的任何内容均不构成额外担保。HPE 不会为此处出现的技术或编辑错误或遗漏承担任何责任。

此处所含信息如有更改，恕不另行通知。

受限权利声明

机密计算机软件。必须拥有 HPE 授予的有效许可证，方可拥有、使用或复制本软件。按照 FAR 12.211 和 12.212，并根据供应商的标准商业许可的规定，商业计算机软件、计算机软件文档与商品技术数据授权给美国政府使用。

版权声明

商标声明

Adobe® 是 Adobe Systems Incorporated 的商标。

Microsoft® 和 Windows® 是 Microsoft Corporation 在美国的注册商标。

UNIX® 是 The Open Group 的注册商标。

文档更新

本文档的标题页包含以下标识信息：

- 软件版本号，指示软件版本。
- 文档发布日期，该日期将在每次更新文档时更改。
- 软件发布日期，用于指示该版本软件的发布日期。

要检查是否有最新的更新，或者验证是否正在使用最新版本的文档，请访问：https://softwaresupport.hpe.com/。

需要注册 HPE Passport 才能登录此站点。要注册 HPE Passport ID，请单机 HPE 软件支持站点上的 **Register** 或单击“HP Passport”登录页面上的**Create an Account**。

此外，如果订阅了相应的产品支持服务，则还会收到更新的版本或新版本。有关详细信息，请与您的 HPE 销售代表联系。

支持

访问 HPE 软件支持网站，地址为:https://softwaresupport.hpe.com。

此网站提供了联系信息，以及有关 HPE 软件提供的产品、服务和支持的详细信息。

HPE 软件联机支持提供客户自助解决功能。通过该联机支持，可快速高效地访问用于管理业务的各种交互式技术支持工具。作为尊贵的支持客户，您可以通过该支持网站获得下列支持：

- 搜索感兴趣的知识文档
- 提交并跟踪支持案例和改进请求
- 下载软件修补程序
- 管理支持合同
- 查找 HPE 支持联系人
- 查看有关可用服务的信息
- 参与其他软件客户的讨论
- 研究和注册软件培训

大多数提供支持的区域都要求您注册为 HPE Passport 用户再登录，很多区域还要求用户提供支持合同。要注册 HPE Passport ID, 请单击 HPE 支持站点上的 **Register**，或单击“HP Passport”登录页面上的 **Create an Account**。

要查找有关访问级别的详细信息，请访问：https://softwaresupport.hpe.com/web/softwaresupport/access-levels。

HPE Software Solutions Now 可访问 HPSW 解决方案和集成门户网站。此网站将帮助您寻找可满足您业务需求的 HPE 产品解决方案，包括 HPE 产品之间的集成的完整列表以及 ITIL 流程的列表。此网站的 URL 为 https://softwaresupport.hpe.com/。

内容

开发人员指南概述

Server Automation 平台中包含一组 API 和一个运行时环境，用于帮助集成和扩展 SA。Server Automation 平台 API 提供了各种核心服务，例如审核符合性、Windows 修补程序管理和操作系统配置。运行时环境将执行可用于访问全局文件系统 (OGFS) 的全局 Shell 脚本。

本主题提供以下信息:

概述

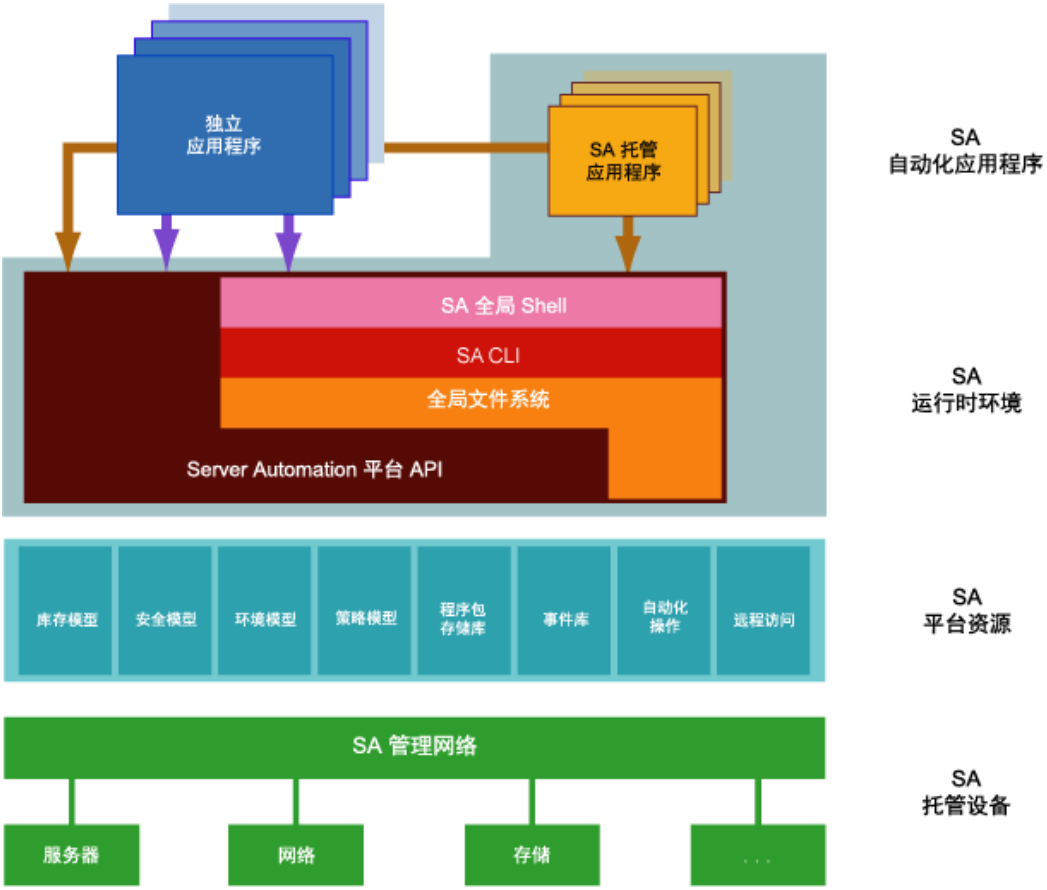
使用 Server Automation 平台，您可以执行以下任务：

- 构建新自动化应用程序和扩展 SA 以提高 IT 生产力，并契合您的 IT 策略。
- 与其他 IT 系统交换信息，例如与现有的监控、问题工单处理、计费和虚拟化技术系统交换信息。
- 使用 SA 模型库存储和组织关于运营、环境和资产的关键 IT 信息。
- 自动管理各种应用程序和操作系统。
- 将现有 Unix 和 Windows 脚本包含到 SA 中，使脚本能够在经审核的安全环境中运行。

组件

下图显示 Server Automation 平台的主要元素。

Server Automation 平台组件



如上图所示，该平台包括以下五个关键元素。将在随后的章节中更详细地讨论每一个元素。

- **SA 自动化应用程序：**应用程序用户在平台的基础上进行编写。这些应用程序可以是运行于正在运行的 SA 环境中的 SA 托管应用程序，也可以是运行于现有业务和管理系统环境中的独立应用程序。
- **SA 运行时环境：**提供了一系列预置的强大运行时服务，以及一个独立于语言的相应编程模型。该编程模型经过专门设计，可供各种程序员轻松访问，从脚本编写人员到 Web 开发人员再到经验丰富的企业 Java 程序员都能够轻松访问它。
- **SA 平台资源：**使开发人员能够轻松访问平台的丰富数据对象和自动化操作(如修补程序、配置和审核)以及各种功能(例如，远程访问每个托管服务器的运行时环境)。
- **SA 管理网络：**一套强大的连接、安全和缓存技术，无论设备的位置、IP 地址空间、带宽可用性等特性如何，平台都能够访问设备。
- **SA 托管设备：**通过 SA 管理网络连接到平台的托管服务器和网络设备。

SA 自动化应用程序

如上图所示，自动化应用程序位于堆栈的顶部。它们是用户在平台的基础上编写的应用程序。

自动化应用程序可以是在 SA 运行时环境中运行的 SA 托管应用程序，也可以是在完全独立的环境中运行的独立应用程序。独立应用程序可通过 Web 服务调用远程访问平台。

用户可以在几分钟内编写像简单 Unix shell 脚本一样的简单应用程序。较复杂应用程序(例如与现有源代码控制或工单处理系统集成)的编写时间可能要长一点，并且可能涉及 Python、Microsoft .NET 或 Java 编码。在这些情况下，平台被设计为独立于语言的系统，可供各类开发人员轻松采用。

SA 运行时环境

平台堆栈接下来的是 SA 运行时环境，它提供了一系列预置的强大运行时服务，以及一个独立于语言的相应编程模型。SA 托管的应用程序在 SA 运行时环境中运行。

运行时环境核心包含两个组件：全局 Shell 和全局文件系统。这两个组件一起工作，对所有托管设备进行组织，并提供人们熟悉的 Linux/Unix shell“文件和目录”范例的设备访问方式。

全局 Shell

全局 Shell 是全局文件系统 (OGFS) 的命令行界面。此命令行界面通过在终端窗口中运行的 Linux shell(例如 `bash`)显示。OGFS 将 SA 数据模型以及托管服务器的内容(包括文件)统一放到一个虚拟文件系统中。

全局文件系统

OGFS 以文件目录和文本文件的层次结构表示平台数据模型中的对象(如设施、客户和设备组)以及平台托管设备上的可用信息(如托管网络设备的配置设置或托管服务器的文件系统)。例如，在 OGFS 中，`/opsw/Customer` 目录包含客户对象的详细信息，`/opsw/Server` 目录包含托管服务器的信息。`/opsw/Server` 目录还包含反映托管服务器内容(如文件系统和注册表)的子目录。

该“文件和目录”范例使管理员能够熟悉 shell 脚本以轻松编写脚本，这些脚本通过遍历表示服务器的目录，可在不同服务器上执行相同任务。在后台，全局文件系统安全地向每个托管服务器提供和执行脚本中的任何逻辑。

设备的内容可以通过全局文件系统访问。全局文件系统是一个虚拟文件系统，表示 SA 和 Network Automation (NA) 管理的所有设备。在得到必要安全授权的情况下，最终用户和自动化应用程序可以通过 OGFS 浏览到远程服务器的文件系统。在 Windows 服务器上，管理员还可以访问注册表、II 元数据库和 COM+ 对象。

SA命令行界面

SA 命令行界面 (CLI) 为系统管理员和平台自动化应用程序提供了从命令行调用自动化任务 (例如，配置软件、为设备安装修补程序或运行审核)的方法。通过丰富的语法，用户可将大量对象类型表示为输入内容，或将它们作为 CLI 调用操作的输出内容接收。

CLI 实际上是在平台 API 的基础上以编程方式生成的。将在下一节讨论平台 API。这样做的好处是，一旦开发人员向平台 API 添加新 API，将会自动为其提供相应的 CLI 方法。换句话说，在产品中提供新功能之后，平台中将立即出现相应的 CLI 方法。

SA平台 API

SA 平台 API 是 SA 的 Win32 API：它定义了一系列应用程序编程接口，用于获取和设置值，以及执行操作。SA 用户界面(包括 SA 客户端和 SA 命令行界面 (CLI))都建立在 SA 平台 API 基础之上。该 API 中包括 Java RMI 客户端的库，以及基于 SOAP 的 Web 服务客户端的 WSDL。在具有 Web 服务支持的情况下，程序员可以使用 Perl、C# 和 Python 等流行语言创建客户端。

SA 平台资源

SA 平台资源位于 SA 运行时环境下，开发人员可访问其中丰富的对象和操作，并在自己的应用程序中重用和操作它们。

库存模型

库存模型提供了 SA 收集的关于每个托管设备的所有信息，例如品牌、制造商、CPU、操作系统、已安装的软件等。库存信息通过 SA API 提供，也作为全局文件系统中的文件提供(在 `attr` 子目录中)。库存模型中包括服务器和网络设备等对象。

管理员可以扩展与库存对象关联的数据。例如，如果用户希望存储设备图片、租赁过期日期以及设备所插入的 UPS 的 ID，该平台可以很容易地将这些特性添加到每个设备记录。然后，用户可以添加、删除和处理这些特性，就像处理预置特性一样。

安全模型

安全模型使开发人员能够利用内置的 SA 身份验证和授权安全系统。

平台的所有客户端(SA 提供的管理应用程序、脚本以及最终用户界面)都由相同的安全框架控制。

安全管理员(而非开发人员)将创建用户角色并授予权限。开发人员可以在自己的应用程序环境中重用所有这些用户角色和权限。例如，网络管理员可以编写 shell 脚本，将其与其他网络管理员共享，并确保这些网络管理员只能在他们有权管理的网络设备上运行该脚本。

授权机制将访问权限控制在多个级别：用户可以执行的任务类型、任务访问的服务器和网络设备，以及 SA 对象(如软件策略)。

环境模型

环境模型定义了设备所在的整体业务环境。在通常情况下，设备属于一个或多个客户，位于特定设施中，并属于一个或多个组。平台使应用程序开发人员可以访问其中的每个对象(客户设施、设备组和其他对象)。

与库存对象一样，环境对象也很容易扩展。这使某些操作变得简单，例如，定义在特定数据中心使用的 SNMP 陷阱接收器等特性、仅在特定设施中可用的打印机，或仅由特定业务单位使用的 Apache 配置。

策略模型

策略模型使开发人员能够访问在 SA 中定义的所有最佳实践。策略描述了服务器或网络设备上的期望状态。例如，修补程序策略描述了应在服务器上安装的修补程序，而软件策略描述了应在服务器上安装的软件，等等。

主题内容专家将定义这些策略。任何授权的系统管理员都可使用这些策略来审核设备，以便发现设备上的实际对象是否与需要的对象不同。程序员可以访问这些完整的策略库，以便在自己的应用程序中使用。

软件策略被组织到一些文件夹中，这些文件夹可以定义安全边界。也就是说，应用程序将只能访问他们的用户有权访问的软件策略。

程序包存储库

包存储库使开发人员能够访问 SA 中存储的所有软件和修补程序。其中包括操作系统内部版本、操作系统修补程序、中间件、代理以及用户已上载到 SA 的任何其他软件。

事件库

事件库中包含在执行操作(通过用户界面或通过平台以编程方式执行)时，SA 生成的数字签名的审核跟踪。与其他平台对象一样，这些事件通过编程方式提供。

自动化操作

自动化操作允许开发人员以编程方式启动 SA 可对托管设备执行的任何操作，从运行审核到配置软件再到应用最新操作系统修补程序等。

通过平台可访问 SA 客户端中的最终用户可用的相同功能。这些功能包括一些任务，例如安装修补程序、配置操作系统，以及安装和删除软件策略。实际上，SA 客户端将调用通过 SA 运行时环境以编程方式显示的相同 API。

远程访问

远程访问功能允许开发人员以编程方式访问托管设备的文件系统(对于服务器)和执行环境(对于所有设备)。开发人员可以轻松编写应用程序来检查文件或特定软件包是否存在、运行操作系统命令以检查磁盘使用情况，或者运行系统脚本来执行例行维护任务。

SA 管理网络

管理网络是一个强大的技术组合，使开发人员能够安全地访问所管理的任何设备。管理网络提供了几个重要服务：

- **连接**：允许平台(以及自动化应用程序)访问任何托管设备。
- **安全**：包括基于 SSL/TLS 的加密、身份验证和消息完整性。
- **地址空间虚拟化**：使平台能够在多个重叠的 IP 地址空间中定位服务器。大多数复杂企业网络都具有多个专用 IP 地址空间。
- **可用性**：允许系统体系结构定义任何给定托管设备的冗余路径，即使任何给定网络路径发生故障，也仍然能够访问设备。
- **缓存**：允许服务器从较近的服务器下载软件和修补程序，而不是从较远服务器下载，从而节省时间和网络连接费用。
- **带宽限制**：让系统体系结构确定在 SA 通过网络向特定设备传输数据时，SA 及其任何应用程序可使用的带宽。
- **最低成本路由**：允许系统设计人员设置规则以管理要使用哪个路径访问特定设备，从而最大限度降低网络连接成本。

SA 托管设备

平台堆栈的底部是管理的实际设备。平台可管理超过 65 台服务器操作系统版本，以及超过 35 个不同的网络设备供应商，出厂支持数千个设备型号/版本。

支持的设备列表在不断更新。平台开发人员和脚本编写人员将从此设备列表受益，因为他们的自动化应用程序可以在熟悉的相同编程环境中持续访问内容不断丰富的托管设备库存。

SA 平台的优势

SA 平台具有以下主要优势。

强大的安全性

平台提供了下列全面的安全机制，因此开发人员无需在自己的应用程序中提供这些机制。

- **安全通信通道**：从自动化应用程序到托管设备的端到端通信将进行加密和身份验证。
- **基于角色的访问控制**：平台采用 SA 中内置的基于角色的访问控制，开发人员能够轻松共享其应用程序，并确保它们只在管理员已授予访问权限的设备上运行。
- **数字签名的审核跟踪**：在自动化应用程序运行后，平台将生成数字签名的审核跟踪，捕获运行应用程序的用户、应用程序的执行时间，以及应用程序所在的设备。
- **全面访问**：平台提供对所有设备的全面访问，使系统管理员和开发人员能够轻松访问设备。
- **市场领先的平台覆盖范围**：支持的设备包括 65 个以上的服务器操作系统版本，以及 1000 多种网络设备。
- **位于任何物理位置**：设备可以位于世界上的任何位置，无论是在主要的数据中心、零售商店还是分支办事处。
- **位于任何 IP 地址空间中**：因为平台支持多个重叠的 IP 地址空间，设备可以属于任何 IP 地址空间。
- **在 DMZ 中**：设备可以位于 DMZ 或其他难以访问的网络空间中，因此开发人员或系统管理员无需担心设备访问权限的具体细节(例如，通过防御主机)。

丰富的服务

平台提供了基础自动化系统中的所有相关数据和操作：

- **丰富的预置数据**：开发人员可以很方便地访问丰富的数据，这些数据一部分由平台自身生成(例如，设备库存数据和设施信息)，一部分由与平台交互的用户生成(例如，设备组客户、最佳实践策略以及上载的软件、修补程序和脚本)。开发人员可以轻松编写用于读取和写入这些数据的应用程序。

- **可扩展的数据存储**：开发人员可以轻松扩展本地平台对象，以包含自己的数据。可以扩展设备库存模型，以包含平台本身不包含的特性。可以扩展客户和设施对象，以包含用于引导对该客户相关设备进行配置或审核的特性。
- **自动化任务**：平台几乎向开发人员公开了基础自动化系统的所有功能：安装修补程序、配置、审核及其他。这使得开发人员能够编写跨多个系统的复杂工作流，以便轻松从自动化应用程序环境中调用这些操作。

可供各种程序员轻松访问

平台经过专门设计，适用于各种开发人员，包括 Unix shell 和 Visual Basic 脚本编写人员、Perl 和 Python 程序员以及企业 .NET 或 Java 程序员等。平台的运行时服务层支持以“文件和目录”范例提供大多数平台对象，并从命令行界面 (SA CLI) 获取大多数平台服务。这允许习惯编写 shell 脚本的系统管理员可以立即使用平台，而无需学习使用新的编程语言和工具。他们可以使用自己喜欢的文本编辑器(熟悉的 Unix shell 程序)开始工作，然后快速开发脚本。

对于更复杂的应用程序以及与现有系统的集成，系统程序员可以使用任何具有 Web 服务绑定的编程工具和语言。

SA 平台 API 设计

平台 API 由 Java 接口定义，并被组织到 Java 包中。为了支持多种客户端语言和远程访问协议，API 采用面向函数的“按值调用”模型。

服务

在平台 API 中，一个服务封装一组相关函数。每个服务通过 Java 接口指定，其名称以 Service 结尾，例如 ServerService、FolderService 和 JobService。

服务是 API 的入口点。要访问 API，客户端将调用由服务器接口定义的方法。例如，要检索托管服务器上已安装的软件的列表，客户端将调用 ServerService 接口的 getInstalledSoftware 方法。其他 ServerService 方法示例包括 checkDuplex、setPrimaryInterface 和 changeCustomer。

SA 平台 API 包含 70 多个服务 – 数量太多，无法在这里一一描述。下表列出了一小部分您可能需首先使用到的服务。有关完整服务列表，请在浏览器中转到API 文档和 Twister (第 35 页)中所示的 URL。

SA API 的部分服务列表

服务名称	该服务提供的部分操作
AuditTaskService	创建、获取和运行审核任务。
ConfigurationService	创建应用程序配置，使用应用程序配置获取软件策略。
DeviceGroupService	创建设备组，向组分配设备，获取组成员，设置动态规则。
EventCacheService	触发操作，例如更新值对象的客户端缓存。请参见 事件缓存 (第 34 页)。
FolderService	创建文件夹，获取文件夹的子级，设置文件夹的客户，移动文件夹。
InstallProfileService	创建、获取和更新操作系统安装配置文件。
JobService	获取作业进度和结果，取消作业，更新作业计划。
NasConnectionService	获取 NA 服务器的主机名，在 NA 服务器上运行命令。
NetworkDeviceService	根据指定的搜索筛选器获取系列、名称、型号和类型等信息。

SA API 的部分服务列表(续)

服务名称	该服务提供的部分操作
SequenceService	创建、获取和运行操作系统序列，在服务器上安装操作系统。
ServerService	获取服务器信息，协调(修正)服务器上的策略(安装软件)，获取和设置自定义字段和特性，执行操作系统序列(安装操作系统)。
SoftwarePolicyService	创建软件策略，向服务器分配策略，获取策略内容，修正(协调)服务器策略。
SolPatchService	安装和卸载 Solaris 修补程序，添加策略覆盖。
VirtualColumnService	管理自定义字段和自定义特性。
WindowsPatchService	安装和卸载 Windows 修补程序，添加策略覆盖。

API 中的对象

虽然 SA 平台 API 是面向函数的，但是它的设计允许客户端创建面向对象的库。SA 数据模型包括服务器、文件夹和客户等对象。这是对象是永久对象，也就是说，它们存储在模型库中。在 API 中，这些对象具有以下项：

- 一个用于定义对象行为的服务。例如，ServerService 的方法将指定托管服务器对象的行为。
- 一个表示永久对象实例的对象(身份)引用。例如，ServerRef 是用于唯一标识托管服务器的引用。在 ServerService 中，大多数方法的第一个参数是 ServerRef，它标识由该方法操作的托管服务器。ServerRef 的 Id 特性是存储在模型库中的服务器对象的主键。
- 一个或多个表示永久对象数据成员(特性、字段)的值对象 (VO)。例如，ServerVO 包含 agentVersion 和 loopbackIP 等特性。ServerHardwareVO 的特性包括 manufacturer、model 和 assetTag。大多数特性不能由客户端应用程序更改。如果特性可以更改，则 setter 方法的 API 文档中将包含“Field can be set by clients”。

出于性能方面的考虑，对永久对象的更新操作是粗粒度的。例如，ServerService 的 update 方法接受完整 ServerVO 而非单个特性作为参数。

异常

特定于 SA 的所有 API 异常都派生自以下异常之一：

OpswareException - 在发生应用程序级错误的情况下引发，例如，当最终用户输入将传递给方法的无效值时。通常情况下，客户端应用程序可以从这类异常中恢复。派生自 OpswareException 的异常示例包括 NotFoundException、NotInFolderException 和 JobNotScheduledException。

OpswareSystemException - SA 中发生错误时引发。通常，SA 管理员必须首先解决问题，然后客户端应用程序才能运行。

以下是与安全相关的异常：

AuthenticationException - 在指定了无效的 SA 用户名或密码时引发。

AuthorizationException - 在用户无权执行操作或访问对象时引发。有关权限的详细信息，请参见《SA 10.50 管理指南》。

事件缓存

某些客户端应用程序需要保留 SA 对象的本地副本。缓存由客户端通过 EventCacheService 访问，其中包含用于描述对 SA 对象执行的最近更改的事件。客户端可以定期轮询缓存，以便检查是否已创建、更新或删除对象。该缓存将保留配置的滑动时间窗口内的事件。默认情况下，会保留最近两小时内的事件。要更改滑动窗口大小，请按照 HPE SSO 门户上的《Server Automation 管理指南》中所述来编辑 Web 服务数据访问引擎配置文件。

搜索

SA 平台 API 的搜索机制将根据值对象的的特性(字段)检索对象引用。例如，getServerRefs 方法按 ServerVO 值对象的特性执行搜索。getServerRefs 方法具有以下签名：

```
public ServerRef[] getServerRefs(Filter filter)...
```

每个 get*Refs 方法都接受 filter 参数，即用于指定搜索条件的对象。使用简单表达式的 filter 参数具有以下语法：

```
value-object.attribute  operator  value
```

(该语法已简化。有关完整的定义，请参见筛选器语法 (第 208 页)。)

下面的示例是 getServerRefs 方法的 filter 参数：

```
ServerVO.hostName = "d04.example.com"
ServerVO.model BEGINS_WITH "POWER"
ServerVO.use IN "UNKNOWN" "PRODUCTION"
```

允许使用复杂表达式，例如：

```
(ServerVO.model BEGINS_WITH "POWER") AND (ServerVO.use = "UNKNOWN")
```

并不是值对象的每个特性都可以在 filter 参数中指定。例如，允许在 filter 参数中指定 ServerVO.state，但不允许指定 ServerVO.OsFlavor。要找出允许的特性，请在 API 文档中找到值对象，并查找注释“Field can be used in a filter query”。

安全

SA 平台的用户必须进行身份验证和授权，以在 SA 自动化平台 API 上调用方法。要连接到 SA，客户端需提供 SA 用户名和密码(身份验证)。要调用方法，SA 用户必须属于具有必要权限(授权)的用户组。这些权限不仅会限制用户可执行的操作类型，还会限制对在操作中使用的服务器和网络设备的访问权限。

SA 管理员必须首先在命令中心中指定所需的用户和权限，然后应用程序客户端才能在平台上运行。有关说明，请参见《SA 10.50 管理指南》中的“用户组和设置”一节。有关与安全相关的异常的信息，请参见异常 (第 33 页)。

客户端与 SA 之间的通信已加密。对于 Web 服务客户端，将使用通过 HTTP over SSL (HTTPS) 加密请求和响应 SOAP 消息(执行操作调用)。

API 文档和 Twister

SA 包括用于描述 SA 平台 API 的 API 文档 (Javadoc)。要访问 API 文档，请在浏览器中指定以下 URL：

```
https://<SA_core_host>/twister
```

其中 <SA_core_host> 是运行命令中心组件的 SA 核心服务器的 IP 地址或主机名。

Twister 是一个程序，允许您在浏览器中一次调用一个 API 方法。例如，要调用 ServerService.getServerVO 方法，请执行以下步骤：

1. 在浏览器中打开 API 文档。
2. 在“All Classes”窗格中，选择 com.opsware.server。
3. 在 com.opsware.server 窗格中，选择 ServerService。
4. 在主窗格中，向下滚动到 getServerVO 方法。
5. 对 getServerVO 方法单击“尝试”。

6. 输入您的 SA 用户名和密码。
7. 在 ServerService.getServerVO 的“Twister”窗格中，在 oid 字段中输入托管服务器的 ID。
8. 单击“运行”。“Twister”窗格将显示返回的 ServerVO 对象的特性。

常量字段值

一些 API 值对象 (VO) 的字段值被定义为常量。例如，JobInfoVO 具有 status 字段，该字段可具有由常量定义的值，如 STATUS_ACTIVE、STATUS_PENDING 等。API 将常量指定为 Java static final 字段，但是从 API 生成的 WSDL 不定义常量。要查看常量的定义，请在 API 文档中转到“Constant Field Values”页面：

```
https://<SA_core_host>/twister/docs/constant-values.html
```

例如，“Constant Field Values”页面将 STATUS_ACTIVE 定义为整数 1。

支持的客户端

SA 平台支持具有不同技能的程序员，从编写 shell 脚本的系统管理员到熟悉最新工具和技术的 .NET 和 Java 程序员等。所有支持的客户端都调用一组相同的方法，这些方法被组织到 SA 平台的服务中。开发人员可以创建以下类型的客户端以在 SA 平台 API 中调用方法：

- **SA 命令行界面 (CLI)**：shell 脚本从全局 Shell 会话启动，可通过调用 CLI 方法来访问 SA 平台 API，它们是 OGFS 中的可执行程序。每个 CLI 方法对应于 API 中的一个方法。
- **Web 服务**：这些客户端使用 SOAP over HTTPS 向 SA 发送请求并获得响应。Web 服务操作(在 WSDL 中定义)对应于 API 中的方法。开发人员可以使用 Perl 和 C# 等流行语言编写 Web 服务客户端。
- **Java RMI**：这些客户端从其他 Java 虚拟机调用远程 Java 对象。
- **Pytwist**：这些 Python 程序可以在 SA 核心或托管服务器上运行。

Web 服务和 Java RMI 客户端可以在不同于 SA 核心或托管服务器的服务器上运行。CLI 方法将在安装 OGFS 的核心服务器上的全局 Shell 会话中执行。

SA CLI 方法

最终用户通过 SA 客户端访问 SA。有时，高级用户需要在命令行环境中访问 SA，以便在多台服务器上执行批量操作或重复性任务。在 SA 中，命令行环境由全局 Shell (OGSH)、全局文件系统 (OGFS) 和 SA 命令行界面 (CLI) 方法组成。

要通过命令行执行 SA 操作，可从 OGSH 会话中调用 SA CLI 方法。SA CLI 方法是 OGFS 中的可执行文件，对应于 SA API 中的方法。当运行 SA CLI 方法时，将调用基础 API 方法。

为了理解本节，您需要熟悉 OGSH 和 OGFS。有关详细信息，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的 OGSH。

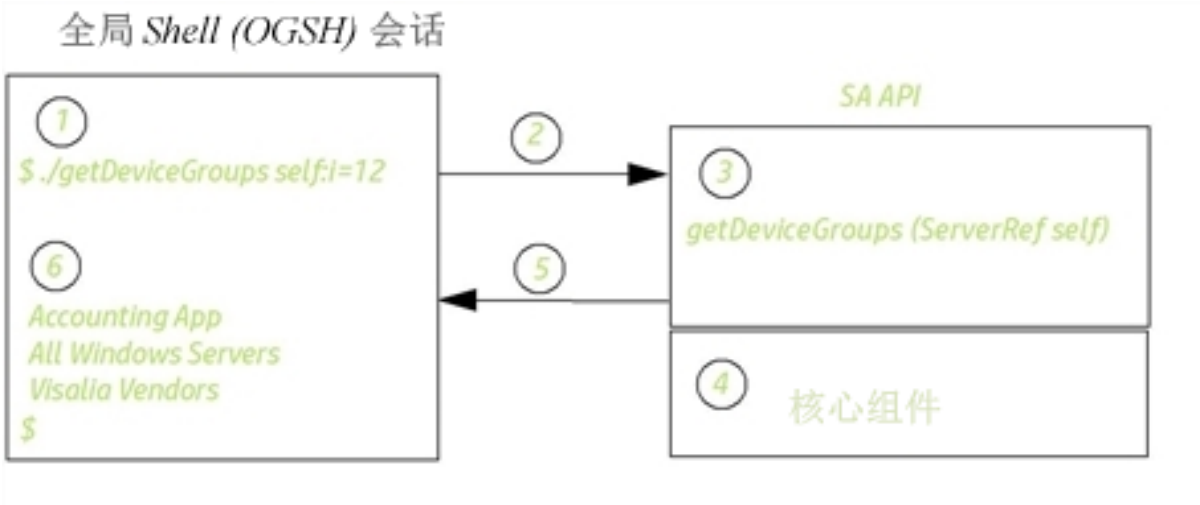
有关 `ouploa`d 和 `odownload` 命令的信息，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的 OCLI 1.0。

方法调用

如下图所示，当在 OGSH 会话中调用 SA CLI 方法时，将执行以下操作：

1. OGSH 解析您输入的命令和参数，以便确定 API 方法。
2. OGSH 调用基础 API 方法。
3. 授权检查过程将验证用户是否有权执行此操作。然后 SA 执行操作。
4. API 方法将结果传送回 SA CLI 方法。
5. SA CLI 方法将返回值写入 OGSH 会话的 `stdout`。如果引发异常，SA CLI 方法将返回非零值状态。

SA CLI 方法调用概述



安全

SA CLI 方法与 SA 客户端使用相同的身份验证和授权机制。当您启动 OGSH 会话时，SA 会对 SA 用户进行身份验证。当运行 SA CLI 方法时，将执行授权操作。要成功运行 SA CLI 方法，您的 SA 用户必须属于具有所需权限的组。有关安全的详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

API 与 SA CLI 方法之间的映射

OGFS 以目录结构表示 SA 对象，以文本文件表示对象特性，以可执行文件表示 API 方法。这些可执行文件即 SA CLI 方法。每个 SA CLI 方法与一个基础 API 方法相匹配。对于这两种类型的方法，方法名称、参数和返回值都相同。

例如，`setCustomer` API 方法具有以下 Java 签名：

```
public void setCustomer(ServerRef self,
          CustomerRef customer)...
```

在 OGFS 中，对应的 SA CLI 方法具有以下语法：

```
setCustomer self:i=server-id customer:i=customer-id
```

请注意，参数名称(`self` 和 `customer`)在这两种语言中是相同的。(`:i` 符号称为格式说明符，将在本节后文进行讨论。)在该示例中，返回类型为 void，因此，SA CLI 方法不会将结果写

入 stdout。有关 SA CLI 方法如何返回用于代表对象的字符串的信息，请参见返回值 (第 57 页)。

SA CLI 方法与 Unix 命令之间的差异

虽然在 OGSH 中可以运行 Unix 命令和 SA CLI 方法，但是 SA CLI 方法在以下几个方面存在不同：

- 与许多 Unix 命令不同，SA CLI 方法不从 stdin 中读取数据。因此，您不能在一组由管道 (|) 连接的命令中插入 SA CLI 方法。(但是，SA CLI 方法会将数据写入 stdout。)
- 大多数 Unix 命令可接受标志和值形式的参数 (例如，ls -l /usr)。在 SA CLI 方法中，命令行参数是以等号连接的“名称-值”对。
- Unix 命令基于文本：它们接受并返回字符串形式的数据。相比之下，SA CLI 方法可以接受和返回复杂的对象。
- 使用 SA CLI 方法时，您可以指定参数和返回值的格式。Unix 命令则没有同等功能。

SA CLI 方法教程

本主题介绍 SA CLI 方法以及一些示例，您可以在自己的环境中试着使用这些示例。完成本教程后，您应该能够运行 SA CLI 方法、检查 SA 对象的 self 文件，并创建用于在多台服务器上调用 SA CLI 方法的脚本。

在开始本教程之前，您需要具有以下能力：

- 可以登录到 SA 客户端。
- 您的 SA 用户至少在一个托管服务器上具有“读取和写入”权限。这些权限通常由安全管理员分配。HPE SSO 门户上的“Server Automation 管理指南”中讨论了这些权限。
- 您的 SA 用户在同一个托管服务器上拥有所有 OGSH 权限。有关这些权限的信息，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的“aaa 实用程序”一节。
- 熟悉 OGSH 和 OGFS。如果是首次使用这些功能，请先参见 HPE SSO 门户上的“Server Automation 用户指南”中的“全局 Shell”一节，然后再继续本教程。

本教程中的示例命令将在名为 abc.example.com 的 Windows 服务器上运行。该服务器属于名为“All Windows Servers”的服务器组。在尝试执行这些命令时，请使用您有权访问的托管服务器的主机名替换 abc.example.com。

1. 打开 OGSH 会话。

   您可以从 SA 客户端中打开全局 Shell 会话。在“操作”菜单中，选择“全局 Shell”。也可以从在桌面上运行的终端客户端中打开 OGSH 会话。有关说明，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的“打开全局 Shell 会话”一节。

2. 列出服务器的 SA CLI 方法。

   特定服务器的 method 子目录中包含一些可执行文件，也就是您可以对该服务器运行的方法。以下示例列出了 abc.example.com 服务器的 SA CLI 方法：

   ```
   $ cd /opsw/Server/@/abc.example.com/method
   $ ls -1
   addDeviceGroups
   attachPolicies
   attachVirtualColumn
   checkDuplex
   clearCustAttrs
   ...
   ```

   这些方法具有实例上下文 - 它们作用于一个特定的服务器实例(在本示例中，为 abc.example.com)。可以从方法的路径推断服务器实例。步骤 5 讨论了具有静态上下文的方法。

3. 运行不带参数的 SA CLI 方法。

   要显示 abc.example.com 所属的公共服务器组，请调用 getDeviceGroups 方法：

   ```
   $ cd /opsw/Server/@/abc.example.com/method
   $ ./getDeviceGroups
   Accounting App
   All Windows Servers
   Visalia Vendors
   ```

4. 运行带参数的方法。

   方法的命令行参数以“名称-值”对的形式表示，各个参数之间由空白字符分隔。在 setCustomer 的以下调用中，参数名称是 customer 且值为 20039。参数名称末尾的 :i 是 ID 格式说明符，将在后面的步骤中讨论它。

   以下方法调用将 abc.example.com 服务器的客户从 Opsware 更改为 C39。客户 C39 的 ID

是 20039。

```
$ cd /opsw/Server/@/abc.example.com
$ cat attr/customer ; echo
Opsware
$ method/setCustomer customer:i=20039
$ cat attr/customer ; echo
C39
```

5. 列出托管服务器的静态上下文方法。

   静态上下文方法驻留在 /opsw/api 目录中。这些方法未被限制到某个特定对象实例。

   要列出服务器的静态方法，请输入以下命令：

   ```
   $ cd /opsw/api/com/opsware/server/ServerService/method
   ```

   ```
   $ ls
   ```

   列出的方法与步骤 2 中显示的方法相同。

6. 运行带 self 参数的方法。

   该步骤会将 getDeviceGroups 作为静态上下文方法调用。与步骤 3 中所示的的实例上下文方法不同，静态上下文方法需要使用 self 参数来标识服务器实例。
   例如，假设 abc.example.com 服务器的 ID 为 530039。要列出该服务器组，请输入以下命令：

   ```
   $ cd /opsw/api/com/opsware/server/ServerService/method
   $ ./getDeviceGroups self:i=530039
   Accounting App
   All Windows Servers
   Visalia Vendors
   ```

   将对 getDeviceGroups 的此调用与步骤 3 中用于演示实例上下文的调用相比较。这两个调用在 API 中运行相同的基础方法，并返回相同的结果。

7. 检查服务器的 self 文件。

   在 SA 中，每个托管服务器都是一个对象。但是，OGFS 是文件系统，而不是对象模型。self 文件提供了对 SA 对象的各种表示形式的访问方法。这些表示形式包括 ID、

名称和结构。

服务器的默认表示形式是服务器名称。例如，要显示服务器的名称，请输入以下命令：

```
$ cd /opsw/Server/@/abc.example.com
$ cat self ; echo
abc.example.com
```

如果您知道服务器的 ID，则可以从 self 文件中获取其名称，如以下示例所示：

```
$ cat /opsw/.Server.ID/530039/self ; echo
abc.example.com
```

8. 在 self 文件上指示 ID 格式说明符。

   要选择 self 文件的特定表示形式，请输入一个句点，然后键入文件名，再键入格式说明符。例如，以下 cat 命令中包含格式说明符 (:i) 以显示服务器 ID：

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat .self:i ; echo
   com.opsware.server.ServerRef:530039
   ```

   此输出显示 abc.example.com 的 ID 是 530039。com.opsware.server.ServerRef 是服务器引用的类名，它是 SA API 中的相应对象。

   > **备注：** 前导句点在文件和方法返回值上的格式说明符中是必需的，但不通过方法参数指示。

9. 指示结构格式说明符。

   结构格式说明符 (:s) 用于表示复杂对象的特性。这些特性将显示为“名称-值”对，所有特性都包含在大括号内。结构格式用于在命令行上指定作为复杂对象的方法参数。有关方法调用的示例，请参见复杂对象和数组作为参数 (第 56 页)。

   以下示例使用结构形式显示了 abc.example.com：

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat .self:s ; echo
   {
   managementIP="192.168.8.217"
   ```

```
modifiedBy="spujare"
manufacturer="DELL COMPUTER CORPORATION"
use="UNKNOWN"
discoveredDate=1149012848000
origin="ASSIMILATED"
osSPVersion="SP4"
locale="English_United States.1252"
reporting=false
netBIOSName=
previousSWReg=1150673874000
osFlavor="Windows 2000 Advanced Server"
...
```

服务器的特性也可由 attr 目录中的文件表示，例如：

```
$ pwd
/opsw/Server/@/abc.example.com
$ cat attr/osFlavor ; echo
Windows 2000 Advanced Server
```

10. 创建用于调用 SA CLI 方法的脚本。

    此步骤中的示例脚本遍历了“All Windows Servers”公共服务器组中的所有服务器。在每台服务器上，脚本将运行 getCommCheckTime SA CLI 方法。

    首先，返回到 OGFS 中的主目录：

    ```
    $ cd
    $ cd public/bin
    ```

    然后，运行 vi 编辑器：

    ```
    $ vi
    ```

    在 vi 中，插入下列行以创建 bash 脚本：

    ```
    #!/bin/bash
    # iterate_time.sh

    METHOD_DIR="/opsw/api/com/opsware/server/ServerService/method"
    GROUP_NAME="All Windows Servers"
    ```

```
cd "/opsw/Group/Public/$GROUP_NAME/@/Server"

for SERVER_NAME in *
do
 SERVER_ID=`cat $SERVER_NAME/.self:i`
 echo $SERVER_NAME
 $METHOD_DIR/getCommCheckTime self:i=$SERVER_ID
 echo
 echo
done
```

在 vi 中保存文件，并将文件命名为 iterate_time.sh。退出 vi。

使用 chmod 更改 iterate_time.sh 的权限，然后运行它:

```
$ chmod 755 iterate_time.sh
$ ./iterate_time.sh
abc.example.com
2006/06/20 16:46:56.000
...
```

格式说明符

格式说明符指示值在 SA CLI 环境的显示或解释方式。可以向方法参数、方法返回类型、self 文件和对象特性应用格式说明符。要指示格式说明符，请在某个字母前加上一个冒号，如下表中所示。

如果为文件或方法返回值指示格式说明符，则必须在文件或方法名称前加上一个句点。对于具有格式说明符的方法返回值，前导句点将不包括在内。

格式说明符摘要

格式说明符	描述	有效对象类型	是否允许用作方法参数?
:n	**名称**：用于标识对象的字符串。建议使用唯一名称，但不是必须使用唯一名称。对于没有名称的对象，此表示形式与 ID 表示形式相同。	SA 对象	是。如果名称不明确，会发生错误。

格式说明符摘要(续)

格式说明符	描述	有效对象类型	是否允许用作方法参数?
:i	**ID**:唯一指示对象类型及其 SA ID 的格式。也成为对象参考。	SA 对象; 日期 (java.util.Calendar) 对象	是。如果已在上下文中清除类型，则可忽略类型。
:s	**结构**：用于在命令行上指定复杂值的精简表示形式。特性包含在大括号中。	任何复杂对象	是
:d	**目录**：以 OGFS 中的目录表示特性。	任何用作特性的复杂对象。该表示形式不能用于方法参数或返回值。	否

格式说明符的位置

格式说明符紧跟在其作用于的项之后。对于文件，格式说明符跟在文件名后面。在以下示例中，请注意前导句点：

```
cat .self:s
```

当应用于方法返回类型时，格式说明符跟在方法名称的后面。以下调用显示了所返回组的 ID：

```
./.getDeviceGroups:i
```

在方法参数中，格式说明符位于参数名称后、等号前，如以下示例所示：

```
./setCustomer self:i=9977 customer:i=239
```

具有格式说明符的方法参数没有前导句点。

默认格式说明符

每个值或对象都具有默认的格式说明符。例如，名称格式说明符是 osVersion 特性的默认格式说明符。以下两个 cat 命令生成相同的输出：

```
cd /opsw/Server/@/d04.example.com/attr
cat osVersion
cat .osVersion:n
```

名称格式说明符是模型库中存储的 SA 对象(例如服务器和客户)的默认格式说明符。结构格式说明符是其他复杂对象的默认格式说明符。

ID 格式说明符示例

以下示例显示了 d04.example.com 服务器所属的设施的 ID：

```
cd /opsw/Server/@/d04.example.com/attr
cat .facility:i ; echo
```

(前面的 echo 命令为可选。它会生成一个换行字符，使输出内容更容易阅读。分号字符分隔在同一行中输入的多个 bash 语句。)

具有 ID 格式说明符的值的输出将以 Java 类名称作为前缀。例如，如果设施值的 ID 为 39，那么前面的 cat 命令会显示以下输出：

```
com.opsware.locality.FacilityRef:39
```

以下 getDeviceGroups 方法调用列出了 d04.example.com 所属的公共服务器组的 ID：

```
cd /opsw/Server/@/d04.example.com/method
./.getDeviceGroups:i
```

有关更多 ID 格式示例，请参见self 文件 (第 52 页)。

结构格式说明符语法

可使用结构格式表示复杂对象。复杂对象中可以包含各种特性。您可以使用此格式来指定作为复杂对象的方法参数。有关示例，请参见复杂对象和数组作为参数 (第 56 页)。

结构格式是一系列“名称-值”对，由空白字符分隔，包含在大括号中。每个“名称-值”对表示一个特性。结构格式具有以下语法：

```
{ name-1=value-1 name-2=value-2 ...}
```

下面是一个简单的示例：

```
{ version=10.1.3 isCurrent=true }
```

可使用任何空白字符作为分隔符：

```
{
       version=10.1.3
       isCurrent=true
}
```

可以将特性指定为结构形式，从而支持表示嵌套对象。在以下示例中，versionDesc 特性以结构表示：

```
{
program=agent
versionDesc={
               version=10.1.3
               isCurrent=true
               comment="Latest version"
               }
}
```

要在一个结构中指定数组，请重复特性名称。以下结构中包含一个名为 steps 数组，该数组具有三个元素，元素值分别为 33、14 和 28。

```
{ moduleName="Some Initiator" steps=33 steps=14 steps=28 }
```

结构格式说明符的例子

以下示例为 facility 特性指定了结构格式：

```
cd /opsw/Server/@/d04.example.com/attr
cat .facility:s
```

此 cat 命令将生成以下输出。请注意，customers 是一个数组，其中将为与该设施关联的每一位客户包含一个元素。

```
{
       modifiedBy="192.168.9.246"
       customers="Customer Independent"
       customers="Not Assigned"
       customers="Opsware Inc."
       customers="Acme Inc."
       . ..
       ontogeny="PROD"
       createdBy=
       status="ACTIVE"
       createdDt=-1
       realms="Transitional"
       realms="C39"
       realms="C39-agents"
       modifiedDt=1146528752000
```

```
        name="C39"
        displayName="C39"
}
```

以下 getDeviceGroups 调用指示返回值的结构格式说明符：

```
cd /opsw/Server/@/d04.example.com/method
./.getDeviceGroups:s
```

此 getDeviceGroups 调用将显示以下输出。由于 d04.example.com 属于两台服务器组，因此输出中包括两个结构。在每个结构中，devices 数组都包含属于该组的服务器的元素。

```
{
        dynamic=true
        devices="m302-w2k-vm1.dev.example.com"
        devices="d04.example.com"
        . ..
        status="ACTIVE"
        34 Chapter 2
        public=true
        fullName="Device Groups Public All Windows Servers"
        description="test"
        createdDt=-1
        modifiedDt=1142019861000
        parent="Public"
}
{
        dynamic=true
        devices="opsware-nibwp.build.example.com"
        devices="glengarriff.snv1.dev.example.com"
        devices="millstreet"
        . ..
        fullName="Device Groups Public z_testsrvgroup"
        . ..
}
```

结构格式说明符是用于检索值对象 (VO) 的方法的默认格式说明符。例如，以下两个 getServerVO 调用是等效的：

```
cd /opsw/Server/@/d04.example.com/method
./.getServerVO:s./getServerVO
```

在该示例中，getServerVO 将显示以下输出：

```
{
        managementIP="192.168.198.93"
        modifiedBy=
        manufacturer="DELL COMPUTER CORPORATION"
        use="UNKNOWN"
        discoveredDate=1145308867000
```

```
        origin="ASSIMILATED"
        osSPVersion="RTM"
        locale="English_United States.1252"
        reporting=false
        netBIOSName=
        previousSWReg=1147678609000
        osFlavor="Windows Server 2003, Standard Edition"
        peerIP="192.168.198.93"
        modifiedDt=1145308868000
        . ..
        serialNumber="HVKZS51"
}
```

该结构代表 SA API 的 ServerVO 类。该结构中的每一个特性对应于 attr 目录中的一个文件。在以下示例中，getServerVO 和 cat 命令都将显示服务器的 serialNumber 特性：

```
cd /opsw/Server/@/d04.example.com
./method/getServerVO | grep serialNumber
cat attr/serialNumber ; echo
```

目录格式说明符示例

以下命令将当前工作目录更改为与服务器 d04.example.com 关联的客户：

```
cd /opsw/Server/@/d04.example.com/attr/.customer:d
```

以下命令列出该客户的名称：

```
cat /opsw/Server/@/d04.example.com/attr/\
.customer:d/attr/name
```

只能在需使用目录名称的命令参数中使用目录说明符。以下 cat 命令因为尝试显示目录而失败：

```
cat /opsw/Server/@/d04.example.com/attr/.customer:d # WRONG!
```

但是，以下命令是合法的：

```
ls /opsw/Server/@/d04.example.com/attr/.customer:d
```

值表示形式

因为 SA CLI 方法在 shell 环境 (OGSH) 中运行，所以它们可接受和返回字符串形式的数据。但是，基础 API 方法可以接受和返回其他数据类型，例如数字、布尔值和对象。下面的各节描述了 OGFS 和 SA CLI 方法是如何表示非字符串数据类型的。

OGFS 中的 SA 对象

SA 数据模型包括服务器、服务器组、客户和设施等对象。在 OGFS 中，这些对象以目录结构表示：

```
/opsw/Customer
/opsw/Facility
/opsw/Group
/opsw/Library
/opsw/Realm
/opsw/Server
...
```

以上列表并非完整的列表。要查看完整列表，请输入 `ls /opsw`。

对象特性

SA 对象的特性以 `attr` 子目录中的文本文件表示。每个文件的名称与相应特性的名称匹配。文件的内容显示特性的值。

例如，`/opsw/Server/@/buzz.example.com/attr` 目录包含以下文件：

```
agentVersion
codeset
createdBy
createdDt
customer
defaultGw
36 Chapter 2
description
discoveredDate
facility
```

```
hostName
locale
lockInfo
loopbackIP
managementIP
manufacturer
...
```

要显示 buzz.example.com 服务器的管理 IP 地址，请输入以下命令：

```
cd /opsw/Server/@/buzz.example.com/attr
cat managementIP ; echo
```

自定义特性

自定义特性是可由您分配给服务器等 SA 对象的“名称-值”对。在 OGFS 中，自定义特性以 CustAttr 子目录中的文本文件表示。您可以在 CustAttr 下创建新文本文件，从而在 OGSH 会话中创建自定义特性。以下示例在 buzz.example.com 服务器上创建一个值为 hello there 的自定义特性 MyGreeting：

```
cd /opsw/Server/@/buzz.example.com/CustAttr
echo -n "hello there" > MyGreeting
```

有关更多示例，请参见《SA 10.50 用户指南》中的“管理自定义特性”一节。

self 文件

self 文件位于服务器或客户等 SA 对象的目录中。此文件提供对当前对象的各种表示形式的访问，具体取决于格式说明符。有关详细信息，请参见格式说明符 (第 45 页)。

要列出 buzz.example.com 服务器的 ID，请输入以下命令：

```
cd /opsw/Server/@/buzz.example.com
cat .self:i ; echo
```

对于服务器，默认格式说明符是名称。下列命令将显示相同的输出：

```
cat self ; echo
cat .self:n ; echo
```

以下命令将以结构格式列出服务器的特性：

```
cat .self:s
```

基元值

下表显示了基元值是如何在 API 与其 SA CLI 方法中的字符串表示形式之间转换的。除了日期，基元值不支持格式说明符。日期支持 ID 格式说明符。

基元类型和 SA CLI 方法之间的转换

基元类型	Java 等效项	SA CLI 方法的输出	SA CLI 方法的输入
字符串	java.lang. String	字符串，以当前会话的编码显示。	字符串，从当前会话编码转换为 Unicode。
数字	byte、short、int、long、float、double；以及它们的对象等效项	十进制格式，未本地化。适用于非常大或非常小的值的科学记数法。	示例 - 十进制:101、512.34、 -104 十六进制:0x1F32, 0x2e40 八进制:0543 科学记数法:4.3E4、6.532e-9、1.945e+02
布尔值	boolean、Boolean	true 或 false	字符串“true”和所有大小写混合变体将计算为 true。所有其他值将计算为 false。
二进制数据	byte[]、Byte[]	二进制字符串。不从会话编码进行转换。	二进制字符串。不向会话编码进行转换。
日期	java.util. 日历	日期值。默认情况下，以下列格式显示： YYYY/MM/DD HH:MM:SS.mmm 时间以 UTC 格式显示。如果指示 ID 格式说明符，则值显示为自新纪元开始所经过的毫秒数(UTC 格式)。	与输出相同。

数组

数组对象的表示形式取决于它们是独立对象(数组特性文件或方法返回值)，还是包含在复杂对象的结构中。

首先，独立数组对象将根据基础类型显示，并由换行字符分隔。在数组元素中，换行字符由“\n”转义，反斜杠字符由“\\”转义。

可以使用基础类型所支持的任何表示形式来输入或输出数组值。例如，默认情况下，getDeviceGroups 方法会以名称形式列出组：

```
All Windows Servers
Servers in Austin
Testing Pool
```

如果指示 ID 格式说明符，(.getDeviceGroups:i)方法将显示组的 ID：

```
com.opsware.device.DeviceGroupRef:15960039
com.opsware.device.DeviceGroupRef:10390039
com.opsware.device.DeviceGroupRef:17380039
```

其次，包含在复杂对象的结构中的数组将表示为“名称-值”对(使用特性作为名称)。特性将出现多次，为数组中的每个元素出现一次。特性的显示顺序将确定数组中元素的顺序。以下示例显示了一个结构，该结构包含两个特性，一个是名为 subject 的字符串，一个是名为 ranks 的包含三个元素的数字数组：

```
{ subject=”my favorites” ranks=17 ranks=44 ranks=24 }
```

数组还可以目录表示。在数组目录中，每个数组元素都具有一个对应文件(基元类型)或子目录(复杂类型)。每个条目的名称是数组元素的索引号，从零开始。

对于作为复杂对象特性的数组，应通过编辑其特性文件来修改数组。此操作将使用编辑过的文件内容完全替换数组。

对于包含复杂对象作为元素的数组，应通过更改其目录表示形式来修改数组。要更改元素值，请编辑元素文件。例如，假设您具有一个包含 5 个字符串元素的数组。ls 命令将按照如下所示列出这些元素：

```
0 1 2 3 4
```

以下命令将更改第三个元素的值：

```
echo -n "My new value" > 2
```

SA CLI 方法参数和返回值

本节将详细讨论方法上下文(实例或静态)、参数用法、返回值和退出状态。

方法上下文与 self 参数

在 OGFS 中，一个方法驻留在多个位置。方法的位置与其上下文(实例或静态)相关。

具有实例上下文的方法驻留在特定 SA 对象的 method 目录中。方法调用不需要 self 参数。方法的位置隐含了受该方法影响的对象实例。以下示例将更改 d04.example.com 服务器的客户：

```
cd /opsw/Server/@/d04.example.com/method
./setCustomer customer:i=9
```

具有静态上下文的方法驻留在 /opsw/api 下的某个位置。方法调用需使用 self 参数来确定受该方法影响的实例。在以下静态上下文示例中，self:i 指定了托管服务器的 ID：

```
cd /opsw/api/com/opsware/server/ServerService/method
./setCustomer self:i=230054 customer:i=9
```

在命令行上传递参数

命令行参数将指定为“名称-值”对的形式，由等号 (=) 连接。“名称-值”对由一个或多个空白字符(通常是空格)分隔。命令行上的名称与 SA API 中对应 Java 方法的参数名称相匹配。

例如，在 SA API 中，setCustomField 方法具有以下定义：

```
public void setCustomField(CustomFieldReference self,
        java.lang.String fieldName, java.lang.String strValue)...
```

以下 SA CLI 方法示例将为 ID 为 3670039 的服务器的自定义字段指定值：

```
cd /opsw/api/com/opsware/server/ServerService/method
./setCustomField self:i=3670039 \
fieldName="Service Agreement" strValue="Gold"
```

编者注: 通过运行它检查上述命令 (待定)

如前一节中所述，具有实例上下文的方法无需使用 self 参数。以下 setCustomField 示例与前面的示例等效：

```
cd /opsw/.Server.ID/3670039
./setCustomField \
fieldName="Service Agreement" strValue="Gold"
```

可以按照任意顺序指定命令行参数。以下两个 SA CLI 方法调用等效：

```
./setCustomField fieldName="My Stuff" strValue="abc"
./setCustomField strValue="abc" fieldName="My Stuff"
```

要为参数指定 Null 值，请忽略该参数，或在等号后插入一个空格。在以下示例中，myParam 的值为 Null：

```
./someMethod myField="more info" myParam= anotherParam=9834
./someMethod myField="more info"         anotherParam=9834
```

指定参数的类型

如果某个方法具有抽象类型的参数，则必须指定具体类型和值。在以下示例中，必须指定 com.opsware.folder.FolderRef 类型：

```
cd /opsw/api/com/opsware/folder/FolderService/method
./remove self:i="com.opsware.folder.FolderRef:730555"
```

如果不指定具体类型，将显示以下错误消息：

Object type type-name is abstract.Specify a concrete sub-type.

复杂对象和数组作为参数

要传递作为复杂对象的参数，请将该对象的特性括在大括号中，如结构格式说明符语法 (第 47 页)中所示。

以下示例将创建公共服务器组 AllMine。create 方法具一个参数 pattern，该参数的特性 parent 及 shortName 包含在大括号中。在该示例中，getPublicRoot 返回 2340555，即顶层公共组的 ID。

```
cd /opsw/api/com/opsware/device/DeviceGroupService/method
./.getPublicRoot:i ; echo
./create “pattern={ parent:i=2340555 shortName=’AllMine’ }”
```

通过重复参数名称(每个数组元素一次)，指定数组参数。例如，以下 assign 方法调用将指定名为 policies 的数组参数中的前两个元素：

```
cd /opsw/api/com/opsware/swmgmt
cd SoftwarePolicyService/method
./attachPolicies self:i=4220039 \
policies:i=4400335 policies:i=4400942
```

重载方法

如果多个同类方法具有相同的名称但不同的参数列表，则 Java 方法名称被重载。在重载 SA CLI 方法中，命令行上的参数名称表示要调用的方法。例如，setCustomField 方法被重载，以支持设置不同的数据类型。下面两个命令将调用不同版本的方法：

```
./setCustomField \
fieldName="Service Agreement" strValue="Gold"
./setCustomField \
fieldName=hmp longValue=2245
```

返回值

如果位于 SA CLI 方法下的 API 方法返回一个值，则 SA CLI 方法会将该值输出到 stdout。与 Unix 命令一样，您可以将方法的 stdout 重定向到文件，或将其分配给环境变量。

要更改返回值的表示形式，请在方法名称中插入一个前导句点并附加一个格式说明符。以下示例返回 ID 而不是默认名称形式的服务器引用：

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i
```

如果所指示的格式说明符与方法的返回类型不兼容，文件系统将发出错误消息作为响应。

退出状态

与 Unix shell 命令类似，SA CLI 方法使用退出状态 ($?) 表示调用结果。退出状态为零值表示成功，非零值表示错误。SA CLI 方法将错误消息输出到 stderr。

SA CLI 方法的退出状态代码

退出状态	类别	描述
0	成功	该方法已成功完成。
1	命令行解析错误	用于方法调用的命令行格式错误，无法解析为一组选项 (--option[=value]) 和参数值 (param=value)。

SA CLI 方法的退出状态代码(续)

退出状态	类别	描述
2	参数解析错误	参数值无法解析为 API 所需的对象类型。
3	API 用法错误	由于用法错误(如无效的参数值)，导致调用失败。
4	访问错误	用户无权执行操作。
5	其他错误	发生除退出状态 1-4 所表示的错误之外的其他错误。

例如，以下 bash 脚本将检查 getDeviceGroups 方法的退出状态：

```
#!/bin/bash

cd /opsw/Server/@/toro.snv1.corp.example.com/method
./getDeviceGroups
cmnd_exit_status=$?
if [ $cmnd_exit_status -eq 0 ]
then
       echo "The command was successful."
else
       echo "The command failed."
       echo "Exit status = " $cmnd_exit_status
fi
```

SA CLI 方法将调用基础 API 方法。如果 API 方法引发异常，SA CLI 方法将返回非零的退出状态。在调试方法调用时，查看关于已引发异常的信息将有助于解决问题。The OGFS 中的 /sys/last-exception 文件包含最近的 API 调用所引发的异常的堆栈跟踪。在阅读此文件后，系统将丢弃该文件的内容。

搜索筛选器和 SA CLI 方法

SA API 中的许多方法可接受对象引用作为参数。要根据搜索条件检索对象引用，可以调用 findServerRefs 和 findJobRefs 等方法。例如，您可以调用 findServerRefs 以搜索 hostname 特性中包含 example.com 的所有服务器。

搜索语法

findServerRefs 等方法具有以下语法：

```
findobjectRefs filter='[object-type:]expression'
```

filter 参数包含一个表达式，用于指定搜索条件。可以使用圆括号或大括号将表达式括起来。简单表达式具有以下语法：

value-object.attribute operator value

该语法已简化。有关完整的定义，请参见筛选器语法 (第 208 页)。

搜索示例

大多数 SA 对象类型具有关联的查找工具方法。本节介绍其中一小部分查找工具方法的用法。要了解如何在其他 SA CLI 方法中使用搜索功能，请参见示例脚本 (第 62 页)。

查找服务器

查找主机名包含 example.com 的服务器：

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i \
filter='device:{ ServerVO.hostname CONTAINS example.com }'
```

查找使用特性值为 UNKNOWN 或 PRODUCTION 的服务器：

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i \
filter='{ ServerVO.use IN "UNKNOWN" "PRODUCTION" }'
```

以下 bash 脚本显示如何搜索服务器、在临时文件中保存服务器 ID，然后将每个 ID 指定为另一个方法调用的参数。该脚本显示了每个 Linux 服务器所属的公共组。

```
#!/bin/bash


TMPFILE=/tmp/server-list.txt

rm -f $TMPFILE


cd /opsw/api/com/opsware/server/ServerService/method


./.findServerRefs:i \

filter='{ ServerVO.osVersion CONTAINS Linux }' > $TMPFILE
```

```
for ID in `cat "$TMPFILE"`
do
  echo Server ID:$ID
  ./getDeviceGroups self:i=$ID
  echo
done
```

查找作业

本节中的示例将返回服务器审核或策略修正等作业的 ID。

查找已成功完成的作业：

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i filter='job:{ job_status = "SUCCESS" }'
```

(有关 job_status 的允许值的列表，请参见《SA 10.50 集成指南》中的“作业批准集成”。)

查找已成功完成或发生警告的作业：

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i \
filter='job:{ job_status IN "SUCCESS" "WARNING" }'
```

查找今天已开始的作业：

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i \
filter='job:{ JobInfoVO.startDate IS_TODAY "" }'
```

查找所有服务器审核作业：

```
cd /opsw/api/com/opsware/job/JobService/method
./findJobRefs \
filter='job:{ JobInfoVO.description = "Server Audit" }'
```

查找已在 ID 为 280039 的服务器上运行的作业：

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i filter='job:{ job_device_id = "280039" }'
```

查找今天已失败的作业：

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i \
filter='job:{ (( JobInfoVO.startDate IS_TODAY "" ) \
& ( job_status = "FAILURE" )) }'
```

查找其他对象

本节包含一些用于搜索软件策略和包的示例。

查找由 SA 用户 jdoe 创建的软件策略：

```
cd /opsw/api/com/opsware/swmgmt/SoftwarePolicyService/method
./.findSoftwarePolicyRefs:i \
filter='{ SoftwarePolicyVO.createdBy CONTAINS jdoe }'
```

查找用于 Windows 2003 平台的含有 ismtool 的 MSI：

```
cd /opsw/api/com/opsware/pkg/UnitService/method
./.findUnitRefs:i \
filter='software_unit:{ ((UnitVO.unitType = "MSI") \
& ( UnitVO.name contains "ismtool" ) \
& ( software_platform_name = "Windows 2003" )) }'
```

查找名为 117170-01 的 Solaris 修补程序：

```
cd /opsw/api/com/opsware/pkg/solaris/SolPatchService/method
./.findSolPatchRefs:i filter='{name = 117170-01}'
```

查找名称包含字符串 Test、父文件夹名为 My Stuff 的文件夹。

```
cd /opsw/api/com/opsware/folder/FolderService/method
./.findFolders:s \
filter='( ( FolderVO.name CONTAINS "Test" ) \
& (folder_parent_name = "My Stuff" ) )'
```

可搜索的特性和有效运算符

并不是值对象的每个特性都可以在搜索筛选器中指定。例如，您可以搜索 ServerVO.use 但不能搜索 ServerVO.OsFlavor。

要找出给定对象类型的可搜索的特性，请调用 getSearchableAttributes 方法。以下示例列出了可在搜索表达式中指定的 ServerVO 的特性：

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableAttributes searchableType=device
```

searchableType 参数用于指示对象类型。要确定 searchableType 的允许值，请输入以下命令：

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableTypes
```

要找出特性的有效运算符，请调用 getSearchableAttributeOperators 方法。以下示例列出了 ServerVO.hostname 特性的有效运算符(如 CONTAINS 和 IN)：

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableAttributeOperators searchableType=device \

searchableAttribute=ServerVO.hostname
```

示例脚本

本节包含用于调用各种 SA CLI 方法的简单 bash 脚本的代码列表。这些脚本演示了如何在命令行上传递方法参数，包括复杂对象和 self 参数。如果确定要复制和粘贴这些示例脚本，您需要更改一些硬编码对象的名称，例如 d04.example.com 服务器。有关如何在 OGFS 中创建和运行脚本的教程说明，请参见SA CLI 方法教程 (第 40 页)。

脚本remediate_policy.sh (第 67 页)将创建一个软件策略、向该策略添加包，并通过调用 startFullRemediateNow 方法在托管服务器上安装该包。

create_custom_field.sh

该脚本将创建一个名为 TestFieldA 的自定义字段(虚拟列)、将该字段附加到所有服务器，然后在一台服务器上设置该字段的值。只有在附加之后，自定义字段才会出现在 SA 客户端中。可以为服务器、设备组或软件策略创建自定义字段。要创建自定义字段，您的 SA 用户必须属于具有“管理虚拟列”权限的用户组。

与自定义特性不同，自定义字段将应用于某个类型的所有实例。有关在 OGFS 中创建自定义特性的示例，请参见 HPE SSO 门户上“Server Automation 用户指南”中的“管理自定义特性”。

create_custom_field.sh 脚本中包含以下代码：

```
#!/bin/bash
# create_custom_field.sh

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

# Create a virtual column.
# Remember the name because you cannot search for the
# displayName.
./create vo='{ name=TestFieldA type=SHORT_STRING \
displayName="Test Field A" }'

column_id=`./.findVirtualColumn:i name=TestFieldA`

echo --- column_id = $column_id

cd /opsw/api/com/opsware/server/ServerService/method

# Attach the column to all servers.
# All servers will have this custom field.
./attachVirtualColumn virtualColumn:i=$column_id
```

```
# Get the ID of the server named d04.example.com
devices_id=‘./.findServerRefs:i \
filter=\
’device:{ ServerVO.hostname CONTAINS "d04.example.com" }’‘

echo --- devices_id = $devices_id

# Set the value of the custom field (virtual column) for
# a specific server.
./setCustomField self:i=$devices_id fieldName=TestFieldA \
strValue="This is something."
```

create_device_group.sh

该脚本将创建一个静态设备组，并向组中添加服务器。然后，该脚本将创建一个动态组、设置该组的规则，并刷新该组的成员资格。该脚本的最后一条语句将列出属于该动态组的设备。

下面是该脚本的代码：

```
#!/bin/bash
# create_device_group.sh

cd /opsw/api/com/opsware/device/DeviceGroupService/method

# Get the ID of the public root group (top of hierarchy).
public_root=‘./.getPublicRoot:i‘

# Create a public static group.
./create "vo={ parent:i=$public_root shortName=’Test Group A’ }"

# Get the ID of the group just created.
group_id=‘./.findDeviceGroupRefs:i \
```

```
filter=’{ DeviceGroupVO.shortName = "Test Group A" }’ ‘

echo --- group_id = $group_id

cd /opsw/api/com/opsware/server/ServerService/method

# Get the ID of the server named d04.example.com
devices_id=‘./.findServerRefs:i \
filter=\
’device:{ ServerVO.hostname CONTAINS "d04.example.com" }’‘

echo --- devices_id = $devices_id

cd /opsw/api/com/opsware/device/DeviceGroupService/method

# Add a server to the device group.
./addDevices \
self:i=$group_id devices:i=$devices_id

# Create a dynamic device group.
./create \
"vo={ parent:i=$public_root \
shortName=’Test Dyn B’ dynamic=true }"

# Get the ID of the device group.
dynamic_group_id=‘./.findDeviceGroupRefs:i \
filter=’{ DeviceGroupVO.shortName = "Test Dyn B" }’ ‘

echo --- dynamic_group_id = $dynamic_group_id

# Set the rule so that this group contains servers with
```

```
# hostnames containing the string example.com.
# The rule parameter has the same syntax as the filter
# parameter of the find methods.
./setDynamicRule self:i=$dynamic_group_id \
rule=’device:{ ServerVO.hostname CONTAINS example.com }’

# By default, membership in dynamic device groups is refreshed
# once
# an hour, so force the refresh now.
./refreshMembership selves:i=$dynamic_group_id now=true

# Display the names of the devices that belong to the group.
echo --- Devices in group:
./getDevices selves:i=$dynamic_group_id
```

create_folder.sh

该脚本将创建名为 /Test 1的文件夹、列出根 (/)文件夹下的文件夹，然后创建子文件夹 /Test 1/Test 2。创建这些文件夹后，您可以在 SA 客户端的导航窗格的“库”中查看它们。

下面是该脚本的代码：

```
#!/bin/bash
# create_folder.sh

cd /opsw/api/com/opsware/folder/FolderService/method

# Get the ID of the root (top) folder.
root_id=`./.getRoot:i`

# Create a new folder under the root folder.
./create vo="{ name='Test 1' folder:i=$root_id }"
```

```
# Display the names of the folders under the root folder.
./getChildren self:i=$root_id

# Get the ID of the folder "/Test 1"
folder_id=`./.getFolderRef:i path="Test 1"`

# Create a subfolder.
./create vo="{ name='Test 2' folder:i=$folder_id }"

# Get the ID of the folder "/Test 1/Test 2"
folder_id=`./.getFolderRef:i path="Test 1" path="Test 2"`
echo folder_id = $folder_id
```

remediate_policy.sh

该脚本将在现有文件夹 Test 2 中创建名为 TestPolicyA 的软件策略、向该策略添加包含 ismtool 的包、将该策略附加到一个(而不是一组)服务器，然后修正该服务器。修正操作会启动在服务器上安装包的作业。可以在 SA 客户端中检查作业的进度和结果。有关使用 SA CLI 方法搜索作业的示例，请参见查找作业 (第 60 页)。

在该脚本中，SoftwarePolicyService 的 create 方法中的 platforms 参数值为硬编码。在大部分示例脚本中，已通过按名称搜索对象来避免硬编码。对于平台，很难按 name 特性执行搜索，因为该特性不同于 displayName 特性，它显示在 SA 客户端中，但是无法搜索。用于查找平台 ID 的最简单方法是，转到 Twister 并运行不带任何参数的 PlatformService.findPlatformRefs 方法。

该脚本中的 update 方法将对 softwarePolicyItems(如果软件数据库包含很多名称相似的包，则很难按名称搜索到这个对象)的 ID 进行硬编码。一种用于获取 ID 的方法是，运行 SA 客户端，按字段(例如“文件名”和“操作系统”)搜索“软件”，打开通过搜索找到的包，并记录包的属性视图中的 SA ID。

在下面的列表中，update 方法中存在错误的换行符。如果要复制此代码，请编辑脚本，以便 vo 参数位于一行中。

下面是 remediate_policy.sh 脚本的源代码：

```
#!/bin/bash
# remediate_policy.sh
```

```
# Get the ID of the folder where the policy will reside.
cd /opsw/api/com/opsware/folder/FolderService/method
folder_id=`./.findFolders:i filter='{ FolderVO.name = "Test 2" }'`

cd /opsw/api/com/opsware/swmgmt/SoftwarePolicyService/method

# Create a software policy named TestPolicyA.
# This policy resides in the folder located in the preceding findFolders
# call.
# The platform for this policy is Windows 2008 (ID 160076)
./create vo="{ platforms:i=160076 name="TestPolicyA" \
folder:i=$folder_id lifecycle=AVAILABLE }"

policy_id=`./.findSoftwarePolicyRefs:i \
filter='{ SoftwarePolicyVO.name = "TestPolicyA" }'`

echo --- policy_id = $policy_id

# Call the update method to add a package to the software policy.
# The package ID for the "ismtool" msi installer is 4010001.

# Note that "force = true" is required.

./update self:i=$policy_id force=true \
vo='{ softwarePolicyItems:i=com.opsware.pkg.windows.MSIRef:4010001 }'

cd /opsw/api/com/opsware/server/ServerService/method

# Get the ID of the server named d04.opsware.com
devices_id=`./.findServerRefs:i \
```

```
filter='device:{ ServerVO.hostname CONTAINS "d04.opsware.com" }'`

echo --- devices_id = $devices_id

# Attach the policy to a single server (not a group).
./attachPolicies self:i=$devices_id \
policies:i=$policy_id

# Remediate the server to install the package in the policy.
job_id=`./.startFullRemediateNow:i self:i=$devices_id`

echo --- job_id = $job_id
```

remove_custom_field.sh

虽然在操作环境中不会经常执行自定义字段删除操作，但是在测试环境中有时必须执行该操作。请注意，必须首先取消附加自定义字段，然后才能将其删除。

下面是 remove_custom_field.sh 的代码：

```
#!/bin/bash
# remove_custom_field.sh

if [ !-n "$1" ]
   then
   echo "Usage:‘basename $0‘ <name>"
   echo "Example:‘basename $0‘ hmp"
   exit
fi

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

column_id=‘./.findVirtualColumn:i name=$1‘
```

```
echo --- column_id = $column_id

cd /opsw/api/com/opsware/server/ServerService/method

# Column must be detached before it can be removed.
./detachVirtualColumn virtualColumn:i=$column_id

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

# Remove the virtual column.
./remove self:i=$column_id
```

schedule_audit_task.sh

该脚本将启动审核任务，计划在将来某个日期运行。在采用 SA CLI 方法时，将使用以下语法指定日期参数：

```
YYYY/MM/DD HH:MM:SS.sss
```

用于启动任务的 startAudit 方法将返回执行审核的作业的 ID。有关使用 SA CLI 方法搜索作业的示例，请参见查找作业 (第 60 页)。

下面是 schedule_audit_task.sh 的代码：

```
#!/bin/bash
# schedule_audit_task.sh

cd /opsw/api/com/opsware/compliance/sco/AuditTaskService/method

# Get the ID of the audit task to schedule.

audit_task_id=`./.findAuditTask:i \
filter='audit_task:{ (( AuditTaskVO.name BEGINS_WITH "HW check" ) \
& ( AuditTaskVO.createdBy = "gsmith" )) }'`
```

```
echo --- audit_task_id = $audit_task_id

# Schedule the audit task for Oct. 16, 2013.
# In the startDate parameter, note that the last delimiter for the time
# is a period, not a colon.

job_id=`./.startAudit:i self:i=$audit_task_id
schedule:s='{ startDate="2013/10/16 00:00:00.000" }' \
notification:s='{ onFailureOwner="sjones@opsware.com" \
onFailureRecipients="jdoe@opsware.com" \
onSuccessOwner="sjones@opsware.com" \
onSuccessRecipients="jdoe@opsware.com" }'`

echo --- job_id = $job_id
```

获取 SA CLI 方法的用法信息

在将来的发行版中，SA CLI 方法将显示用法信息。在此之前，您可以使用下列各节所述的方法从 API 文档或 OGFS 中获取必需的信息。

列出服务

SA API 方法已组织到一些服务中。要找出可用于 SA CLI 方法的服务，请在 OGSH 会话中输入以下命令：

```
cd /opsw/api/com/opsware
find .-name "*Service"
```

要列出 API 文档中的服务，请在浏览器中指定以下 URL：

```
https://occ_host:1032
```

其中 occ_host 是运行命令中心组件的核心服务器的 IP 地址或主机名。

查找 API 文档中的服务

OGFS 中的服务路径将映射到 API 文档中的 Java 包名称。例如，在 OGFS 中，ServerService 方法显示在以下目录中：

```
/opsw/api/com/opsware/server
```

在 API 文档中，以下接口定义了这些方法：

```
com.opsware.server.ServerService
```

列出服务的方法

在 OGFS 中，您可以列出某个服务的 method 目录的内容。例如，要显示 ServerService 的方法名称，请输入以下命令：

```
ls /opsw/api/com/opsware/server/ServerService/method
```

在 API 文档中，执行以下步骤查看 ServerService 的方法：

在左上部的窗格中，选择 com.opsware.server。

在左下部的窗格中，选择 ServerService。

在主窗格中，向下滚动查看方法。

列出方法的参数

在 API 文档中，执行上一节中所述的步骤。在服务接口页面的方法详细信息部分中，查看参数和返回类型。有关方法参数的详细信息，请参见在命令行上传递参数 (第 55 页)。

获取关于值对象的信息

API 文档中介绍到，一些服务方法会传递或返回值对象 (VO)，其中包含数据成员(特性)。例如，ServerService.getServerVO 方法返回 ServerVO 对象。要找出 ServerVO 所包含的特性，请执行以下步骤：

在 API 文档中，选择 ServerVO 链接。可以在多个位置找到该链接：

- getServerVO 的方法签名
- com.opsware.server 的类列表(左下部窗格)
- 在索引页上。API 文档的主窗格顶部提供了一个指向索引页的链接。
- 在 ServerVO 页面上，记录 getter 和 setter 方法。每个 getter-setter 对与值对象中的一个特性相对应。例如，getCustomer 和 setCustomer 指示 ServerVO 包含名为 customer 的特性。

确定特性是否可修改

只有少数对象特性可由客户端应用程序修改。要找出可修改的特性，请执行以下步骤：

1. 在 API 文档中，转到值对象页面，如上一节中所述。
2. 在 setter 方法的方法详细信息部分中，查找“Field can be set by clients”。

对于 OGFS 中表示的 SA 对象(如服务器和客户)，可通过检查 attr 目录中文件的访问类型，确定可修改的特性。具有读写 (rw) 访问类型的文件对应于可修改的特性。例如，要列出某台服务器的可修改特性，请输入以下命令：

```
cd /opsw/Server/@/server-name/attr
ls -l | grep rw
```

确定某个特性是否可用于筛选器查询中

要确定是否可以将值对象的某个特性用于筛选器查询(搜索)中，请执行以下步骤：

1. 在 API 文档中，转到值对象页面。
2. 在对应于该特性的 getter 方法的方法详细信息部分中，查找字符串“Field can be used in a filter query”。

从 OGSH 会话中，要确定是否可以搜索某个特性，请使用可搜索的特性和有效运算符 (第 62 页)中所述的方法。

通过 Pytwist 执行 Python API 访问

Pytwist 概述

Pytwist 是一组 Python 库，允许托管服务器和自定义扩展访问 SA API。(twist 是 Web 服务数据访问引擎的内部名称。)对于托管服务器，您可以设置将通过 Pytwist 调用 SA API 的 Python 脚本，以使最终用户可以将这些脚本作为 DSE 或 ISM 控件调用。自定义扩展是由 HPE SA Professional Services 创建的 Python 脚本，可在命令引擎 (Way) 中运行。Pytwist 允许自定义扩展访问 SA 数据模型中最近添加的内容(例如文件夹和软件策略)，这些内容无法从命令引擎脚本进行访问。

本主题的目标读者包括熟悉 SA 数据模型、自定义扩展、代理和 Python 编程语言的开发人员和顾问。

Pytwist 安装

在尝试执行本节中的示例之前，请确保您的环境满足下列各节中详细说明的安装要求。

Pytwist 支持的平台

Pytwist 在托管服务器和核心服务器上受支持。有关这些服务器支持的操作系统的列表。

Pytwist 依赖于 SA 代理和自定义扩展所使用的 Python 2.7.10 版。

与 Web 服务和 Java RMI 客户端不同，Pytwist 客户端依赖于内部 SA 库。如果您的客户端程序需要从非托管或核心的服务器访问 SA API，请使用 Web 服务或 Java RMI 客户端而不是 Pytwist。

Pytwist 的访问要求

Pytwist 需要访问正在运行 Web 服务数据访问引擎的核心服务器的端口 1032。默认情况下，引擎在端口 1032 上侦听。

安装 Pytwist 库

Pytwist 库是代理库的一部分，因此无需安装。

Pytwist 示例

本节中的 Python 代码示例显示了如何从托管服务器获取信息、创建文件夹以及修正软件策略。每个 Pytwist 示例将执行以下操作：

1. 导入包。

   在导入 SA API 命名空间的对象(如 Filter)时，路径中将包含 Java 包名称，且该名称前将附加 pytwist。下面是 get_server_info.py 的 import 语句示例：

   ```
   import sys
   from pytwist import *
   from pytwist.com.opsware.search import Filter
   ```

2. 创建 TwistServer 对象：

   ```
   ts = twistserver.TwistServer()
   ```

   有关方法参数的信息，请参见 TwistServer 方法语法。

3. 获取对服务的引用。

   服务的 Python 包名称与 Java 包名称相同，但是没有前导 opsware.com。例如，Java com.opsware.server.ServerService 包映射到 Pytwist server.ServerService：

   ```
   serverservice = ts.server.ServerService
   ```

4. 调用服务的 SA API 方法：

   ```
   filter = Filter()
   ...
   servers = serverservice.findServerRefs(filter)
   ...
   for server in servers: vo = serverservice.getServerVO(server)
   ...
   ```

get_server_info.py

该脚本将搜索主机名包含命令行参数的所有托管服务器。搜索方法 findServerRefs 将返回一组对服务器永久对象的引用。对于每一个引用，getServerVO 方法将返回值对象 (VO)，

它是用于保存服务器特性的数据表示形式。下面是 get_server_info.py 脚本的代码：

```
#!/opt/opsware/agent/bin/python
# get_server_info.py

# Search for servers by partial hostname.

import sys
from pytwist import *
from pytwist.com.opsware.search import Filter

# Check for the command-line argument.
if len(sys.argv) < 2:
       print "You must specify part of the hostname as the search target."
       print "Example:" + sys.argv[0] + " " + "opsware.com"
       sys.exit(2)

# Construct a search filter.
filter = Filter()
filter.expression = 'device_hostname *=* "%s" ' % (sys.argv[1])

# Create a TwistServer object.
ts = twistserver.TwistServer()

# Get a reference to ServerService.
serverservice = ts.server.ServerService

# Perform the search, returning a tuple of references.
servers = serverservice.findServerRefs(filter)

if len(servers) < 1:
       print "No matching servers found"
       sys.exit(3)

# For each server found, get the server's value object (VO)
# and print some of the VO's attributes.
for server in servers:
       vo = serverservice.getServerVO(server)
       print "Name:" + vo.name
       print "Management IP:" + vo.managementIP
       print "OS Version:" + vo.osVersion
```

create_folder.py

该脚本将通过调用 createPath 方法创建名为 /TestA/TestB 的文件夹。请注意，createPath 的 path 参数不包含斜杠。path 中的每个字符串元素表示文件夹中的一个层级。然后，该

脚本将检索并打印根文件夹的所有下级文件夹的名称。create_folder.py 脚本的列表如下所示：

```
#!/opt/opsware/agent/bin/python
# create_folder.py

# Create a folder in SA.

import sys
from pytwist import *

# Create a TwistServer object.
ts = twistserver.TwistServer()

# Get a reference to FolderService.
folderservice = ts.folder.FolderService

# Get a reference to the root folder.
rootfolder = folderservice.getRoot()
# Construct the path of the new folder.
path = 'TestA', 'TestB'

# Create the folder /TestA/TestB relative to the root.
folderservice.createPath(rootfolder, path)

# Get the child folders of the root folder.
rootchildren = folderservice.getChildren(rootfolder,
'com.opsware.folder.FolderRef')

# Print the names of the child folders.
for child in rootchildren:
       vo = folderservice.getFolderVO(child)
       print vo.name
```

remediate_policy.py

该脚本将创建一个软件策略，并将其附加到服务器，然后修正该策略。脚本中的服务器名称是硬编码的：平台、服务器和父文件夹。您也可以在命令行中指定策略名称。如果要多次运行脚本，则该操作很方便。软件策略的平台必须与策略中所含的包的操作系统匹配。因此，如果更改硬编码的平台名称，则还需更改 unitfilter.expression 中的名称。

下面的列表中存在多个错误的换行符。如果要复制此代码，请首先修复这些错误的换行符，然后再运行代码。以“NOTE”开头的注释行指出了错误的换行符。

```
#!/opt/opsware/agent/bin/python
# remediate_policy.py
```

```
# Create, attach, and remediate a software policy.

import sys
from pytwist import *
from pytwist.com.opsware.search import Filter
from pytwist.com.opsware.swmgmt import SoftwarePolicyVO

# Initialize the names used by this script.
foldername = 'TestB'
platformname = 'Windows 2003'
servername = 'd04.example.com'
# If a command-line argument is specified,
# use it as the policy name
if len(sys.argv) == 2:
       policyname = sys.argv[1]
else:
       policyname = 'TestPolicyA'

# Create a TwistServer object.
ts = twistserver.TwistServer()
ts.authenticate("SAUser", "SAPassword")

# Get the references to the services used by this script.
folderservice = ts.folder.FolderService
swpolicyservice = ts.swmgmt.SoftwarePolicyService
serverservice = ts.server.ServerService
unitservice = ts.pkg.UnitService
platformservice = ts.device.PlatformService

# Search for the folder that will contain the policy.
folderfilter = Filter()
folderfilter.expression = 'FolderVO.name = %s' % foldername
folderrefs = folderservice.findFolderRefs(folderfilter)
if len(folderrefs) == 1:
       parent = folderrefs[0]
elif len(folderrefs) < 1:
       print "No matching folders found."
       sys.exit(2)
else:
       print "Non-unique folder name:" + foldername
       sys.exit(3)
# Search for the reference to the platform "Windows Server 2003."
platformfilter = Filter()
platformfilter.objectType = 'platform'
# Because the platform name contains spaces,
# it’s enclosed in double quotes
# NOTE:The following code line has a bad line break.
# The assignment statement should be on a single line.
```

```
platformfilter.expression = 'platform_name = "%s"' % platformname
platformrefs = platformservice.findPlatformRefs(platformfilter)

if len(platformrefs) == 0:
       print "No matching platforms found."
       sys.exit(4)

# Search for the references to some software packages.
unitfilter = Filter()
unitfilter.objectType = 'software_unit'
# NOTE:The following code line has a bad line break.
# The assignment statement should be on a single line.
unitfilter.expression = '((UnitVO.unitType = "MSI") & ( UnitVO.name contains
"ismtool" ) & ( software_platform_name = "Windows 2003" ))'
unitrefs = unitservice.findUnitRefs(unitfilter)

# Create a value object for the new software policy.
vo = SoftwarePolicyVO()
vo.name = policyname
vo.folder = parent
vo.platforms = platformrefs
vo.softwarePolicyItems = unitrefs

# Create the software policy.
swpolicyvo = swpolicyservice.create(vo)

# Search by hostname for the reference to a managed server.
serverfilter = Filter()
serverfilter.objectType = 'server'
# NOTE:The following code line has a bad line break.
# The assignment statement should be on a single line.
serverfilter.expression = 'ServerVO.hostname = %s' % servername
serverrefs = serverservice.findServerRefs(serverfilter)

if len(serverrefs) == 0:
       print "No matching servers found."
       sys.exit(5)

# Create an array that has a reference to the
# newly created policy.
swpolicyrefs = [1]
swpolicyrefs[0] = swpolicyvo.ref

# Attach the software policy to the server.
swpolicyservice.attachToPolicies(swpolicyrefs, serverrefs)

# Remediate the policy and the server.
# 注意：The following code line has a bad line break.
# The assignment statement should be on a single line.
```

```
jobref = swpolicyservice.startRemediateNow(swpolicyrefs, serverrefs)
print "The remediation job ID is %d" % jobref.id
```

虚拟化 Pytwist 示例

本节提供了使用 SA API 创建和部署虚拟机 (VM) 的示例。有关虚拟化的更多示例，请参见 HPE SSO 门户上的“Server Automation 用户指南”。

createVM_WithOSBP.py

此基本示例使用含有静态 IP 配置的 CD 启动在 VMware vCenter 上创建虚拟机。

在这些示例中，所有属性尚未设置。请参考 API 文档 (javadoc) 了解并为您的用例设置属性。

```
#!/opt/opsware/agent/bin/python
from pytwist import twistserver
from pytwist.com.opsware.locality import CustomerRef, RealmRef
from pytwist.com.opsware.osprov import OSBuildPlanRef
from pytwist.com.opsware.pkg import UnknownPkgRef
from pytwist.com.opsware.v12n import AdapterIPSettings, V12nHypervisorRef, \
       V12nHypervisorService, V12nInventoryFolderRef, V12nResourcePoolRef, \
       V12nResourcePoolRef, V12nVIManagerService, VirtualCpuConfig,
VirtualDevice, \
       VirtualDeviceChangeConfig, VirtualDeviceTypeConstant,
VirtualHardwareConfigSpec, \
       VirtualMemoryConfig, VirtualServerCDProvisioningSpec,
VirtualServerComputeSpec, \
       VirtualServerConfigSpec, VirtualServerCreateSpec,
VirtualStorageDeviceConstant, \
       VirtualStorageDeviceHWConfig
from pytwist.com.opsware.v12n.vmware import V12nDatastoreRef, \
       VmwareVirtualInterfaceBacking, VmwareVirtualNicHWConfig, \
       VmwareVirtualServerDetails, VmwareVirtualServerStorageSpec, \
       VmwareVirtualStorageFileBacking
import time


# This is a bare bones example of creating a Virtual Machine on a VMware
# vCenter while booting from CD with Static IP configuration.It also
# provisions the Virtual Machine with the give OS Build Plan.For more
# detailed information please refer to the java doc. All the properties have
# not been set in the example below, please review the java doc to understand
# and set the properties for your use case.
```

```
#This method constructs the create specification to create the Virtual
# Machine and provision it.
def constructCreateSpec():

        #Construct VmwareVirtualServerDetails
        detail = VmwareVirtualServerDetails()
        # Virtual Machine Name
        detail.name = "Test VM"
        # Description for the Virtual Machine
        detail.description = "Sample test create VM"
        # This is the key for the guest operating system that will installed on
        # the Virtual Machine.
        #V12nVIManagerService.getGuestOSList() provides the supported list for
        # the given V12n Manager and hypervisor.
        detail.guestId = "rhel6Guest"
        # This is folder where the VM will reside in you can see the list of
        # folders at V12nInventoryFolderService.findV12nInventoryFolderRefs() it
        # is the inventory location of the Virtual Machine
        folder = V12nInventoryFolderRef(2020001)
        detail.inventoryFolderRef = folder


        # Configure the number of Virtual processors on the Virtual Machine
        cpuConfig = VirtualCpuConfig()
        cpuConfig.virtualCpuCount = 1

        # Configure the Memory for the Virtual Machine
        memoryConfig = VirtualMemoryConfig()
        memoryConfig.size = 1024*1024*1024

        # Configure NICs
        # Construct the virtual device of type network i.e a NIC
        virtualNetworkDevice = VirtualDevice()
        virtualNetworkDevice.type = VirtualDeviceTypeConstant.NETWORK
        # A unique identifier for the virtual device
        virtualNetworkDevice.key = "4001"
        backingNetwork = VmwareVirtualInterfaceBacking()
        # This is the port group that the nic will be assigned to
        backingNetwork.portGroup = "VLAN 625"

        hwConfigNetwork = VmwareVirtualNicHWConfig()
        # The kind of network adapter to use, other options are listed in
        # VmwareVirtualNicHWConfig
        hwConfigNetwork.adapterType = VmwareVirtualNicHWConfig.E1000
        hwConfigNetwork.macAddressIsDynamic = True

        virtualNetworkDevice.hwConfig = hwConfigNetwork
        virtualNetworkDevice.backingInfo = backingNetwork
```

```
        virtualNetworkDevice.connected = True
        virtualNetworkDevice.startConnected = True

        # Configure Hard Disk
        virtualDiskDevice = VirtualDevice()
        virtualDiskDevice.type = VirtualDeviceTypeConstant.STORAGE

        backingStorage = VmwareVirtualStorageFileBacking()

        # This is Ref for the data store on the hypervisor where the VM will be
        # hosted.The list of datastores associated with the Hypervisors are
        # listed at V12nHypervisorService.getV12nHypervisorVO() under storage
        # config
        dataStoreRef = V12nDatastoreRef(90001)
        backingStorage.datastore = dataStoreRef
        backingStorage.lazyAllocation = True

        hwConfigStorage = VirtualStorageDeviceHWConfig()
        hwConfigStorage.capacity = 10*1024*1024*1024
        hwConfigStorage.usageType =
VirtualStorageDeviceConstant.USAGE_TYPE_DISK_DRIVE

        virtualDiskDevice.hwConfig = hwConfigStorage
        virtualDiskDevice.backingInfo = backingStorage

        # Add both the virtual devices to be created, i.e. the hard disk and the
        # nic
        virtualDvcs_toAdd = []
        virtualDvcs_toAdd.append(virtualNetworkDevice)
        virtualDvcs_toAdd.append(virtualDiskDevice)
        deviceChange = VirtualDeviceChangeConfig()
        deviceChange.addList = virtualDvcs_toAdd

        # Finalize the Config Spec
        configSpec = VirtualServerConfigSpec()
        configSpec.detail = detail
        configSpec.virtualHardware = VirtualHardwareConfigSpec()
        configSpec.virtualHardware.cpuConfig = cpuConfig
        configSpec.virtualHardware.memoryConfig = memoryConfig
        configSpec.virtualHardware.deviceChange = deviceChange

        # Constructing the Compute Spec
        computeSpec = VirtualServerComputeSpec()
        # This is the hypervisor hosting the VM
        hypervisorRef = V12nHypervisorRef(2030001)
        computeSpec.computeProviderRef = hypervisorRef
        # This is resource pool on the hypervisor/cluster that the VM belongs to
        # It can be retrieved by using hypervisorVO.children or the Cluster
        # children
```

```
resourcePool = V12nResourcePoolRef(2040001)
computeSpec.resourcePoolRef = resourcePool

storageSpec = VmwareVirtualServerStorageSpec()
storageSpec.datastore = dataStoreRef

# This example deals with provisioning a VM through CD boot and with
# static IP configuration.The example deals setting the boot ISO and
# network information to be used.
#All the information for this is contained in the
# VirtualServerCDProvisioningSpec

# Set all the network information
gateways =[]
gw ="192.168.135.33"
gateways.append(gw)
dnsServers =[]
dnsServer = "192.168.2.13"
dnsServers.append(dnsServer)


interfaces =[]
interface = AdapterIPSettings()

# Construct the network interface
interface.useDHCP=False
# Note this is the virtual device we have created above, we use the same
# device key to indicate to provisioning which virtual device is to be
# used for provisioning
interface.virtualDeviceKey="4001"
interface.gateways=gateways
interface.ipAddress="192.168.135.45"
interface.netmask="255.255.255.224"
interface.dnsServerList=dnsServers

interfaces.append(interface)

# This is the boot ISO Ref that will be used to get the server into
# maintenance mode
# The name and the id need to match the packages on the core.
# Use the UnitService.findUnitRefs() to find the boot ISO's
bootISORef = UnknownPkgRef(5340001)
bootISORef.name="HPSA_linux_boot_cd.iso"
# The realm assigned to the Virtual Machine will be the realm of the
# Virtualization Service
realmRef = RealmRef(30001)
# The OS Build Plan that needs be run on the Virtual Machine after the VM
# has been created.
osbpRef = OSBuildPlanRef(580001)
```

```
        provisioningSpec = VirtualServerCDProvisioningSpec()

        provisioningSpec.bootISORef = bootISORef
        provisioningSpec.interfaces = interfaces
        provisioningSpec.realmRef = realmRef
        provisioningSpec.oSBuildPlanRef = osbpRef

        # Finally put together all the information to be set on the Create
        # Specification
        createSpec = VirtualServerCreateSpec()
        createSpec.configSpec = configSpec
        createSpec.computeSpec = computeSpec
        createSpec.storageSpec = storageSpec

        createSpec.provisioningSpec = provisioningSpec
        #Set the customer to be associated with the Virtual Machine
        customer = CustomerRef(9)
        createSpec.setCustomerRef(customer)
        return createSpec

def createVirtualMachine():
        twist = twistserver.TwistServer()
        twist.authenticate("hp", "opsware")
        vmService = twist.v12n.V12nVirtualServerService
        createSpec = constructCreateSpec()
        jobRef = vmService.startCreate(createSpec,4*60*60,"Sample create
VM",None, None)


createVirtualMachine()
```

deployVM.py

此基本示例显示如何在 VMware vCenter 上从虚拟机模板部署虚拟机以及自定义已部署虚拟机的来宾 OS。

在这些示例中，所有属性尚未设置。请参考 API 文档 (javadoc) 了解并为您的用例设置属性。

```
#!/opt/opsware/agent/bin/python
from pytwist import twistserver
from pytwist.com.opsware.locality import CustomerRef, RealmRef
from pytwist.com.opsware.osprov import OSBuildPlanRef
from pytwist.com.opsware.pkg import UnknownPkgRef
from pytwist.com.opsware.v12n import AdapterIPSettings, V12nHypervisorRef, \
```

```
        V12nHypervisorService, V12nInventoryFolderRef, V12nResourcePoolRef, \
        V12nResourcePoolRef, V12nVIManagerService, VirtualCpuConfig,
VirtualDevice, \
        VirtualDeviceChangeConfig, VirtualDeviceTypeConstant,
VirtualHardwareConfigSpec, \
        VirtualMemoryConfig, VirtualServerCDProvisioningSpec,
VirtualServerComputeSpec, \
        VirtualServerConfigSpec, VirtualServerCreateSpec,
VirtualStorageDeviceConstant, \
        VirtualStorageDeviceHWConfig, V12nVirtualServerTemplateRef, \
        VirtualServerCloneSpec, VirtualServerGuestCustomizationSpec
from pytwist.com.opsware.v12n.vmware import V12nDatastoreRef, \
        VmwareVirtualInterfaceBacking, VmwareVirtualNicHWConfig, \
        VmwareVirtualServerDetails, VmwareVirtualServerStorageSpec, \
        VmwareVirtualStorageFileBacking
import time

# This is a bare bones example of deploying a Template VMware vCenter.It
# deploys the template and then guest customizes the deployed Virtual Machine.
# For more detailed information please refer to the java doc. All the
# properties have not been set in the example below, please review the java
# doc to understand and set the properties for your use case.

#This method constructs the deploy specification to deploy the Template and
# customizes it.
def constructDeploySpec(sourceTemplateVO):

        #Construct the Deploy Spec
        clonespec = VirtualServerCloneSpec()

        clonespec.computeSpec = VirtualServerComputeSpec()
        # This is the hypervisor hosting the VM
        targetHypervisorRef = V12nHypervisorRef(2030001)
        clonespec.computeSpec.computeProviderRef = targetHypervisorRef

        computeSpec = VirtualServerComputeSpec()
        # This is the resource pool on the hypervisor/cluster that the VM belongs to
        # It can be retrieved by using hypervisorVO.children or the Cluster
        # children
        targetResourcePoolRef = V12nResourcePoolRef(2040001)
        computeSpec.resourcePoolRef = targetResourcePoolRef
        clonespec.computeSpec.resourcePoolRef = targetResourcePoolRef

        storageSpec = VmwareVirtualServerStorageSpec()
        dataStoreRef = V12nDatastoreRef(90001)
        storageSpec.datastore = dataStoreRef
        clonespec.storageSpec = storageSpec
        # Construct VmwareVirtualServerDetails
        detail = VmwareVirtualServerDetails()
```

```
        # Virtual Machine Name
        detail.name = "Test Deploy VM"
        # Description for the Virtual Machine
        detail.description = "Sample Deploy create VM"

        # This is the folder where the VM will reside in. You can see the list of
        # folders at V12nInventoryFolderService.findV12nInventoryFolderRefs().It
        # is the inventory location of the Virtual Machine
        targetFolderRef = V12nInventoryFolderRef(2020001)
        detail.inventoryFolderRef = targetFolderRef
        configSpec = VirtualServerConfigSpec()
        configSpec.detail = detail
        clonespec.configSpec=configSpec

        # Create the Guest Customization Spec, this is needed to customized the
        # deployed VM so that it does not use the network settings and host name
        # of the source template
        # In this example all the interfaces are set to DHCP but you can
        # customize each of the interfaces by either providing static or DHCP
        # configuration details
        interfaces = createInterfaces(sourceTemplateVO)
        # The realm assigned to the Virtual Machine will be the realm of the
        # Virtualization Service
        realmRef = RealmRef(30001)
        clonespec.guestCustomizationSpec =
createGuestCustomizationSpec("testDeployVM",realmRef,interfaces)
        clonespec.setPowerOn(True)
        # Set the customer to be associated with the Virtual Machine
        customerRef = CustomerRef(9)
        clonespec.customerRef = customerRef
        return clonespec

def createGuestCustomizationSpec(newVmNameVal,realmRef,interfaces):
        gcSpec = VirtualServerGuestCustomizationSpec()
        gcSpec.computerName = newVmNameVal
        gcSpec.interfaces = interfaces
        gcSpec.realmRef = realmRef
        return gcSpec

def createInterfaces(virtualServerVO):
        interfaces = []
        virtualDevices = virtualServerVO.virtualHardware.deviceList
        vNICs = [vd for vd in virtualDevices if vd.type ==
VirtualDeviceTypeConstant.NETWORK]
        for vNIC in vNICs:
                intf = AdapterIPSettings()
                intf.useDHCP = True
                intf.hardwareAddress = vNIC.hwConfig.macAddress
                intf.virtualDeviceKey = vNIC.key
```

```
                interfaces.append(intf)
        return interfaces


def deployVirtualMachine():
        twist = twistserver.TwistServer()
        twist.authenticate("hp", "opsware")
        vmTemplateService = twist.v12n.V12nVirtualServerTemplateService
        vmService = twist.v12n.V12nVirtualServerBaseService
        sourceTemplateRef = V12nVirtualServerTemplateRef(1520001)
        sourceTemplateVO =
vmService.getV12nVirtualServerBaseVO(sourceTemplateRef)
        deploySpec = constructDeploySpec(sourceTemplateVO)
        jobRef =
vmTemplateService.startDeploy(sourceTemplateRef,deploySpec,30*60,"Sample
Deploy VM",None, None);

deployVirtualMachine()
```

Pytwist 详细信息

本节描述了特定于 Pytwist 的行为和语法。

身份验证模式

Pytwist 客户端的身份验证模式很重要，因为它会影响客户端可以访问的 SA 功能及资源。Pytwist 客户端可在以下模式中运行：

- **已验证**：客户端已在 `TwistServer` 对象上调用 `authenticate(username, password)` 方法。在调用 `authenticate` 方法后，客户端被授予 `username` 参数所指定的 SA 用户权限，这与登录到 SA 客户端的最终用户非常相似。
- **未验证**：客户端未调用 `TwistServer.authenticate` 方法。在托管服务器上，会将客户端验证为类似代理证书控制设备的身份。在自定义扩展中使用时，未验证的 Pytwist 客户端需要访问命令引擎证书。有关自定义扩展和证书的详细信息，请联系技术支持代表。

TwistServer 方法语法

`TwistServer` 方法用于配置从客户端到 Web 服务数据访问引擎的连接。(有关示例调用，请参见 Pytwist 示例。)`TwistServer` 的所有参数都是可选的。下表列出了参数的默认值。

TwistServer 方法的参数

参数	描述	默认值
host	要连接到的主机名。	twist
port	要连接到的端口号。	1032
secure	是否对连接使用 https 协议。允许的值：1 (true) 或 0 (false)。	1
ctx	连接的 SSL 上下文。	无。(另请参见身份验证模式 (第 90 页)。)

当创建 `TwistServer` 对象时，客户端不会建立与服务器的连接。因此，如果发生连接问题，直到客户端调用 `authenticate` 或 SA API 方法时才会遇到此问题。

错误处理

如果 TwistServer.authenticate 方法或 SA API 方法遇到问题，会引发 Python 异常。可以在 except 子句中捕获这些异常，如下例所示：

```
# Create the TwistServer object.
ts = twistserver.TwistServer('localhost')
# Authenticate by passing an SA user name and password.
try:
       ts.authenticate('jdoe', 'secretpass')
except:
       print "Authentication failed."
       sys.exit(2)
```

将 Java 包名称和数据类型映射到 Pytwist

Pytwist 接口适用于 Python，但 SA API 是使用 Java 语言编写的。由于这两种编程语言之间的差异，Pytwist 客户端必须采用本节所述的映射规则。

在 SA API 文档中，Java 包名称以 com.opsware 开头。当在 Pytwist 中指定包名称时，请在开头处插入 pytwist，例如：

```
from pytwist.com.opsware.compliance.sco import *
```

SA API 文档指定方法参数并返回 Java 数据类型的值。下表显示对于 Pytwist 中的 API 方法调用如何从 Java 数据类型映射到 Python。

从 Java 数据类型映射到 Python

SA API 中的 Java 数据类型	Pytwist 中的 Python 数据类型
Boolean	整数 1 表示 true，整数 0 表示 false。
Object[] (object array)	作为 API 方法调用的输入参数，对象数组可以是 Python 元组或列表。作为 API 方法调用的输出，对象数组将以 Python 元组形式返回。
Map	字典
Date	long 数据类型，表示自新纪元(1970 年 1 月 1 日午夜)开始所经过的毫秒数。

自动化平台扩展 (APX)

本主题描述了如何创建和管理自动化平台扩展 (APX)，通常简称为“扩展”。APX 提供了一个框架，允许任何熟悉基于脚本的编程工具(如 Shell 脚本、Python、Perl 和 PHP)的用户扩展 SA 功能以及创建与 SA 紧密集成的应用程序。SA 提供两种类型的 APX：

- **程序 APX**(又称为**脚本 APX**)在全局文件系统 (OGFS) 中运行，并且可以使用所有 OGFS 功能。您可以使用典型的编程方法来利用 SA API 以及访问核心的托管服务器，以实现新的自定义功能。例如，可以编写一个 APX，用于从托管服务器收集 BIOS 信息并使用 shell 命令填充自定义字段。请参见 程序 APX (第 94 页)。
- **Web APX** 可用于创建基于 Web 的应用程序，在其中使用 GET 或 POST URL 调用 Apache 2.x 进程或 CGI/PHP 脚本。Web APX 可包含静态 Web 资源(如图像)，并且可以使用 CGI 或 PHP 进行动态内容生成。请参见 Web APX (第 95 页)。

APX 允许您访问托管环境的数据，以及与 Web 应用程序、脚本、程序和其他应用程序共享并处理该数据。下面是 APX 的一些优势：

- 列在 SA 库中，可从 SA 客户端使用。
- 通过版本控制进行唯一标识和管理。
- 安全，因为它们可充分利用 SA 的安全模型。当需要时，APX 可以在 APX 会话中安全地将用户权限临时升级到正常默认值之上。
- 可在 SA 核心内部和 SA 核心之间进行扩展。
- 可以计划将它们自动推送到服务器。
- 可审核。
- 可通过 SA 平台升级一直保持有效。在升级后，无需重新编写 APX。

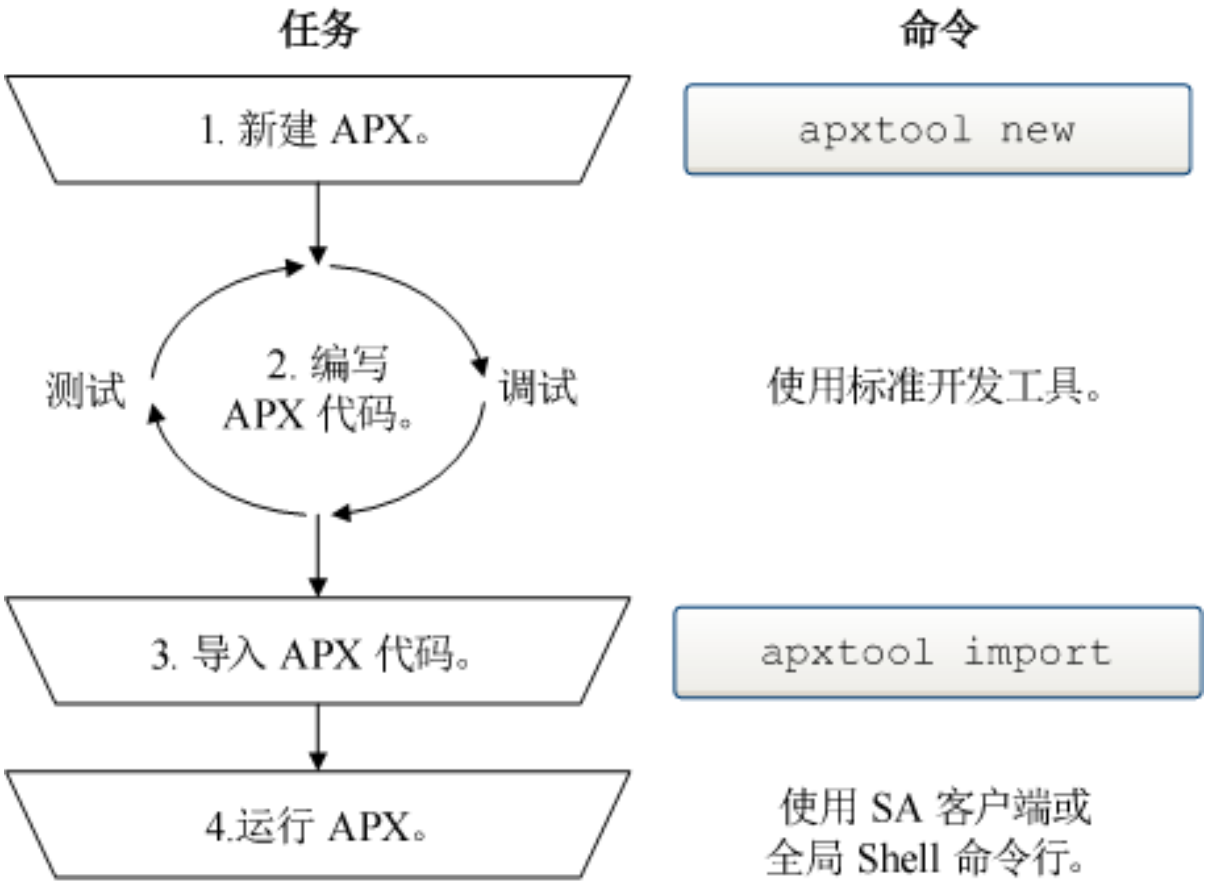
有关使用 APX 扩展的信息，请参见《SA 10.50 用户指南》中的“运行 SA 扩展”一节。另请参见《SA 10.50 用户指南》中的“SA 全局 Shell”一节，这是因为您也可以从 SA 全局 Shell 中运行 APX 扩展。

创建 APX

下图显示了用于创建 APX 及要使用的对应命令的基本步骤。有关如何创建 Web APX 的教程，请参见教程:创建 Web 应用程序 APX (第 116 页)。有关如何创建程序 APX 的教程，请

参见教程:创建程序 APX (第 122 页)。

创建 APX



1. 要创建新 APX，请使用 `apxtool new` 命令。此命令将创建一组模板文件。可以编辑这些文件来创建自己的 APX。

   可以选择使用 `apxtool new` 注册新 APX。注册 APX 时将在 SA 中保留 APX 的名称。如果不在此步骤中注册 APX，可以在下面的步骤 3 中使用 `apxtool import` 命令进行注册。

   请参见 apxtool 命令 (第 102 页)。
2. 在创建 APX 模板文件后开发 APX 代码，方法是修改 `apxtool new` 命令所创建的模板文件并根据需要添加自己的文件。可以对 APX 代码进行测试，确保它能够正常运行。
3. 测试 APX 代码之后，必须使用 `apxtool import` 命令将其导入 SA。
4. 从 SA 客户端或全局 Shell 命令行中运行 APX。

- 在 SA 客户端中：选择“库”>“按类型”选项卡>“扩展”>“程序”。选择 APX。选择“操作”>“运行”菜单。
- 从全局 Shell 命令行：选择“工具”>“全局 Shell”菜单，从 SA 客户端打开全局 Shell。通过输入以下命令运行 APX：
  `/opsw/apx/bin/<APX 名称>`。

  有关更多信息，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的“运行 SA 扩展”和“SA 全局 Shell”章节。

要创建用于在 VMware ESXi 服务器上运行的 APX 扩展，该 APX 扩展必须使用其 Web 服务接口与 ESXi 服务器进行远程通信。有关 VMware ESXi 服务器的详细信息，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的“虚拟服务器管理”一节。

程序 APX

程序 APX 也称为脚本 APX，类似于 shell 命令，可作为 OGFS 服务器脚本执行。可以从 OGFS 命令行调用它们，并使用 STDIN 或命令行参数向其传递输入参数。它们的输出将传递到 STDOUT 和 STDERR 中。

程序 APX 在全局 Shell (OGSH) 会话内执行，并且可以访问对调用该 APX 的用户可用的所有 OGSH 功能。这些功能包括 rosh、CLI、OGFS 等等。您可以使用任何基于脚本的工具(如 shell 脚本、Python 和 Perl 等)编写程序 APX。

可以从 OGSH 命令提示符处调用程序 APX。通常情况下，将以同步方式执行程序 APX，这意味着直到程序 APX 返回时，shell 提示符才会返回。无论是在 twister 还是 OGFS 中，都不能将 APX 计划为重复作业。

程序 APX 位于 OGFS 目录 `/opsw/apx/bin` 中。

在交互式 OGSH 会话中，用户只能在/opsw/apx/bin 中查看其有权执行的程序 APX。如果用户尝试调用其无权执行的程序 APX，将导致从 shell 返回“文件未找到”错误。

程序 APX 也可以由其他 Web APX 或程序 APX 调用。例如，Web APX 中的 CGI 程序或 PHP 脚本可以调用程序 APX。

Web APX

Web APX 使用 CGI 程序或 PHP 脚本执行。这些 CGI 程序和 PHP 脚本在特定于用户的 OGSH 会话中执行。它们可以访问各种 SA 设施，例如 rosh、SA API、CLI 或可在 OGSH 会话中执行的任何命令。Web APX 由已启用 PHP 模块的内置 Apache Web 服务器提供服务。

可以通过两种方式访问 Web APX：使用诸如 Internet Explorer 或 Firefox 的独立 Web 浏览器，或从 SA 客户端。不支持 Microsoft ActiveX。

首次从独立 Web 浏览器调用 Web APX 时将触发一个登录对话框，其中要求对 SA 用户凭据执行验证。从 SA 客户端调用 Web APX 时不需要额外登录。Web APX 可以用于构建自定义客户应用程序的用户界面。

要在已启用“增强的安全配置”的 Windows Server 2003、2008 和 2012 上使用 Microsoft Internet Explorer 6 和 7 启动 APX，必须首先将 SA 客户端 URL 添加到 Internet Explorer 的可信站点列表中。

APX 用户角色

APX 用户有三种常规角色，如下表所示:

APX 用户角色

用户角色	描述
最终用户	运行 APX。该类用户通常无权修改 APX 或查看其内容。
APX 开发人员	创建和发布 APX。该类用户可以导入和导出 APX，并且可以修改 APX 内容。
APX 管理员	确定允许运行的 APX 用户。这些用户将通过管理文件夹权限来分配可执行权限以运行 APX。APX 管理员可能无权修改 APX 本身，但有权查看 APX 内容以确定要允许执行的 APX。

APX 权限

APX 需要您拥有 SA 客户端功能权限**管理扩展**。可以为用户组分配以下权限之一：

- 管理扩展：读取
- 管理扩展：读取和写入
- 管理扩展：无

APX 功能权限

这些功能权限只适用于 APX 开发人员和管理员，而不适用于只需要运行 APX 的用户。

- **读取**权限将授予显示 APX 源内容或导出(下载)APX 源存档的能力。
- **读取和写入**权限除了授予读取访问权限之外，还将授予修改 APX 内容的能力。
- **无**权限将拒绝对 APX 源的所有访问。

除了 SA 客户端功能权限**管理扩展**之外，必须使用文件夹权限(列出、读取、写入、执行)确定用户有权访问的 APX。

APX 权限

权限	描述
列出	列出系统的 APX 的权限。
读取	查看 APX 内容的权限。
写入	修改 APX 内容及导入和导出 APX 的权限。
执行	运行 APX 和查看 APX 属性的权限。

下表显示了如何根据“管理扩展”功能权限和文件夹权限的组合来确定权限的列表。

APX 权限列表

文件夹权限：	**管理扩展权限：**		
	读取	**读取和写入**	**无**
列出	列出 APX	列出 APX	列出 APX
读取	导出 APX	导出 APX	列出 APX
写入	导出 APX	导入和导出 APX	列出 APX
执行	运行 APX	运行 APX	运行 APX

与其他 SA 功能类似，您可以向用户授予访问 APX 的权限，以及指定用户可向其应用 APX 的托管服务器和/或策略。

如果用户尝试访问其无权执行的 Web APX，Web 浏览器会收到返回代码 HTTP 403 Forbidden。

有关 SA 权限的详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

权限升级

在执行 APX 时，用户仅有权访问已在 SA 中授权的资源和操作。然而在某些情况下，在执行 APX 时，有必要临时向用户授予升级的权限，即超出 SA 权限的权限。您可以在运行 APX 的同时，明确地向用户临时授予高于其默认 SA 权限的特定权限。权限升级对于运行 APX 的用户而言是透明的。

例如，您可能希望某个用户能够在托管服务器上运行 BIOS 信息收集应用程序，但该用户没有执行该操作的授权。您可以为没有 BIOS 信息收集应用程序运行权限的用户编写一个 APX，来向该用户临时授予所需的权限。在 APX 结束其运行后，该用户的权限将恢复为默认值。

权限升级可在 apx.perm 文件中指定。有关详细信息，请参见 APX 权限升级配置文件 - apx.perm (第 113 页)。

APX 结构

APX 具有以下特性：

- APX 类型：程序 APX(也称为脚本 APX)或 Web APX。
- APX 的唯一名称：这是 APX 必须唯一具有的完整名称。例如，com.hpe.sa.RestartMyApp。
- APX 显示名称：该名称通常比 APX 的唯一名称短。例如，RestartMyApp。
- APX 版本：可以通过设置版本字符串来维护多个 APX 版本，也可以让 SA 自动管理版本。APX 版本可以是一个简单数字，如版本 1、2、3 等，也可以是任何字母数字字符串。

有关详细信息，请参见将 APX 导入 SA - apxtool import (第 107 页)和设置 APX 的当前版本 - apxtool setCurrent (第 110 页)。

文件结构

对于 SA，APX 就是一组符合 APX 类型(程序 APX 或 Web APX)协定的文件和目录，允许 APX 运行时任务可以正确地执行它。例如，Web APX 可能需要 index.html 文件或 index.php 文件。程序 APX 可能需要使用与该 APX 具有相同名称的 shell 命令。

有关 APX 中的文件的详细信息，请参见APX 文件 (第 111 页)。

OGFS 集成

在 SA 文件系统中，APX 基础结构依赖于 OGFS 管理用户会话及显示 APX 的各个部分。下列各节描述了 APX 是如何集成到 OGFS 及其各应用程序中的。

APX 可执行文件目录

程序 APX 将在全局 Shell (OGSH) 中以可执行程序形式进行处理。这些 APX 在 OGSH 中作为可执行的命令显示。这使得 shell 用户能够像运行 shell 命令一样调用 APX。

APX 可执行文件目录具有以下格式：

```
/opsw/apx/bin/{apx_name}
```

其中 apx_name 是 APX 的名称。在 /opsw/apx/bin/{apx_name} 中运行 apx_name 将调用 apx_name 的当前版本。

APX 运行时目录

APX 运行时目录由 APX 运行时用来支持 APX 执行。APX 运行时目录必须具有 APX 源的访问权限。此外，拥有 APX 开发人员权限和读取权限的用户也可以访问 APX。对于全局 Shell 中的非 APX 开发人员，APX 运行时目录将不可用。

APX 运行时目录将引用当前 APX 版本的源。它具有以下格式：

```
/opsw/apx/runtime/{apx_type}/{apx_name}
```

其中 apx_type 可以是 script 或 web。

APX 接口 - 定义 APX 扩展的类别

APX 接口允许您创建命名的 APX 类别，以及查找给定类别的所有 APX。接口是类别的名称。例如，您可以创建一类 APX，全部接受特定的输入参数集，并生成特定类型的输出数据。也可以创建一类 APX，全部执行特定的操作集。

此外，还可以创建 APX 或外部应用程序，用于获取所需类别的所有 APX 的名称并执行这些 APX。或者，APX 或应用程序可以仅显示所需类别的 APX 的列表，而由用户选择要执行的 APX。

APX 接口是一个名称，用于定义 APX 调用者与 APX 之间的非正式协定。

- **定义接口名称**的 APX 将创建一类具有该名称的 APX。
- **实现接口**的 APX 会将其自身声明为属于该类别的 APX。

示例接口

SA 提供了一个名为 RightClickToRun 的接口。该接口定义了一类 APX，这类 APX 接受一个或多个设备作为输入参数，并针对这些设备运行。此外，SA 客户端将在“操作”>“运行扩展”菜单中显示所有实现此接口的 APX，以允许用户选择一个或多个设备，并对选定设备运行这些 APX。有关此接口的详细信息，请参见RightClickToRun 接口 (第 101 页)。

定义接口

APX 接口定义了一类 APX 的名称。所有实现该接口的 APX 将属于该类别，并且必须符合该接口的约定。要创建新类别，您可以让 APX“定义”接口。

要让 APX 定义接口，请执行以下步骤：

1. 使用 `apxtool new` 命令创建 APX。有关此命令的详细信息，请参见创建新 APX - apxtool new (第 104 页)。
2. 找到新 APX 的文件，并在文本编辑器中打开名为 `interfaces` 的文件。该接口文件位于 APX 目录的 APX-INF 目录中。
3. 在 `interfaces` 文件的末尾添加对应于以下项的三行：
   - 文件中的接口部分的名称。这是接口的唯一名称。
   - 接口的显示名称。
   - 接口描述。

   例如，下面显示了名为 "com.hpe.sa.MyNewInterface" 的接口的接口部分名称、显示名称和描述：

```
[com.hpe.sa.MyNewInterface]
name=MyNewInterface
description=”This is a simple interface for testing purposes.”
```

4. 保存更改并关闭文件。
5. 使用 apxtool import 命令将修改后的 APX 导入 SA。有关此命令的详细信息，请参见将 APX 导入 SA - apxtool import (第 107 页)。

要升级现有 APX 以定义接口，必须创建 interfaces 文件，并如上所述添加接口。

实现接口

APX 接口指定了一类须符合该接口约定的 APX。要将 APX 指定为属于某个类别，可以让 APX“实现”接口。要让 APX 实现接口，请执行以下步骤。

1. 使用 apxtool new 命令创建 APX。有关此命令的详细信息，请参见创建新 APX - apxtool new (第 104 页)。
2. 找到新 APX 的文件，并在文本编辑器中打开名为 apx.cfg 的文件。
3. 在 apx.cfg 文件中找到关于“实现”的部分。该部分简要介绍了如何指定 APX 所实现的接口。
4. 在 apx.cfg 文件中找到以下行：

   ```
   [Implementing]
   interfaces=
   ```

5. 修改 interfaces= 行，并在该行结尾处添加接口名称。例如，如果 APX 实现了名为 "com.hpe.sa.MyNewInterface" 的接口，则 apx.cfg 文件应包含以下行：

   ```
   [Implementing]
   interfaces=com.hpe.sa.MyNewInterface
   ```

   要实现多个接口，请将它们添加到接口行，各接口之间用冒号分隔，如下所示：

   ```
   [Implementing]
   interfaces=com.hpe.sa.MyNewInterface:com.hpe.sa.AnotherInterface
   ```

6. 保存更改并关闭 apx.cfg 文件。

7. 使用 apxtool import 命令将修改后的 APX 导入 SA。有关此命令的详细信息，请参见 将 APX 导入 SA - apxtool import (第 107 页)。

必须设置 APX 的当前版本，才能在通过 SA 客户端或 apxtool query 命令查看 APX 时看到已实现的接口。有关详细信息，请参见 设置 APX 的当前版本 - apxtool setCurrent (第 110 页)。

要升级现有 APX 以使用接口，必须按如上所述将接口添加到现有 apx.cfg 文件中。

RightClickToRun 接口

SA 提供了一个可在您的 APX 中使用的接口，名为 com.hpe.client.server.RightClickToRun。该接口仅适用于程序 APX，不适用于 Web APX。当希望 APX 执行以下所有操作时，请使用该接口：

- 接受一个或多个设备作为 APX 的输入参数。实现此接口的 APX 必须接受“-d device id”作为输入参数。
- 出现在“操作”>“运行扩展”>“选择扩展...”窗口中。
- 出现在 SA 客户端 的“操作”>“运行扩展”菜单中。当已使用“操作”>“运行扩展”>“选择扩展...”菜单运行一次 APX 之后，APX 将出现在该菜单中。

要从“操作”>“运行扩展”菜单中执行 APX，用户必须具有执行该 APX 的权限。用户无权执行的任何 APX 都不会出现在该菜单项中。有关权限的信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

RightClickToRun 接口允许用户在 SA 客户端中选择一个或多个设备，并针对这些设备运行 APX。

当选择“操作”>“运行扩展”菜单项时，SA 客户端显示实现 com.hpe.client.server.RightClickToRun 接口的所有程序 APX。当选择 APX 时，将针对所有选定服务器运行。将为每个选定的服务器调用一次 APX。

有关如何让 APX 实现此接口的说明，请参见实现接口 (第 100 页)。有关使用实现此接口的 APX 的详细信息，请参见 HPE SSO 门户上“Server Automation 用户指南”中的“运行 SA 扩展”一节。

CoreAffinity 接口

SA 提供了一个可在您的 APX 中使用的接口，名为 'com.hpe.client.server.CoreAffinity'。如果您要在 CoreAffinity 模式中运行 APX，则可使用此接口。

仅当您的网状网络至少拥有两个 SA 核心时，CoreAffinity 模式才适用。如果为每个目标服务器启用此模式，则无论实际作业在何处启动，APX 均会已注册此目标服务器的 SA 核心上执行。

例如：

- 您的网状网络具有两个核心：核心 A 和核心 B
- 您从两个目标服务器 MA (注册至核心 A) 和 MB (注册至核心 B) 上的核心 A 中启动 APX 作业

在 CoreAffinity 模式中，此作将为核心 A 上的 MA 和核心 B 上的 MB 运行 APX。如果 CoreAffinity 已禁用，则两项执行均会在核心 A 上完成 (这是因为作业是在此核心上启动的)。

有关如何让 APX 实现 CoreAffinity 接口的说明，请参见实现接口 (第 100 页)。

使用接口 API

您可以使用 SA API 将自己的应用程序与 SA 和 APX 集成。您的应用程序可以使用 SA API 中 com.opsware.apx 包内的 APXInterfaceService 接口，确定将实现该特定接口的所有 APX。有关使用 SA API 的信息，请参见API 文档和 Twister (第 35 页)。

apxtool 命令

可以在 OGFS 会话中使用 apxtool 命令创建和管理 APX。apxtool 命令可在全局 Shell 中使用，位于 `/opsw/bin/apxtool` 目录中。

有关如何使用 apxtool 创建 Web APX 的教程，请参见教程:创建 Web 应用程序 APX (第 116 页)。

apxtool 的语法

可在 OGFS 命令行中按照如下所示调用该 APX 工具：

```
apxtool [-h | --help] {function} arguments
```

要获取该 APX 工具支持的命令和参数的完整列表，请从 OGSH 命令行中运行不带参数的 apxtool。

APX 工具支持以下主要函数：

APX 工具函数

函数	用法
new	在 OGFS 中创建新的 APX 源目录和一组新模板文件。可以选择在 SA 中注册 APX。通过注册，会分配一个 APX ID，并使您的 APX 名称对使用 SA 的其他人(需具有相应权限)可用。有关详细信息，请参见创建新 APX - apxtool new (第 104 页)。
import	将 APX 文件导入 SA 库，并创建 APX 的新版本。可以选择在 SA 中注册 APX。通过注册，会分配一个 APX ID，并使您的 APX 名称对使用 SA 的其他人(需具有相应权限)可用。有关详细信息，请参见将 APX 导入 SA - apxtool import (第 107 页)。
setcurrent	设置 SA 库中 APX 的当前版本。SA 中可存在多个版本的 APX，但只能执行当前版本。有关详细信息，请参见设置 APX 的当前版本 - apxtool setCurrent (第 110 页)。
query	显示关于 APX 的信息。有关详细信息，请参见查询 APX 信息 - apxtool query (第 108 页)。
export	将所有 APX 文件从 SA 库复制到一组单独的文件。
delete	从 SA 库中删除 APX。

使用短命令选项和长命令选项

apxtool 命令的大多数选项接受短格式或长格式。

- 短格式包括一个连字符和一个字符，例如“-t”和“-v”。
- 长格式包括两个连字符后跟一个单词，例如“--type”和“--view”。

有些选项需要后跟一个参数。例如，“-t webapp”和“-t details”。可以按照四种格式之一指定参数，这些格式是等效的。为了进行说明，下列这些命令等效并会生成相同的结果：

```
apxtool query -t webapp
apxtool query -twebapp
apxtool query -tw
apxtool query --type webapp
apxtool query --type=webapp
```

有些选项只需要键入足以标识选项参数的最少字符数。例如，在查询函数中，--view 选项需要 "list"、"details"、"versions" 参数。下列命令将生成相同的结果：

```
apxtool query --view=details
apxtool query --view=d
apxtool query -vdetails
apxtool query -vd
```

创建新 APX - apxtool new

可以使用该 APX 工具创建新 APX，并选择在 SA 中注册该 APX 的名称。此命令将创建一组可供您修改的 APX 模板文件。有关组成 APX 的文件的信息，请参见APX 文件 (第 111 页)。

用法

```
apxtool new [options] {src_dir}
```

其中 src_dir 参数用于指定要在其中创建新 APX 的模板文件的目录。如果省略该参数，会将模板文件放置到当前目录中。

下表列出了可用于创建新 APX 的选项：

apxtool new 的选项

选项	用法
-h, --help	显示此帮助消息并退出。
-t <type> --type=<type>	(必需)APX 类型。有效值包括：script 或 webapp。例如，-ts 表示脚本 APX，-tw 表示 Web APX。(脚本 APX 也称为程序 APX)。
-u <unique name>	(必需)APX 的唯一名称。唯一名称是符合文件系统格式的点

apxtool new 的选项(续)

选项	用法
--uniquename=<unique name>	分隔名称。必须至少包含一个点。有效字符包括：[a-zA-Z0-9_.]。 **例如：** com.hpe.sa.security.scan_ports
-n <name> --name=<name>	(**可选**)APX 在文件夹中的显示名称。如果未指定名称，但是指定了唯一名称，则会将 APX 唯一名称的结尾部分用作显示名称。请注意，该名称必须在指定的文件夹中唯一。 例如，如果唯一名称为 com.hpe.sa.MyWebExt，则默认显示名称为 MyWebExt。
-d <description> --description=<description>	(**必需**)对 APX 的简短描述。如果描述是带有扩展名 .txt 文件名，则会假定该文件是一个文本文件，其内容将用作 APX 描述。
-r --register	(**可选**)向系统中注册 APX 的名称。如果指定此选项，则还必须指定 -f 或 --folder。 如果没有在 apxtool new 中指定 -r 和 -f，则必须在 apxtool import 中使用 -f。
-f <path> --folder=<path>	(**可选**)将在其中注册 APX 的 SA 文件夹路径。它可以是完整路径、部分路径、绝对路径或相对路径，只要它可以唯一标识特定文件夹即可。只有在使用了 -r 或 --register 时，才需要使用此选项。 如果没有在 apxtool new 中指定 -r 和 -f，则必须在 apxtool import 中使用 -f。
-Q, --quiet	(**可选**)禁止显示输出消息。
-F, --force	(**可选**)禁止显示确认提示。

删除 APX - apxtool delete

可以使用该 APX 工具从 SA 库中删除现有 APX。

用法

```
apxtool delete [options]
```

下表列出了可用于删除 APX 的选项:

apxtool delete 的选项

选项	用法
-h --help	显示此帮助消息并退出。
-t <type> --type=<type>	(必需)APX 类型。有效值包括：script 或 webapp。例如，-ts 表示脚本。
--id=<APX id>	(可选)所需 APX 的对象标识符。
-u <unique_name> --uniquename=<unique_name>	(可选)APX 的唯一名称。唯一名称是符合文件系统格式的点分隔名称。必须至少包含一个点。有效字符包括：[a-zA-Z0-9_.]。 例如： com.hpe.sa.security.scan_ports
-n <name>, --name=<name>	(可选)APX 在文件夹中的显示名称。
-f <path>, --folder=<path>	(可选)SA 文件夹路径。Path 可以是完整路径、部分路径、绝对路径或相对路径，只要它可以唯一标识特定文件夹即可。
-Q, --quiet	(可选)禁止显示输出消息。
-F, --force	(可选)禁止显示确认提示。

从 SA 导出 APX - apxtool export

可以使用该 APX 工具导出 APX。导出操作会下载 APX 源存档文件的特定版本，并将这些文件放到一个目录或 .zip 存档文件中。

用法

```
apxtool export [options] {target_dir}
```

其中参数 target_dir 是要将 APX 源存档文件复制到的目录，或要将 APX 源存档内容扩展到的目录，具体取决于是否指定了 --archive 选项。如果省略该参数，将使用当前目录。

下表列出了可用于导出 APX 的选项。

apxtool export 的选项

选项	用法
-h, --help	显示此帮助消息并退出。
-t <type>, --type=<type>	(必需)APX 类型。有效值包括：script 或 webapp。例如，-ts 表示脚本。
--id=<APX id>	(可选)所需 APX 的对象标识符。
-u <unique_name>, --uniquename=<unique_name>	(可选)APX 的唯一名称。唯一名称是符合文件系统格式的点分隔名称。必须至少包含一个点。有效字符包括：[a-zA-Z0-9_.]。 例如： com.hpe.sa.security.scan_ports
-n <name>, --name=<name>	(可选)APX 在文件夹中的显示名称。
-f <path>, --folder=<path>	(可选)SA 文件夹路径。Path 可以是完整路径、部分路径、绝对路径或相对路径，只要它可以唯一标识特定文件夹即可。
-v v<ersion_string>, --version=<version_string>	(可选)此选项指定要下载的 APX 版本。如果省略该参数，将下载当前版本。
-a, --archive	如果已指定该选项，会将 APX 原始源存档中的 APX 源以 ZIP 或 JAR 文件形式导出。
-Q, --quiet	(可选)禁止显示输出消息。
-F, --force	(可选)禁止显示确认提示。

将 APX 导入 SA - apxtool import

可以使用该 APX 工具导入 APX。导入操作将发布 APX 的新版本，并且可以选择将该版本设置为当前版本。如果 APX 尚未注册，则该命令还会注册 APX。

只能运行 APX 的当前版本。如果不设置当前版本，APX 将不可运行。可以使用 apxtool import 或 apxtool setcurrent 设置当前版本。有关详细信息，请参见设置 APX 的当前版本 - apxtool setCurrent (第 110 页)。

用法

```
apxtool import [options] {apx_src}
```

其中 apx_src 可以是扩展名为 .zip 或 .jar 的存档 APX 源文件，也可以是要发布的 APX 文件所在目录的名称。apx_src 可以是相对或绝对路径。如果省略该参数，将使用当前目录。所指定的目录或存档文件必须包含目录 APX-INF。

下表列出了可在导入 APX 时使用的选项:

apxtool import 的选项

选项	用法
-h, --help	显示此帮助消息并退出。
-c, --setcurrent	如果已指定该选项，会将新发布的版本设置为 APX 的当前版本。
--version=<version_string>	该 APX 的新版本。如果已经在 apx.cfg 中指定了 version_string，则不能使用此选项。如果未指定版本，系统会自动分配一个版本。
-f <path>, --folder=<path>	(可选)SA 文件夹路径。Path 可以是完整路径、部分路径、绝对路径或相对路径，只要它可以唯一标识特定文件夹即可。 如果没有在 apxtool new 中指定 -r 和 -f，则必须在 apxtool import 中使用 -r。
-Q, --quiet	(可选)禁止显示输出消息。
-F, --force	(可选)禁止显示确认提示。

查询 APX 信息 - apxtool query

可以使用该 APX 工具获取和查看 APX 信息。可以指定其他选项来限制所得到的 APX。多个相同选项将构成逻辑 OR 表达式。如果没有找到匹配的结果，该命令将返回退出代码 100。

用法

```
apxtool query [options]
```

下表列出了可在查询 APX 信息时使用的选项:

apxtool query 的选项

选项	用法
-h, --help	显示此帮助消息并退出。
-v <view>, --view=<view>	(可选)选择查询结果的预定义视图之一。选项包括 list(默认值)、details 和 versions。 -v list 是 APX 基本信息的单行显示，以表格形式呈现。 -v details 是 APX 信息的多行显示。 -v versions 将列出所有 APX 版本。您仅需为视图类型指定足够的字符；例如，-vd 与 -v details 是相同的。如果选择 versions 布局，则查询必须得到单个 APX 对象。
-t <type>, --type=<type>	(可选)指定要显示的 APX 类型。有效值包括: script 或 webapp 或 interface。默认情况下将显示所有类型。 -t script 将显示所有脚本 APX。 -t webapp 将显示所有 Web APX。 -t interface 将显示已定义一个或多个接口的所有 APX。 例如，apxtool query -ts 将显示所有脚本 APX。
--id=<APX id>	(可选)所需 APX 的对象标识符。
-u <unique_name> --uniquename=<unique_name>	(可选)APX 的唯一名称。唯一名称是符合文件系统格式的点分隔名称。必须至少包含一个点。有效字符包括：[a-zA-Z0-9_.]。 **例如：** com.hpe.sa.security.scan_ports
-n <name>, --name=<name>	(可选)APX 在文件夹中的显示名称。
-f <path>, --folder=<path>	(可选)SA 文件夹路径。Path 可以是完整路径、部分路径、绝对路径或相对路径，只要它可以唯一标识特定文件夹即可。
--current	(可选)如果已指定该选项，则仅查询已设置当前版本的 APX 对象。
--format=<format_string>	(可选)此高级选项允许您指定 APX 列表的自定义显示格式。 format_string 是包含嵌入式标记名称的字符串，在显示时这些嵌入式标记名称将替换为相应的值。标记名称的格式必须是 %(tag_name)。 可使用格式字符串“__show_tags__”显示所有支持的标记名称的列表。

apxtool query 的选项(续)

选项	用法
`--csv`	(可选)以逗号分隔值格式显示输出。如果已指定 `--format` 选项，则将忽略该选项。
`-Q, --quiet`	(可选)禁止显示无关的输出消息。

设置 APX 的当前版本 - apxtool setCurrent

可以使用该 APX 工具将一个 APX 版本设置为当前版本。

只能运行 APX 的当前版本。如果不设置当前版本，APX 将不可运行。可以使用 `apxtool import` 或 `apxtool setcurrent` 设置当前版本。有关详细信息，请参见将 APX 导入 SA - apxtool import (第 107 页)。

用法

```
apxtool setcurrent [options] {version_str}
```

其中 version_str 参数为必需，用于唯一标识 APX 的现有版本。

下表列出了可在设置 APX 版本时使用的选项:

apxtool setcurrent 的选项

选项	用法
`-h, --help`	显示此帮助消息并退出。
`-t <type>, --type=<type>`	(必需)APX 类型。有效值包括：`script`, `webapp`.例如，-ts 表示脚本。
`--id=<APX id>`	(可选)所需 APX 的对象标识符。
`-u <unique_name>, --uniquename=<unique_name>`	(可选)APX 唯一名称。唯一名称是符合文件系统格式的点分隔名称，必须至少包含一个点。有效字符包括: [a-zA-Z0-9_.]。 **例如：** `com.hpe.sa.security.scan_ports`
`-n <name>, --name=<name>`	(可选)APX 在文件夹中的显示名称。

apxtool setcurrent 的选项(续)

选项	用法
-f <path>, --folder=<path>	(可选)SA 文件夹路径。Path 可以是完整路径、部分路径、绝对路径或相对路径，只要它可以唯一标识特定文件夹即可。
-Q, --quiet	(可选)禁止显示输出消息。
-F, --force	(可选)禁止显示确认提示。

错误处理

APX 工具命令符合标准 POSIX 约定。在成功时返回 0，在发生其他错误时返回非零值。该 APX 工具会将正常输出发送到 STDOUT，将错误和警告发送到 STDERR。当发生错误时，该 APX 工具通常会向 STDERR 返回一条描述性消息。

下表中显示了典型的错误情况分类:

APX 工具错误情况

返回代码	描述
0	成功
1	语法或用法错误
2	权限相关错误
3	用户取消操作
4	运行时错误

可能还存在其他一些未记录的退出代码。唯一可确保的是，如果退出代码为 0，则说明命令已成功完成其操作。

APX 文件

本节描述了在运行 apxtool new 命令时创建的模板文件。下表概述了这些文件。后面的各节详细描述了其中一些文件。

APX 文件

文件名	描述
apx.cfg	APX 配置文件，其中包含用于完整描述 APX 的元数据。有关详细信息，请参见APX 配置文件 - apx.cfg (第 112 页)。
apx.perm	APX 权限文件，用于指定权限升级规则。有关详细信息，请参见APX 权限升级配置文件 - apx.perm (第 113 页)。
description.txt	对 APX 的文本描述。在 apxtool new -d 选项中指定。有关详细信息，请参见创建新 APX - apxtool new (第 104 页)。
interfaces	APX 接口定义文件。指定 APX 定义或实现的接口。有关详细信息，请参见APX 接口 - 定义 APX 扩展的类别 (第 99 页)。
usage.txt	对 APX 用法的文本描述。
run.sh	仅适用于程序 APX。该文件包含 APX 的可执行代码。该文件包含程序 APX 的功能。有关示例，请参见教程:创建程序 APX (第 122 页)，了解详细信息。
index.php	仅适用于 Web APX。该文件包含 Web APX 的 PHP 源代码。该文件包含 Web APX 的功能。有关示例，请参见教程:创建 Web 应用程序 APX (第 116 页)，了解详细信息。

APX 配置文件 - apx.cfg

无论类型如何，所有 APX 都必须具有配置文件 apx.cfg。apxtool new 命令将创建该文件的模板，供您修改。该文件包含用于完整描述 APX 的元数据。apx.cfg 使用 "key=value" 格式定义 APX 的属性。将使用行继续符 "\" 连接多行。

APX 配置文件特性 (第 112 页)描述了所有 APX 的通用特性。特定于 APX 类型的特性将在对应 APX 类型的功能规范中描述。请注意，某些特性可从 apx.cfg 配置文件中提取，并在 SA 中进行管理。对于可修改的特性(例如描述)，后续更新 apx.cfg 文件时将相应地更新 SA 管理的数据。

要查看 apx.cfg 文件的示例，请运行 apxtool new 命令，并打开该命令创建的文件。

APX 配置文件特性

特性	是否可修改？	描述
type	否	APX 的类型。必须是 webapp 或 script。(脚本 APX 也称为程序 APX)。APX 类型一旦创建，即不能更改。
name	是	这是 APX 显示名称，可包含多字节字符。可以随时更改该名称。该名称将列在 SA 客户端 APX 文件夹

APX 配置文件特性(续)

特性	是否可修改？	描述
		中。
unique_name	否	APX 的唯一名称。该名称将用作显示在 OGFS 中的 APX 文件名。该名称与类型一起构成唯一标识 APX 的键。该名称一旦创建，即不能更改。由于该名称将在文件系统中使用，所以它必须符合文件系统命名规范。通常情况下，该名称应该是 ASCII。
version	是	表示 APX 当前版本的版本字符串。如果值以字符串“auto:”开头，则 SA 将使用针对每个新版本递增的整数自动管理版本。
description	是	对 APX 所执行操作的文本描述。也可以使用 description.txt 文件来代替该特性。
usage	是	对 APX 用法的文本描述。也可以使用 usage.txt 文件来代替该特性。
interfaces	是	APX 实现的一个或多个接口。使用冒号 (:) 字符分隔多个接口。
command	是	在调用时将由 APX 运行的可执行文件。

APX 权限升级配置文件 - apx.perm

可使用 apx.perm 文件指定权限升级规则。如果该文件不存在，或者不包含升级权限，将使用用户的默认权限运行 APX。

当使用 APX 工具的 **New** 命令创建新 APX 时，它会生成特定默认文件，包括默认的 apx.perm 文件，该文件默认情况下未定义升级权限。该默认文件包含一些注释掉的示例，APX 开发人员可以将这些示例用作模板。

可使用如下所述的三种方法指定升级。

- 未升级 (第 114 页)
- 所有权限 (第 114 页)
- 执行升级 (第 114 页)

未升级

未指定升级特性。APX 运行时将使用当前用户权限执行 APX。如果 APX 调用了用户无权执行的特权操作，APX 将终止执行并显示错误。

所有权限

这是一个特殊权限，用于向用户临时授予所有操作权限。该权限仅用于开发或演示用途。对于需快速证明概念或执行演示的情况，这是非常有用的工具，因为用户无需关注细粒度的权限调整。而对于生产环境，这并不是一个好的选择，因为其安全性不高。

要授予所有权限，请通过使用通配符匹配所有功能的宏来编辑 apx.perm 文件。例如：

```
use_feature(name="*")
```

执行升级

在 apx.perm 文件中指定一组预定义的常用操作。执行 APX 时，APX 运行时临时将这些权限授予 APX。SA 具有功能和资源权限的全面列表。为了简化用于升级相关功能的任务，用户可以使用通配符来匹配相关功能组。例如：

```
@use_feature(name="Application.*")
```

显示 APX 的进度

可以在程序 APX 中使用 apxprogress 命令提供 APX 进度信息。对于运行很长时间的程序 APX，如果您需要向用户提供该 APX 的进度状态，则这很有用。

可以使用 Web APX 作为程序 APX 的前端，并显示 Web APX 中的进度。

apxprogress 命令

可使用 apxprogress 命令定义程序 APX 执行过程中的步骤数，以及记录每个步骤的完成时间。这使得用户能够了解 APX 的进展情况。

apxprogress 的语法

apxprogress {option}...。

apxprogress 命令的选项

选项	描述
-i <total number of steps>	指定 APX 运行的步骤总数。可以在 APX 开始时使用该选项指定 APX 将运行的步骤总数。 可以在 APX 中多次使用该选项以增加步骤数。每次使用之后，都会按指定的值递增步骤总数。
-c <current step>	指定当前步骤编号。在 APX 代码中的每个步骤完成后，可使用该选项调用 apxprogress。
-m <message>	指定用于描述 APX 状态的文本消息。
-a <data>	指定 APX 可提供的关于其自身的其他信息。
-d	表示调试模式。显示命令向 stdout 输出的内容，以供调试。
-h	显示关于 apxprogress 命令的帮助信息。

使用 apxprogress 的 Shell 脚本示例

以下 shell 脚本是使用 apxprogress 命令的程序 APX 的一部分。该 APX 总共定义了 100 个步骤，并将声明其当前进度 100 次。每次声明进度时，它还会提供一条包括步骤编号的消息。

```
#!/bin/sh
################################################################################
# A simple shell script for a program APX that displays progress
# about itself.
#Author:<name>
```

```
#############################################################################
echo "This is a simple APX that uses apxprogress."
totalsteps=100
apxprogress -i $totalsteps -c 1
for i in `seq $totalsteps`; do
apxprogress -c $i -m "APX is running, working on step $i" -d
sleep 10
done
```

查看 APX 进度

可以使用 SA API 方法 JobService.getProgress() 来访问已调用 apxprogress 命令、正在运行的 APX 的进度信息。有关该方法的示例，请参见教程:创建程序 APX (第 122 页)中的在 Twister 界面中查看 APX 进度 (第 128 页)。

教程:创建 Web 应用程序 APX

本教程演示了如何创建、发布和运行名为 mywebapp 的简单 Web 应用程序 APX。

运行本教程中所创建的 APX 默认版本将显示 PHP 命令 phpinfo 的输出。本教程后面将告诉您如何修改 PHP 代码，以便显示托管服务器的列表。因为本教程提供了源代码，所以无需预先掌握 PHP 知识。

按照顺序完成以下任务。

1. 设置权限并创建教程文件夹 (第 117 页)
2. 创建新 Web 应用程序 (第 118 页)
3. 向 SA 导入新 Web 应用程序 (第 119 页)
4. 运行新的 Web 应用程序 (第 120 页)
5. 修改 Web 应用程序 (第 121 页)
6. 运行修改后的 Web 应用程序 (第 122 页)

教程先决条件

要完成本教程，您必须具有以下能力和环境：

- 可作为 admin 或其他**超级管理员**组成员登录 SA。以 admin 身份登录将允许您设置权限。
- 可作为属于**高级用户**组的用户登录 SA。
- 高级用户有权创建和运行 Web 应用程序。在本教程所显示的示例命令中，该用户的名称是 jdoe。
- 了解如何在 SA 客户端中设置客户端功能权限。
- 有关权限的详细信息，请参见《SA 10.50 管理指南》中的“用户和组设置”一节。
- 了解如何在 SA 客户端中创建文件夹。
- 有关文件夹的详细信息，请参见《SA 10.50 用户指南》。
- 了解如何打开全局 Shell 会话。
- 了解 ls 和 cd 等基本 UNIX 命令。
- 具有开发在 HTTP 服务器上运行的 Web 应用程序的经验。

设置权限并创建教程文件夹

1. 以**高级用户**组成员身份登录 SA 客户端，并在 SA 库中创建以下文件夹：

   ```
   /Dev/MyApp
   ```

   在本教程的后面，您将向 MyApp 文件夹上载 Web 应用程序。在非教程环境中，此文件夹的名称可以是任意名称。可以创建或选择任何其他文件夹来包含 Web 应用程序。
2. 退出 SA 客户端。
3. 以 admin 身份登录 SA 客户端，并打开 MyApp 文件夹的“文件夹属性”。
4. 在“文件夹属性”的“权限”选项卡中，确保**高级**用户组具有下列权限：
   - 列出文件夹内容
   - 在文件夹中读取对象
   - 在文件夹中写入对象
   - 在文件夹中执行对象
5. 退出 SA 客户端。

创建新 Web 应用程序

1. 以属于**高级用户**组的 SA 用户身份打开全局 Shell 会话。
2. 在核心的 OGFS 主目录中，创建一个名为 mywebapp 的目录，然后更改该目录：

   ```
   $ mkdir mywebapp
   $ cd mywebapp
   ```

   Web 应用程序文件将存储在 mywebapp 目录中。
3. 使用 apxtool new 命令创建 Web 应用程序的目录结构和默认文件，如下所示。

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ ls
   $
   $ apxtool new -tw -d "This is my first app."\
   -u com.hpe.sa.jdoe.mywebapp
   Create source directory /home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp?Y/N y
   Info:Successfully created APX 'mywebapp' source directory:/home/jdoe/mywebapp.
   ```

   -tw 选项表示 APX 类型是 Web 应用程序，-d 指定描述信息，-u 指定应用程序的唯一名称。

   有关 apxtool new 命令选项的详细信息，请参见联机帮助：

   ```
   $ apxtool new -h
   ```
4. 将目录更改为 apxtool new 命令所创建的新目录，并列出目录中的文件。

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ cd com.hpe.sa.jdoe.mywebapp
   $ ls
   APX-INF cgi-bin css images index.php
   $ ls -R
   .:
   APX-INF cgi-bin css images index.php
   ./APX-INF:
   ```

```
apx.cfg apx.perm description.txt interfaces usage.txt
./cgi-bin:
./css:
hp_sa.css
./images:
```

5. 显示默认 index.php 文件的内容：

```
$ cat index.php
<?php
// Show information about PHP
phpinfo();
?>
```

与其他 Web 应用程序一样，您可以用 index.html 文件替换 index.php 文件。不过该本教程中使用的是 index.php 文件，您将在下一节中对其进行修改。

6. 检查 APX-INF 目录中的一些文件。有关详细信息，请参见 APX 文件 (第 111 页)。

APX-INF 目录包含特定于 APX Web 应用程序的信息。如以下 cat 命令所示，description.txt 文件中保存了使用 apxtool new 的 -d 选项指定的文本。

```
$ ls APX-INF/
description.txt apx.cfg apx.perm usage.txt
$ cat APX-INF/description.txt
This is my first app $
```

以下 grep 命令显示了 APX 配置文件 apx.cfg 中的一些属性。type 和 uniquename 的值来自于 apxtool new 命令的 -t 和 -u 选项。有关 APX 配置文件的详细信息，请参见APX 配置文件 - apx.cfg (第 112 页)。

```
$ grep "=" APX-INF/apx.cfg
type=webapp
name=mywebapp
unique_name=com.hpe.sa.jdoe.mywebapp
```

向 SA 导入新 Web 应用程序

导入 Web 应用程序时将执行以下操作：

- 在 SA 中的 HTTP 服务器上安装 Web 应用程序。
- 将 Web 应用程序复制到在 SA 库和全局 Shell 中显示的文件夹。
- 为 Web 应用程序分配版本号。

输入 apxtool import 命令并使用 y 来响应提示消息，如下所示。-f 选项指定 SA 库中将用于存储 Web 应用程序的文件夹。-c 选项设置 Web 应用程序的当前版本。

```
$ pwd
/home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp
$
$ apxtool import -f "/Dev/MyApp" -c
APX source is not specified.
Do you want to publish current directory:/home/jdoe/mywebapp/
com.hpe.sa.jdoe.mywebapp?Y/N y
APX with unique name 'com.hpe.sa.jdoe.mywebapp' does not exist.
Register it into the system?Y/N y
Info:Successfully registered APX 'mywebapp' (310001) in folder ‘/Dev/
MyApp’.
Info:Successfully published a new version '1' for APX 'mywebapp'.
Info:Successfully set APX 'mywebapp'(310001) current version as '1'.
```

运行新的 Web 应用程序

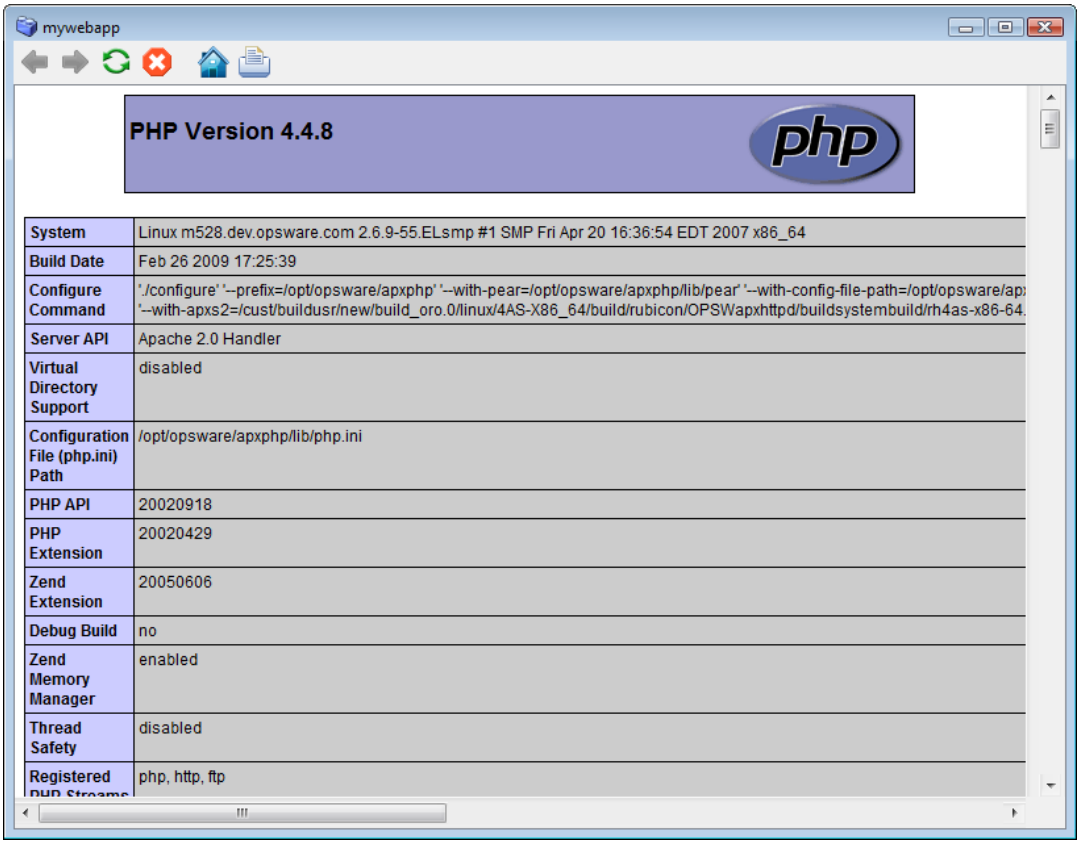
现在，您已发布了 Web 应用程序，可以像最终用户那样在 SA 客户端中运行该 Web 应用程序。

1. 以属于**高级用户**组的用户身份登录 SA 客户端。
2. 选择“库”选项卡，然后选择“按类型”选项卡。
3. 导航到“扩展”>“Web”节点，您将在此处看到 mywebapp 扩展。

   如果未看到 mywebapp，请确保您拥有设置权限并创建教程文件夹 (第 117 页)中所述的必需权限。
4. 要运行 Web 应用程序，请选择 mywebapp，并选择“操作”>“运行”菜单。

   此时将显示下图。Web 应用程序将显示由 index.php 文件的 phpinfo 语句所生成的信息。

Web 应用程序版本 1



修改 Web 应用程序

运行默认 index.php 文件是用来检查开发环境的一个好方法，但是它无法充分利用 SA 功能。在本节中，将修改 index.php 文件，使其列出 SA 管理的服务器的名称。

1. 在全局 Shell 会话中，找到 Web 应用程序的 index.php 文件。

   ```
   $ cd /home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp
   $ ls
   APX-INF cgi-bin css images index.php
   ```

2. 在文本编辑器(如 vi)中打开 index.php 文件。
3. 使用以下几行替换 index.php 的内容：

   ```
   <html>
   <head>
   ```

```
<title>Servers</title>
</head>
<body>
<p>List of servers:</p>
<?php
passthru("ls /opsw/Server/@");
?>
</body>
</html>
```

上面的 passthru 语句将运行 ls 命令，并将 stdout(无 reinflate)传回网页。ls 命令将列出托管服务器在 OGFS 中显示的名称。

4. 保存 index.php 文件，并退出文本编辑器。
5. 发布修改后的 Web 应用程序。

   以下 apxtool import 命令将当前版本设置为 2。-F 选项抑制确认提示。

```
$ apxtool import -f "/home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp" \
-c --version=2 -F
Info:Successfully published a new version '2' for APX 'mywebapp'
Info:Successfully set APX 'mywebapp'(310001) current version as '2'.
```

运行修改后的 Web 应用程序

1. 在 SA 客户端中，使用“视图”>“刷新”菜单刷新 Web 扩展视图，此时该视图应包含 mywebapp 的版本 2。
2. 选择“mywebapp”，然后选择“操作”>“运行”菜单。输出内容将类似于 Web 应用程序版本 1，不同之处是它将显示 PHP passthru 语句和 OGSH ls 语句的输出，其中列出了所有托管服务器。请注意，passthru 语句将删除用于分隔 ls 命令所返回的服务器名称的换行符。

教程:创建程序 APX

本教程演示了如何创建、发布和运行名为 myshellapp 并运行简单 shell 脚本的简单程序 APX。本教程后面将告诉您如何修改 shell 脚本来调用 apxprogress 命令以及提供进度信

息。因为本教程提供了源代码，所以无需预先掌握 shell 编程知识。

按照顺序完成以下任务。

- 设置权限并创建教程文件夹 (第 124 页)
- 创建新程序 APX (第 124 页)
- 向 SA 导入新 APX (第 126 页)
- 运行新 APX (第 126 页)
- 修改 APX (第 127 页)
- 运行修改后的 APX (第 128 页)
- 在 Twister 界面中查看 APX 进度 (第 128 页)

教程先决条件

要完成本教程，您必须具有以下能力和环境：

- 可作为 admin 或其他**超级管理员**组成员登录 SA。以 admin 身份登录将允许您设置权限。
- 可作为属于**高级用户**组的用户登录 SA。
- 高级用户有权创建和运行 Web 应用程序。在本教程所显示的示例命令中，该用户的名称是 jdoe。
- 了解如何在 SA 客户端中设置客户端功能权限。
- 有关权限的详细信息，请参见《SA 10.50 管理指南》中的“用户和组设置”一节。
- 了解如何在 SA 客户端中创建文件夹。
- 有关文件夹的详细信息，请参见《SA 10.50 用户指南》。
- 了解如何打开全局 Shell (OGSH) 会话和使用全局 Shell。
- 了解 ls 和 cd 等基本 UNIX 命令。

设置权限并创建教程文件夹

1. 以 admin 身份登录 SA 客户端，并打开 MyApp 文件夹的“文件夹属性”。
2. 在“文件夹属性”的“权限”选项卡中，确保**高级**用户组具有下列权限：
   - 列出文件夹内容
   - 在文件夹中读取对象
   - 在文件夹中写入对象
   - 在文件夹中执行对象
3. 退出 SA 客户端。

创建新程序 APX

1. 以属于**高级用户**组的 SA 用户身份打开全局 Shell 会话。
2. 在核心的 OGFS 主目录中，创建一个名为 myshellapp 的目录，然后更改该目录：

   ```
   $ mkdir myshellapp
   $ cd myshellapp
   ```

   程序 APX 文件将存储在 myshellapp 目录中。
3. 使用 apxtool new 命令创建程序 APX 的目录结构和默认文件，如下所示。

   ```
   $ pwd
   /home/jdoe/myshellapp
   $ ls
   $
   $ apxtool new -ts -d "This is my first program APX."\
   -u com.hpe.sa.jdoe.myshellapp

   Create source directory under
   ‘/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp’ for APX ‘myshellapp’?Y/N y
   Info:Successfully created source directory
   ‘/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp for APX ‘myshellapp’.
   ```

-ts 选项表示 APX 类型是程序 APX(也称为脚本 APX)，-d 指定描述信息，-u 指定应用程序的唯一名称。

有关 apxtool new 命令选项的详细信息，请参见联机帮助：

```
$ apxtool new -h
```

4. 列出由 apxtool new 命令创建的文件：

```
$ pwd
/home/jdoe/mywebapp
$ ls
com.hpe.sa.jdoe.myshellapp
$ cd com.hpe.sa.jdoe.myshellapp
$ pwd
/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$ ls -R
.:
APX-INF run.sh
./APX-INF:
apx.cfg apx.perm description.txt interfaces usage.txt
```

5. 显示默认 run.sh 文件的内容：

```
$ cat run.sh
#!/bin/sh

##############################################################################
# APX myshellapp
#
# Created by: jdoe
#
##############################################################################
echo "This is APX myshellapp"
```

6. 检查 APX-INF 目录中的一些文件。有关这些文件的详细信息，请参见APX 文件 (第 111 页)。

APX-INF 目录包含特定于 APX 的信息。如以下 cat 命令所示，description.txt 文件中保存了使用 apxtool new 的 -d 选项指定的文本。

```
$ ls APX-INF/
apx.cfg apx.perm  description.txt  interfaces  usage.txt
$ cat APX-INF/description.txt
```

```
This is my first program APX.$
```

以下 grep 命令显示了 APX 配置文件 apx.cfg 中的一些属性。type 和 uniquename 的值来自于 apxtool new 命令的 -t 和 -u 选项。有关 APX 配置文件的详细信息，请参见APX 配置文件 - apx.cfg (第 112 页)。

```
$ grep "=" APX-INF/apx.cfg
type=script
name=myshellapp
unique_name=com.hpe.sa.jdoe.myshellapp
command=run.sh
```

向 SA 导入新 APX

导入 APX 时将执行以下操作：

- 将 APX 复制到在 SA 库中显示的文件夹。
- 为 APX 分配版本号。

输入 apxtool import 命令并使用 y 来响应提示消息，如下所示。-f 选项指定 SA 库中将用于存储 Web 应用程序的文件夹。-c 选项设置 Web 应用程序的当前版本。

```
$ pwd
/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$
$ apxtool import -f "/Dev/MyApp" -c
APX source is not specified.
Do you want to publish current directory:/home/jdoe/myshellapp/
com.hpe.sa.jdoe.myshellapp?Y/N y
APX with unique name 'com.hpe.sa.jdoe.myshellapp' does not exist.
Register it into the system?Y/N y
Info:Successfully registered APX 'myshellapp' (20001).
Info:Successfully published a new version '1' for APX 'myshellapp'
Info:Successfully set APX 'myshellapp'(20001) current version as '1'.
```

现在，您已发布了 APX，可以像 SA 用户那样在 SA 客户端中运行它。

运行新 APX

现在，您已发布了 APX，可以在 SA 客户端中运行它。

1. 以属于**高级用户**组的用户身份登录 SA 客户端。
2. 在导航窗格中，选择“库”选项卡，然后选择“按类型”选项卡。
3. 打开“扩展”节点，并选择“程序”节点。此时将显示 SA 库中的所有程序 APX。您将在此处看到您的 APX。如果未看到 myshellapp，请确保您拥有设置权限并创建教程文件夹 (第 124 页)中所述的必需权限。
4. 选择 APX。
5. 选择“操作”**>**“运行”菜单项。此时将显示“运行程序扩展”向导。
6. 选择“下一步”按钮。
7. 选择“启动作业”按钮。
8. 当您的 APX 完成运行后，选择状态指示器以显示详细信息。
9. 选择“关闭”按钮。

修改 APX

在本节中，您将修改 run.sh 文件，并添加对 apxprogress 命令的调用以提供进度信息。

1. 在全局 Shell 会话中，找到 APX 的 run.sh 文件。

```
$ cd /home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$ ls
APX-INF run.sh
```

2. 在文本编辑器(如 vi)中打开 run.sh 文件。
3. 使用以下几行替换 run.sh 的内容：

```
echo "This is a simple APX that uses apxprogress."


totalsteps=100
apxprogress -i $totalsteps -c 1
for i in `seq $totalsteps`; do
        apxprogress -c $i -m "myshellapx is running, working on step $i" #-d
        sleep 10
done
```

这些 apxprogress 命令指定该 APX 包含 100 个步骤，并且调用 apxprogress 100 次(对每个步骤调用一次，两次调用之间等待 10 秒)。有关详细信息，请参见 显示 APX 的进度 (第 114 页)。

如果要进行调试，可以将“#-d”更改为“-d”，并手动运行 Shell 脚本以显示 apxprogress 命令在 stdout 上的输出。

4. 保存 run.sh 文件，并退出文本编辑器。
5. 发布修改后的 APX。

   以下 apxtool import 命令将加载 APX 的新版本，并将当前版本设置为 2。-F 选项抑制确认提示。

```
$ apxtool import -f "/home/jdoe/myshellapp" \
-c --version=2 -F
Info:Successfully published a new version '2' for APX 'myshellapp'
Info:Successfully set APX 'myshellapp'(20001) current version as '2'.
```

运行修改后的 APX

现在，您已经修改并重新发布了 APX，可像以前一样从 SA 客户端中运行它。

1. 在 SA 客户端中，使用“视图”>“刷新”菜单刷新程序扩展视图，此时该视图应显示 myshellapp 的版本 2。
2. 选择 APX。
3. 选择“操作”>“运行”菜单项。此时将显示“运行程序扩展”向导。
4. 选择“下一步”按钮。
5. 选择“启动作业”按钮。

在 Twister 界面中查看 APX 进度

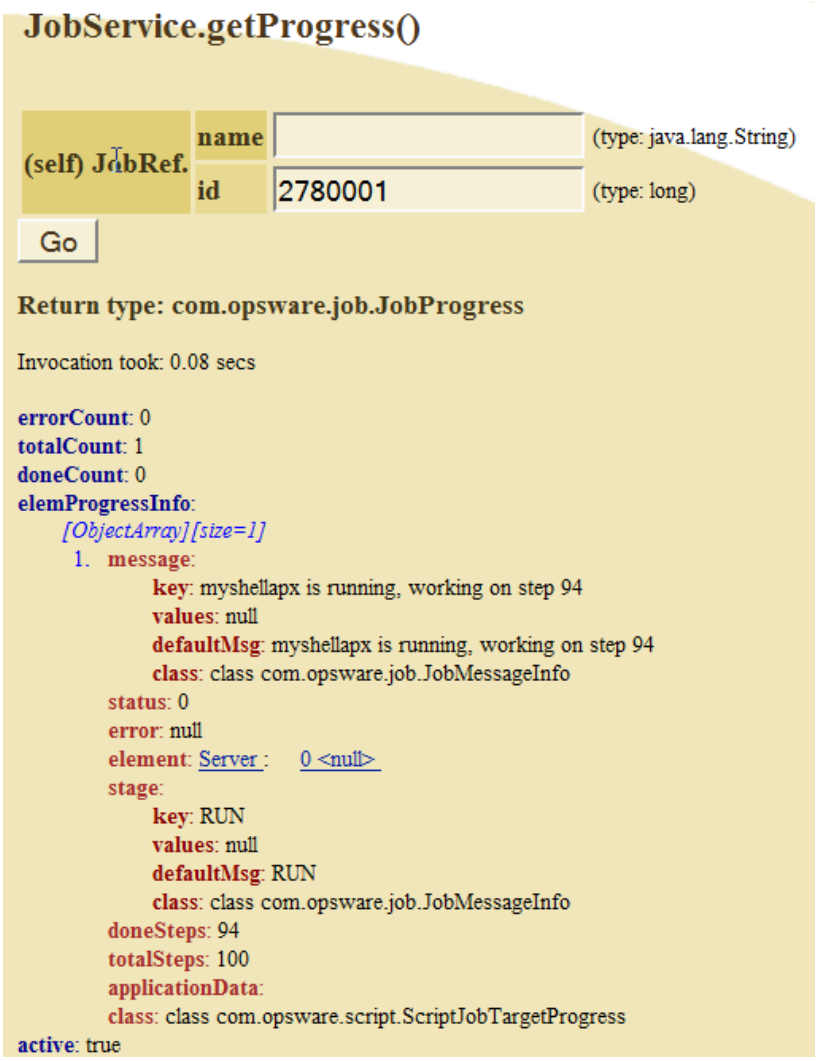
apxprogress 命令将报告正在运行的 APX 的进度。您可以通过调用 API 方法 JobService.getProgress() 来获取此进度信息。本节说明如何从 Twister 界面中运行该方法。有关 SA API 的 Twister 界面的详细信息，请参见API 文档和 Twister (第 35 页)。

1. 在 SA 客户端中，选择“作业和会话”选项卡。
2. 在作业列表中找到您的 APX。
3. 记录 APX 作业的作业 ID 号。将在后面的步骤中使用此 ID 号。
4. 通过在 Web 浏览器中输入以下 URL 来运行 SA Twist 界面：

   ```
   https://<core_host>:1032
   ```

   其中 core_host 是 SA 核心服务器的 IP 地址或主机名。此时将在 Web 浏览器中显示 SA API 的 Twist 界面。
5. 选择“Twister”链接。此时将显示 SA API 的 Twister 界面，您可以在其中获取关于 API 接口、包和方法的完整信息，并可以运行方法。
6. 找到并选择 JobService 接口，该接口位于 com.opsware.job 包中。
7. 向下滚动并找到 getProgress() 方法。
8. 选择位于 getProgress() 方法正上方的“尝试”按钮。
9. 输入 SA 凭据。
10. 选择“登录”按钮。
11. 在“ID”字段中，输入正在运行的 APX 的作业编号 (已从上面的步骤 3 中获取)。
12. 选择“运行”按钮。此时将调用 getProgress() 方法，并显示来自 apxprogress 命令的 APX 当前进度信息，如下所示。请注意，在此截图中，步骤总数为 100，已完成的步骤数为 94。有关 getProgress() 方法的输出内容的详细信息，请参见 Javadoc 文档。

可通过在 Twister Web 浏览器的导航窗格中选择 `getProgress()` 方法来访问该文档。



代理工具

代理工具简介

代理工具是一组经过专门设计的 shell 脚本、批处理文件和 Python 脚本，用于检索和修改托管服务器的信息。这些信息将从 SA 数据库中进行检索和修改。

通过使用脚本，您可以检索和修改诸如自定义字段、客户分配、自定义特性等数据。在具备这种能力之后，您可以自动完成许多在过去必须逐台服务器执行的程序。

此外，您可以将脚本检索的信息包含到自己设计的自定义脚本中。由于客户分配和自定义特性等信息对于不同的托管服务器有所不同，因此能够在自定义脚本中实时 检索和使用这些信息非常重要。

例如：

- 您可能具有一个用于处理特定应用程序的后安装配置的脚本，该脚本必须能够发现注册服务器的设施的名称。代理工具将提供一个脚本来获取设施名称，并将其插入到后安装脚本，而无需人工干预。
- 在安装监控代理时，后安装脚本必须修改配置文件，以包括相应特定设施中的监控服务器的 IP 地址。代理工具将提供一个脚本，通过读取核心上的自定义特性来发现监控服务器的 IP 地址，以便将其插入配置文件。
- 可以编写 DSE 用于从很多服务器检索 EEPROM 版本，并将信息存储为自定义特性或自定义字段。

代理工具脚本的一些其他用途包括：

- 在软件安装过程中收集 SA 核心的信息，以在配置中使用。
- 执行 DSE、全局 Shell 脚本或软件安装时，在 SA 数据库中存储来自托管服务器的元数据。
- 检索托管服务器的自定义特性信息。

安装要求

代理工具套件要求满足下列条件：

操作系统支持

代理工具支持 SA 托管服务器所支持的操作系统。有关支持的操作系统列表，请参见《Server Automation 安装指南》。

安全、访问控制和身份验证

在 UNIX/Linux 系统上，代理工具必须作为 root 用户运行；在 Windows 系统上，必须作为 Administrator 运行。代理工具使用服务器代理的证书连接到 Web 服务数据访问引擎 (twist)(这是 pyTwist 的默认行为)，其被授予的权限与 Web 服务数据访问引擎向代理授予的权限相同。这通常适用于代理工具所在服务器上的读取/写入权限，因此，无需执行用户身份验证。

`set_customer` 脚本是一个例外。您必须具有客户读取访问权限，才能将服务器与该客户关联。代理证书没有其他客户的读取访问权限，因此用户必须在运行此脚本时进行身份验证。

如果已启用 UAC (用户访问控制)，则 Windows 上不支持运行代理工具脚本。

其他要求

- pyTwist 的访问权限
- SA API 的访问权限
- 已安装 Python 2.4(服务器代理附带)

安装

在正常的 HPE SA 安装程序核心安装过程中，会将代理工具安装在核心。但是，还必须在托管服务器上安装代理工具，使其在这些服务器上可用。本节描述了该过程。

代理工具将作为一组可执行脚本安装在托管服务器上。根据操作系统，这些脚本将是由 shell 和批处理脚本调用的 shell 或批处理脚本和 Python 脚本。可以从托管服务器运行这些脚本，以在 SA 核心中检索和修改信息。这些脚本可以手动运行，也可以从包安装脚本、DSE、全局 Shell 脚本等进行调用。

代理工具包含在 Python SA API Access (pyTwist) 软件策略中。该策略位于以下目录中：

```
/Opsware/Tools/Python Opsware API Access
```

手动安装代理工具

要在托管服务器上安装代理工具，请执行以下操作：

1. 启动 SA 客户端。
2. 转到“托管服务器”列表，选择要安装代理工具的托管服务器。
3. 单击右键并选择“安装软件”。
4. 选择“Python Opsware API Access”软件策略。
5. “软件策略”安装向导将引导您完成其余过程。

在安装代理时安装代理工具

也可以指定 Python SA API Access 软件策略 ID，并指定在代理安装过程中对其进行修正。有关代理安装的信息，请参见管理。

升级代理工具

由于代理工具以软件策略形式(作为 pyTwist 软件策略的组成部分)提供，因此您可以在升级核心后通过执行修正操作来升级到更新版本的代理工具。

在升级 SA 核心时，Python SA API Access 软件策略也将更新。将删除任何旧版本的代理工具，并向策略中附加新版本。在升级 SA 核心之后(将在升级核心过程中自动升级代理工具)，可以在托管服务器上执行以下任务来升级代理务工具：

1. 选择已安装代理工具的托管服务器。可以通过打开 Python SA API Access 软件策略来查看附加到该策略的服务器和组的列表。
2. 右键单击所选服务器并选择“修正”。
3. 选择“Python Opsware API Access”软件策略。
4. 将删除旧版本的 pyTwist 和代理工具包，并安装新版本。

数据迁移

由于代理工具不会在托管服务器上保留永久数据，因此在数据迁移或保存方面没有要求。

代理工具脚本

用法

<scriptname>.py|bat|sh --arguments

代理工具脚本

脚本	功能
get_all_cust_attr	检索服务器记录的所有自定义特性。 用法：get_all_cust_attr.py [--localonly] [--mode=python|shell|pretty] 模式确定了输出格式(如 Python 字典、shell 语句等)。Pretty 是默认模式。 **备注：** 当存在多行自定义特性时，Shell 模式无法工作。
get_cust_attr	检索单个自定义特性的值。 **用法：** get_cust_attr.py [--localonly] <custom attribute name>
set_cust_attr	设置服务器上单个自定义特性的值。 **用法**: set_cust_attr.py <自定义特性值> <自定义特性值>|--valuefile <值所在文件的路径>
del_cust_attr	从数据库中的服务器记录中删除自定义特性。 **用法：** del_cust_attr.py <custom attribute name>
get_cust_field	检索单个自定义字段的值。 **用法：** get_cust_field.py <custom field name>
set_cust_field	设置服务器上单个自定义字段的值。 **用法**: set_cust_field.py <自定义字段名称> <自定义字段值>|--valuefile <值所在文件的路径>
get_customer	检索与服务器关联的客户名称。

代理工具脚本(续)

脚本	功能
	用法：./get_customer.py
set_customer	设置与服务器关联的客户名称。 **用法**: set_customer.py <客户名称>
get_facility	检索与服务器关联的设施的名称。 **用法**：./get_facility.py
get_info	打印服务器的所有字段(格式类似于 OGSH 中的服务器信息文件)。 **用法**：get_info.py
get_history	打印出服务器特定的事件。 **用法**： get_history.py --startdate <start date in seconds since epoch> [--enddate <end date in seconds since epoch>] [--username <SAS user name>] [--password <SAS password>]
sub_text_file	在文本文件中读取数据、在文件中查找标记/参数、用自定义特性值替换它们，以及将修改的文件打印到 stdout。有关预期的文件格式的详细信息，请参见下文。 **用法**：sub_text_file.py [--localonly] <标记所在文件的路径>

sub_text_file 脚本的格式

传递到 sub_text_file 脚本的文本文件可以包含任何内容，但是该脚本将查找任何含有两个 @ 字符的行，并将这两个 @ 字符之间的字符串 (包括 @ 字符对) 视为标记。您可以在一行中放置一个 @ 字符，此时该字符会被忽略。但是，同一行中的第二个 @ 字符会导致脚本将两个 @ 字符之间的任何文本视为标记。

标记将被 @ 符号之间指定的自定义特性的值替换。例如，字符串 @dns_server@ 被自定义特性 dns_server 的值替换。如果该自定义特性不存在，或者它的值为空，则标记将被替换为一个空字符串。

例如，包含以下条目的文本文件：

```
IP:@monitoring_server_ip@
```

该脚本将输出类似如下所示的内容：

```
IP:82.159.202.117
```

其中 IP 是 monitoring_server_ip 检索到的值。

输出

sub_text_file 脚本会输出到 stdout。如果需要，可以将输出重定向到文件中。此外，还可以使用存储在 zip 文件中的 .template 文件来设置输出格式。例如：

```
$AGENTTOOLSPATH/sub_text_file.sh petstore_config.template > petstore_config.cfg
```

示例代理工具脚本

下面是一些使用代理工具脚本的简单示例。

UNIX/Linux:

下面的示例将输出一条消息，其中包含用户在登录 UNIX 服务器时在每日消息 (MOTD) 中看到的设施名称。

```
./etc/opt/opsware/pytwist/pytwist.conf
facility_name=`$AGENTTOOLSPATH/get_facility.sh`
echo "You have connected to a server in the $facility_name facility.For hardware
information on this server as stored in Opsware, run $AGENTTOOLSPATH/get_info.sh.">
/etc/motd
```

Windows

下面的 Windows 示例将在所有用户的桌面上输出一个包含服务器信息的文本文件。

```
call "C:\Program Files\Common Files\Opsware\etc\pytwist\
pytwist_conf.bat"
```

```
call"%AGENTTOOLSPATH%\get_info.bat" > "%SYSTEMDRIVE%\Documents and Settings\All
Users\Desktop\server_info_from_Opsware.txt"
```

1. 不要对代理工具的路径执行硬编码，而必须执行以下操作：

   获取 PyTwist 配置文件：

   UNIX:
   ```
   ./etc/opt/opsware/pytwist/pytwist.conf
   ```

   Windows：

```
call
C:\Program Files\Common Files\Opsware\etc\pytwist
\pytwist_conf.bat
```

2. 使用环境变量：

   UNIX:
   ```
   $AGENTTOOLSPATH
   ```

   Windows：
   ```
   %AGENTTOOLSPATH%
   ```

   使用此方法可防止将来更改代理工具路径时脚本出现错误。

Microsoft Windows PowerShell - SA 集成

Microsoft Windows PowerShell 简介

Windows PowerShell 是适用于系统管理员和程序员的可扩展命令 shell，已与 Microsoft .Net 2.0 Framework 类库集成。它使用 .NET 公共语言运行时和 .NET Framework，并接受和返回 .NET 对象。因此，增强了可用于管理和配置 Windows 的工具和方法。

Windows PowerShell 提供了大量 cmdlet，这些 cmdlet 内置到 shell 中，提供了丰富的功能。Cmdlet 可以单独或组合使用，以执行更复杂的任务。

Windows PowerShell 不仅支持访问计算机的文件系统，PowerShell 提供程序还允许您访问数据存储，如注册表和数字签名证书存储。提供程序是一个软件模块，可在服务和数据源之间提供统一的接口。

在 SA 中尝试使用 Windows PowerShell 时，将假定您熟悉 Microsoft Windows PowerShell 并能轻松使用。如果需要关于使用 PowerShell 的背景或说明信息，请访问 http://www.microsoft.com。

> **警告：** 因为所包含的 cmdlet 可以修改托管服务器上的数据，所以您必须深入了解 Windows PowerShell 及其用法。

Windows PowerShell 与 SA 集成

在运行 Windows 的托管服务器上，SA 提供了与 Microsoft Windows PowerShell 的初始集成。PowerShell 可在 SA 用户界面中使用，而 SA 数据可从标准 PowerShell 环境或任何 PowerShell 运行空间中获取。PowerShell 运行空间 是 PowerShell 运行时系统的托管环境。

SA 中提供了下列 PowerShell cmdlet：

- Get-SASServer
- Set-SASServer
- Get-SASJob

SA 还包括一个 PowerShell SAS 提供程序(在 PowerShell 环境中提供对 SA 核心内各对象的访问权限的组件)。

集成的 PowerShell/SA Cmdlet

下表列出并描述了 SA 中的集成 PowerShell/SA cmdlet。

PowerShell cmdlet

Cmdlet	描述	参数
Get-SASServer	从指定的服务器检索服务器数据	-Credential <PSCredential> -Core <Hostname|IPAddress> -Name < ListOfHostnameFragments> | -Id <ListOfServerIDs>
Get-SASJob	检索指定作业的数据	-Credential <PSCredential> -Core <Hostname|IPAddress> -JobFilter <ListOfJobIDs>
Set-SASServer	检索托管服务器的列表	-Credential <PSCredential> -Core <Hostname|IPAddress> -Server <ServerVO>

警告： 如果目标核心正在运行 TLSv1.x 的最低协议版本，则 PowerShell 版本(绑定的基础 NET Framework 版本)必须支持它。有关详细信息，请参见 https://msdn.microsoft.com/en-us/library/system.net.securityprotocoltype(v=vs.110).aspx。

安装要求

包含 cmdlet 和 PowerShell SA 提供程序配置集、配置和安装文件的 MSI 安装程序包，用于在系统管理员的 Windows 桌面上进行安装。

操作系统支持

- Windows Server 2003
- Windows Server 2008
- Windows Server 2008 R2 x64
- Windows Server 2012

安装

要实现 Microsoft Windows PowerShell/SA 集成，必须执行以下任务：

- 在 OCC“库”>“软件策略”中找到 Microsoft Windows PowerShell/SA Connector MSI 包。
- 运行此 MSI 安装可定义特定于 SA 的 cmdlet 和 SA 提供程序的配置集。文件 `readme.rtf` 提供了最新信息。此外，还将安装 Microsoft Windows PowerShell 初始化脚本 profile.ps1 (类似于 .bashrc) 和一组示例 PowerShell 脚本 (用于演示如何在 SA 环境中使用 PowerShell)。

默认情况下，此 MSI 会将连接器安装到 `C:\Program Files\Opsware\PsSas` 中。

文件 `SAS-WSAPI.ps1` 描述了如何在不使用 cmdlet 的情况下从 PowerShell 直接访问 WS-API。

Microsoft Windows PowerShell 与 SA 功能集成

可在以下几种情况中使用 Microsoft Windows PowerShell:

远程访问托管服务器

在 SA 客户端中，可以为任何托管服务器(对于服务器组不可用)打开远程 PowerShell 会话，就像打开远程终端一样。

1. 启动 SA 客户端。
2. 从导航窗格中，选择“设备”>“所有托管服务器”。
3. 选择一个托管服务器并将其打开。

   在“设备资源管理器”窗口中，从“操作”菜单选择“启动远程 PowerShell”。

登录到远程 PowerShell 会话之后，将无法运行包含 WMI 调用的脚本。如果尝试运行包含 WMI 调用的脚本，系统将显示“访问被拒绝”错误，即使您属于有权运行该脚本的组也是如此。

审核和快照规则

Microsoft PowerShell 已集成 SA 审核功能。在配置自定义脚本规则时，Microsoft PowerShell 脚本与批处理、Python 2 和 Visual Basic 脚本一起作为选项提供。有关审核的详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

DSE 脚本集成

对于托管服务器，您可以设置将通过 Pytwist 调用 SA API 的 PowerShell 脚本，以使最终用户可以将这些脚本作为 DSE 或 ISM 控件调用。有关编写用于调用 Pytwist API 的脚本的详细信息，请参见通过 Pytwist 执行 Python API 访问 (第 74 页)。

示例会话

本节提供了四个演示如何使用 Windows PowerShell/SA 集成的场景。

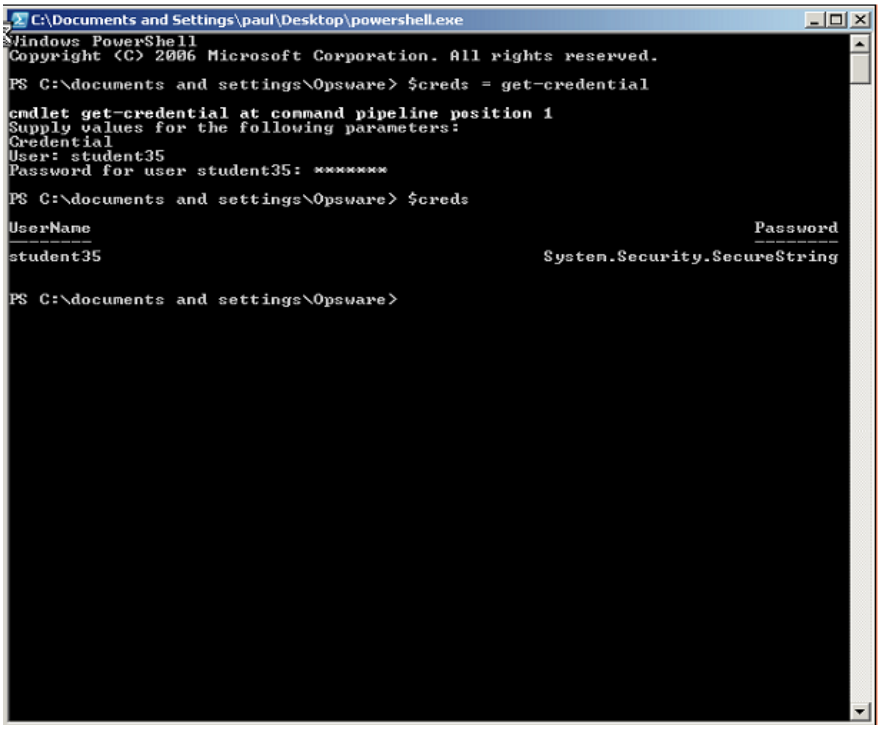
- 场景 1 (第 146 页) 演示了如何从 SA 核心中提取托管服务器数据，以及如何修改这些数据并将其重新写入核心。
- 场景 2 (第 151 页) 演示了如何使用 Windows PowerShell/SA 集成将 SA 托管服务器数据导出到 Excel 电子表格中。
- 场景 3 (第 153 页) 演示了如何将 SA 核心安装为 Windows PowerShell PSDrive，以及如何在虚拟文件系统中导航。
- 场景 4 (第 156 页) 演示了如何列出 Windows PowerShell 环境中可用的所有 SA 对象类型。

场景 1

向 SA 核心进行身份验证、获取关于托管服务器的数据、修改这些数据并将其重新写入 SA 核心。

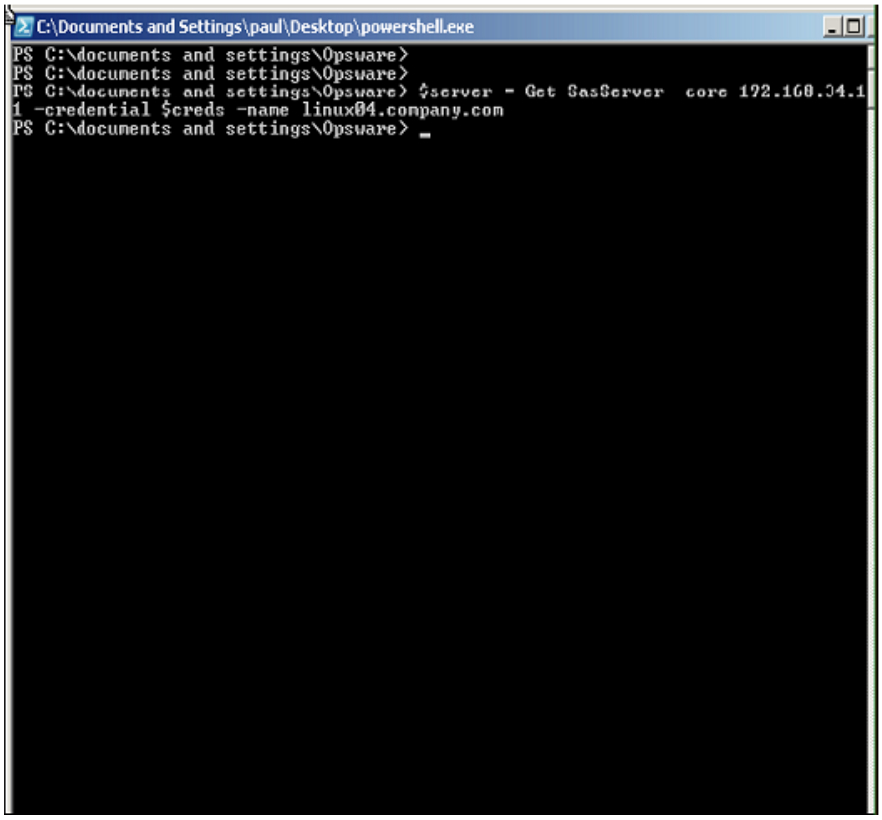
1. 通过桌面图标打开 PowerShell 提示符。
2. 在 PowerShell shell 变量中安全地存储 SA 核心凭据。请参见下图。

在 PowerShell 变量中存储 SA 凭据



3. 使用 Get-SasServer cmdlet，您可以检索代表服务器的 SA 记录，如下图中所示。

   使用 Get-SasServer cmdlet



返回的对象将存储在 shell 变量中。

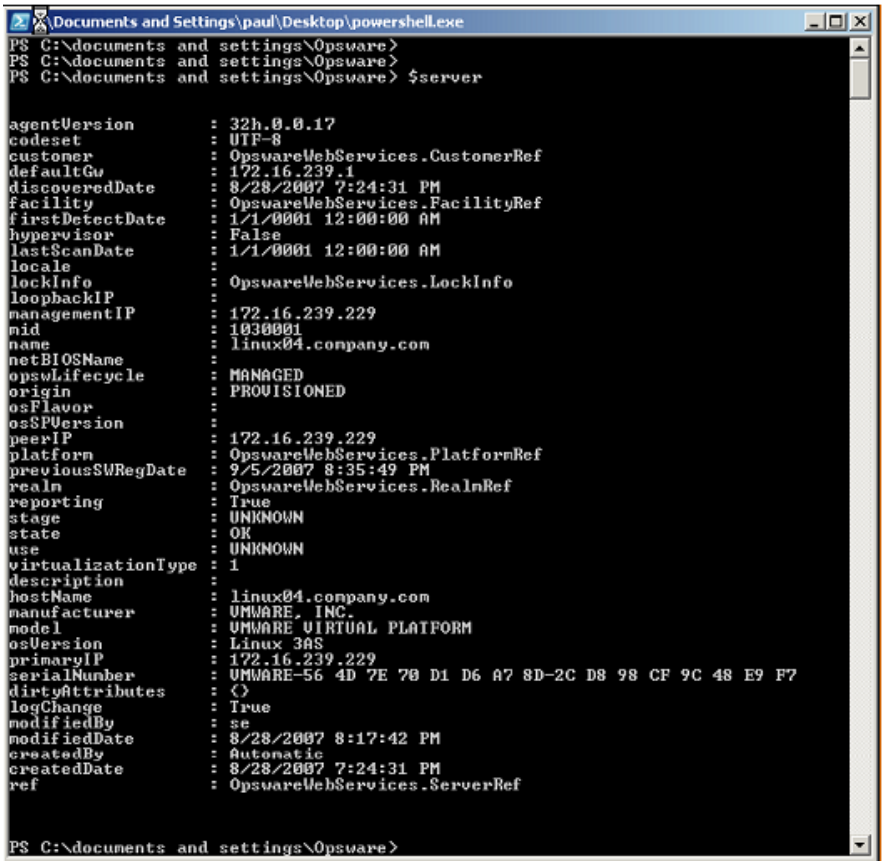
Get-SasServer cmdlet 将使用一个参数确定要从中检索服务器数据的核心，使用另一个参数向核心提供操作凭据 (用于确定和验证要使用其身份尝试操作的用户帐户)，还将使用一个参数来确定所请求的服务器。

可使用 PowerShell Get-Help 基础 cmdlet 获取 Get-SasServer cmdlet 参数或其他任何 cmdlet 的参数的详细信息，例如:

```
Get-Help Get-SasServer -detailed
```

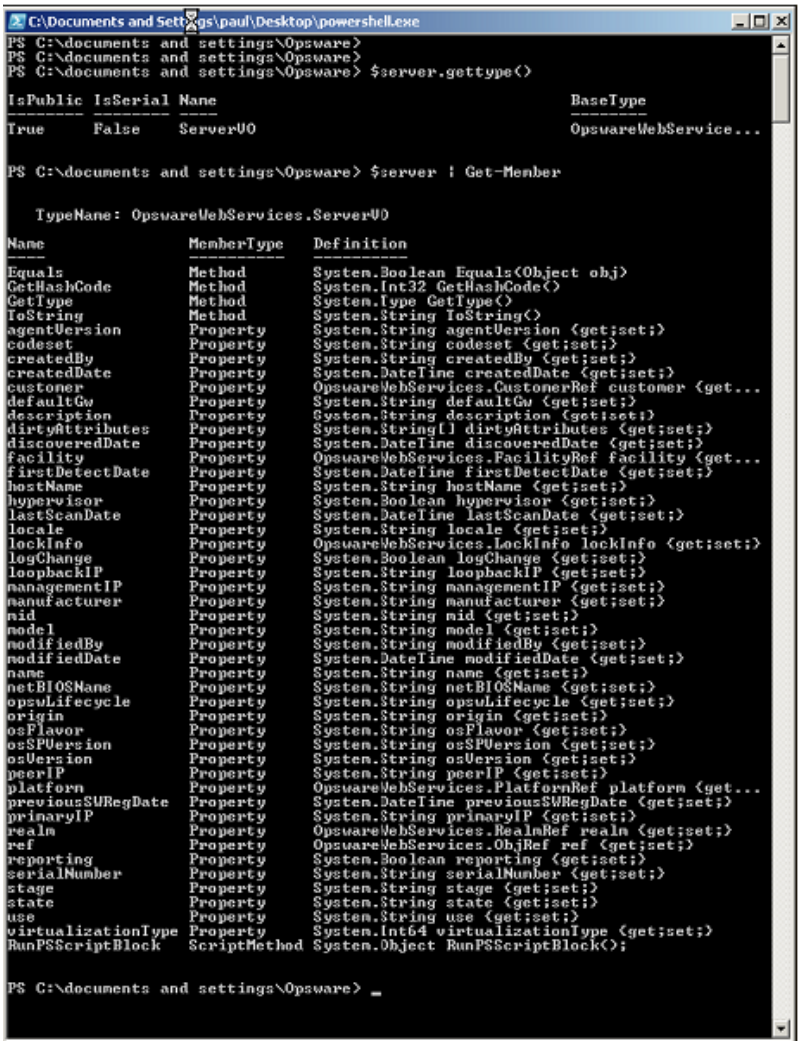
4. 现在，可以通过输入 shell 变量名称来检查所返回对象的属性。请参见下图。

检查 SA 服务器属性



5. 列出对象属性、属性类型，以及可以通过 PowerShell 脚本在对象上调用的方法，如下图中所示。

   列出对象的属性



6. 您可以在 Windows PowerShell 中修改对象的“描述”特性，然后调用 Set-SasServer cmdlet，并将修改后的 ServerVO 对象传递给 cmdlet。此 cmdlet 将接受 ServerVO 对象并更新 SA 核心中的托管服务器记录。Set-SasServer cmdlet 将获取用于标识要写入更新数据的 SA 核心的参数，以及用于标识操作执行身份的 SA 用户帐户的凭据。

   更新操作结束时，会将更新后的 ServerVO 返回到 Windows PowerShell，并在提示符处显示属性，如下图中所示。

   修改对象的描述

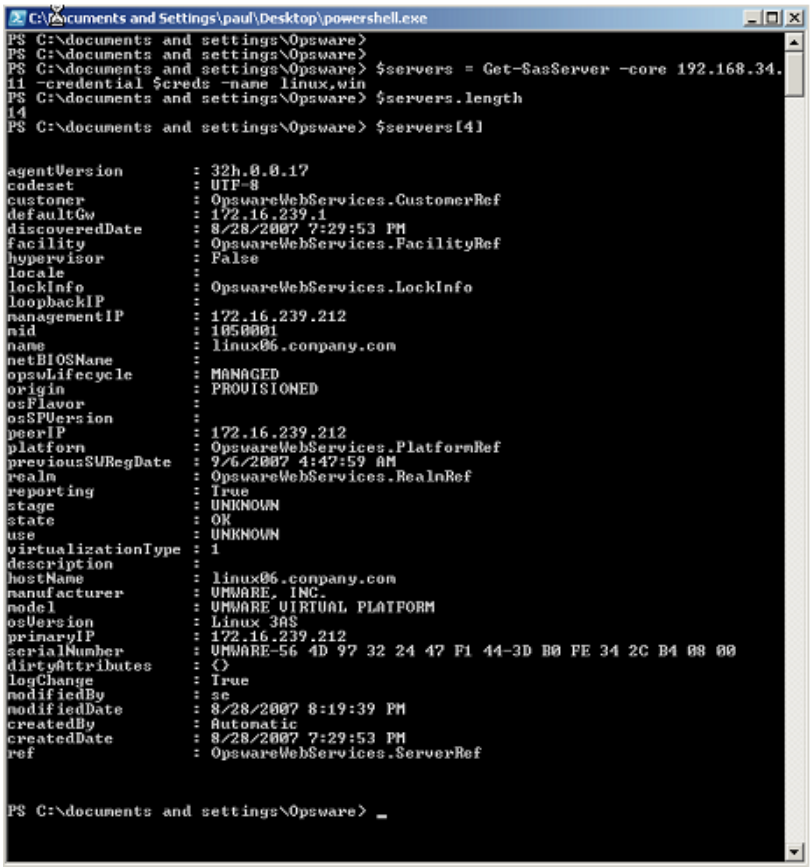
场景 2

此场景演示如何从 SA 核心检索所有托管服务器数据，并将其显示在 Microsoft Excel 中。

1. 可使用 `Get-SasServer` cmdlet 从 SA 核心检索每个 Linux 和 Windows 托管服务器的 ServerVO。在下面的会话中，`-name` 参数用于向 SA 核心提供名称匹配筛选器列表，例如，`-name linux、win`。

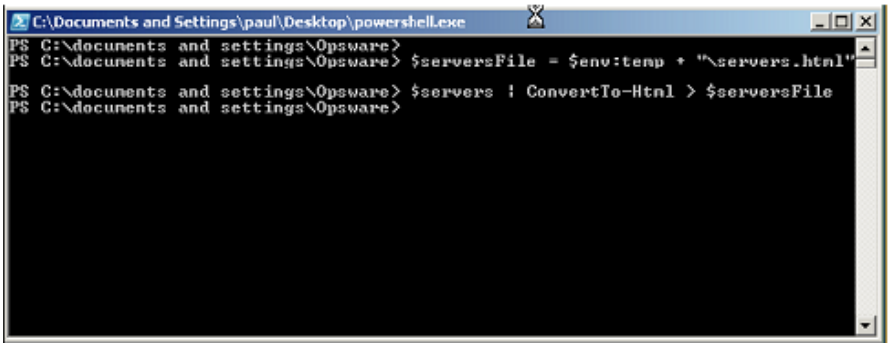
   `Get-SasServer` cmdlet 将返回 ServerVO 数组，在该示例中是包含 14 项的数组。您可以对该数组进行索引，检查其中的任何一个 ServerVO 对象。请参见下图。

   将 Get-SasServer cmdlet 与名称筛选器一起使用



2. 现在，您可以将 ServerVO 数据的格式设置为 HTML，并将其保存到临时文件中。将在 TEMP 目录中创建临时文件。在 PowerShell 会话中，要获取 %TEMP% 环境变量的值，请输入 $env:temp。请参见下图。

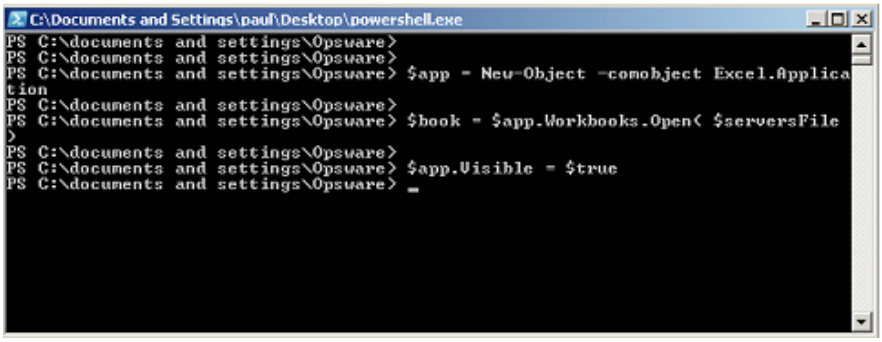
将 ServerVO 数据转换为 HTML 并保存到临时文件



3. 使用 New-Object 基础 Windows PowerShell cmdlet，您可以启动 Microsoft Excel，然后在该 Excel 实例中创建新工作簿，并使用临时文件的内容填充该工作簿。最后，将正在运行的 Excel 实例设置为可见。这样会将 Excel 置于前台。现在，您可以按日期、列值

等对数据进行排序，以确定诸如每台服务器接受核心管理的日期等信息。请参见下图。

使用 New-Object cmdlet 以启动 Microsoft Excel

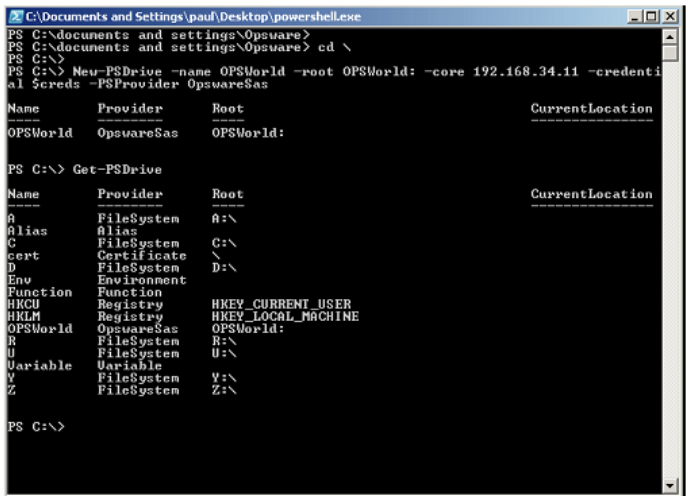


场景 3

此场景演示如何将 SA 核心安装为 Windows PowerShell PSDrive、如何导航到 SA **Jobs** 文件夹，以及如何检索其内容。

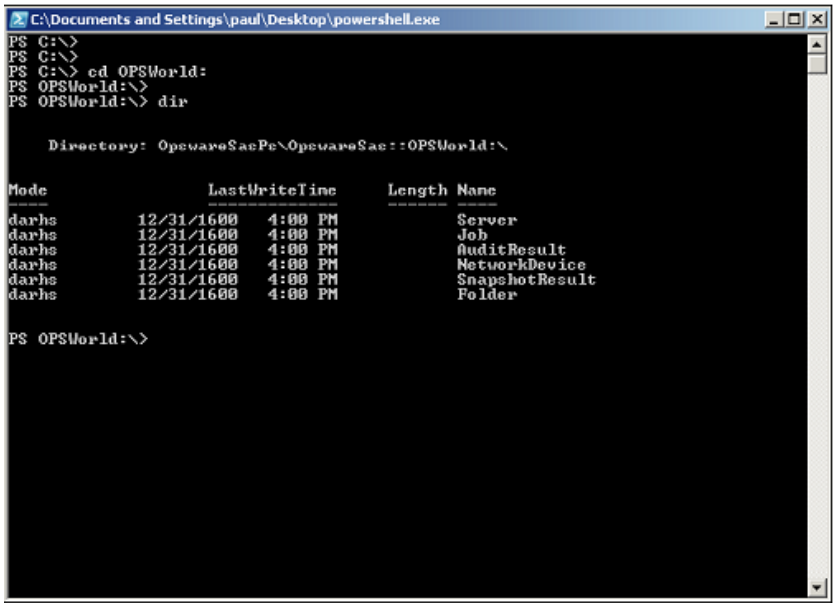
1. 将 SA 核心安装为 Windows PowerShell PSDrive。PowerShell 支持将不同的数据存储或库当做文件系统一样导航。在此场景中，应安装 SA 核心，特别是托管环境数据存储，在此假定它是名为 OPSWorld 的驱动器。然后，每当对此虚拟文件系统进行数据读取或写入或者客户端对此文件系统导航时，Windows PowerShell 基础系统就会调用 PowerShell SAS 提供程序 -PSProvider OpswareSas。请参见下图。

   将 SA 核心安装为 Windows PowerShell PSDrive



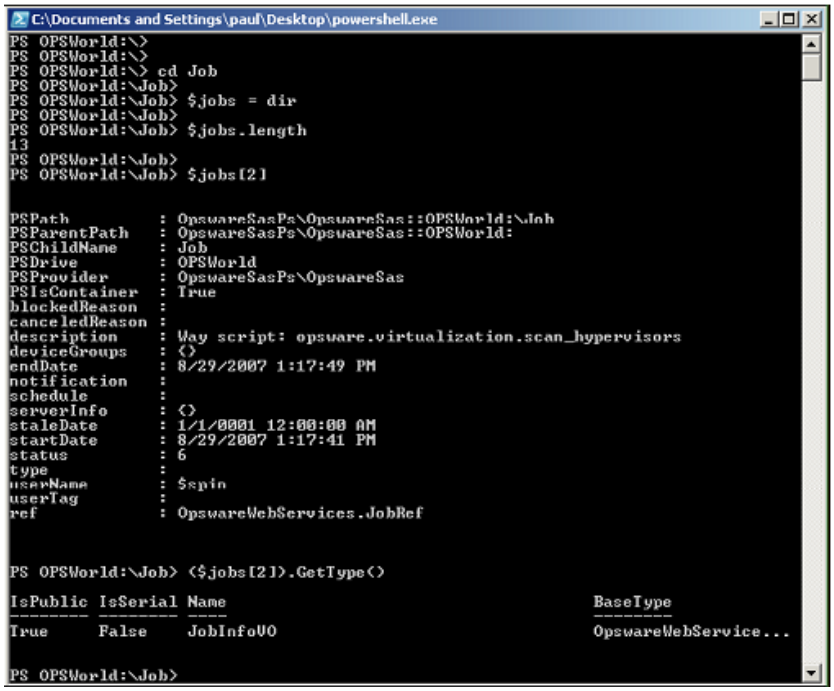
2. 将目录更改为新安装的驱动器并获取目录列表。dir 是 Get-ChildItem cmdlet 的 PowerShell 别名。请参见下图。

DIR 充当 Get-Child cmdlet 的别名



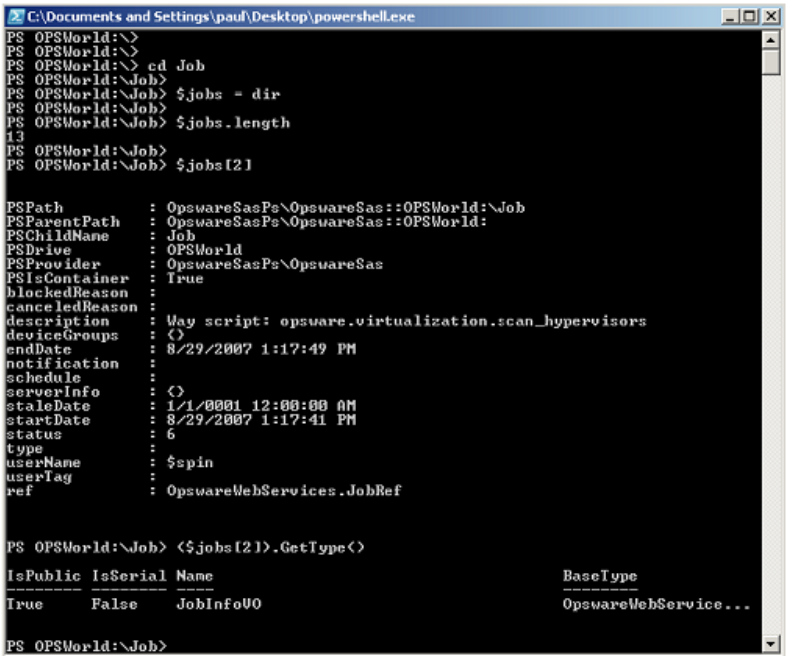
3. 将目录更改为 **Jobs** 文件夹，获取目录列表并将其保存为 shell 变量。该 shell 变量将包含 JobInfoVO 对象 (来自核心) 的数组，可以对该数组进行索引。

将目录列表保存为 PowerShell 变量



4. 将目录更改为 C: 驱动器并删除 OPSWorld PSDrive。

删除 OPSWorld PSDrive

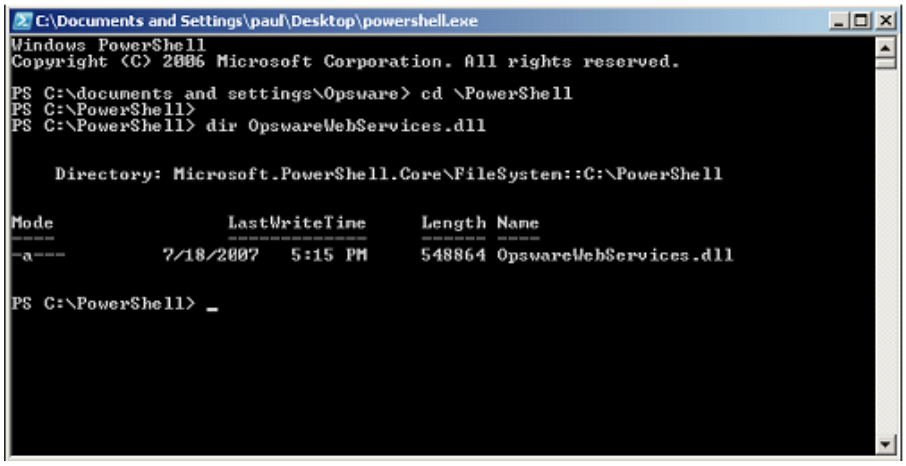


场景 4

此场景描述了如何检查 Windows PowerShell 环境中可用的所有 SA 对象类型。

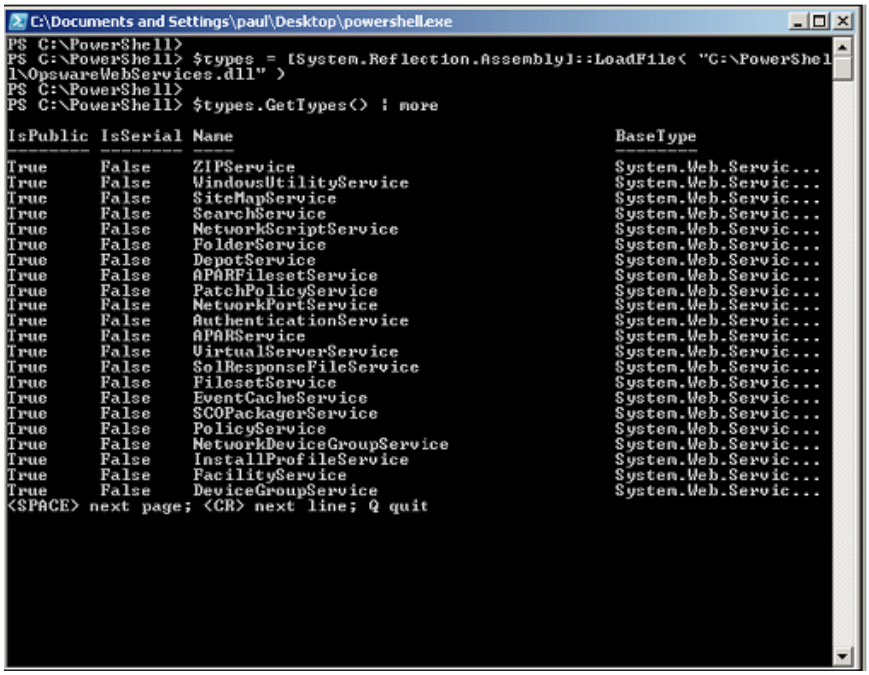
1. 找到包含 PowerShell SAS 提供程序和 cmdlet 的 .NET 配置集。请参见下图。

   找到包含 PowerShell SAS 提供程序和 cmdlet 的 .NET 配置集。



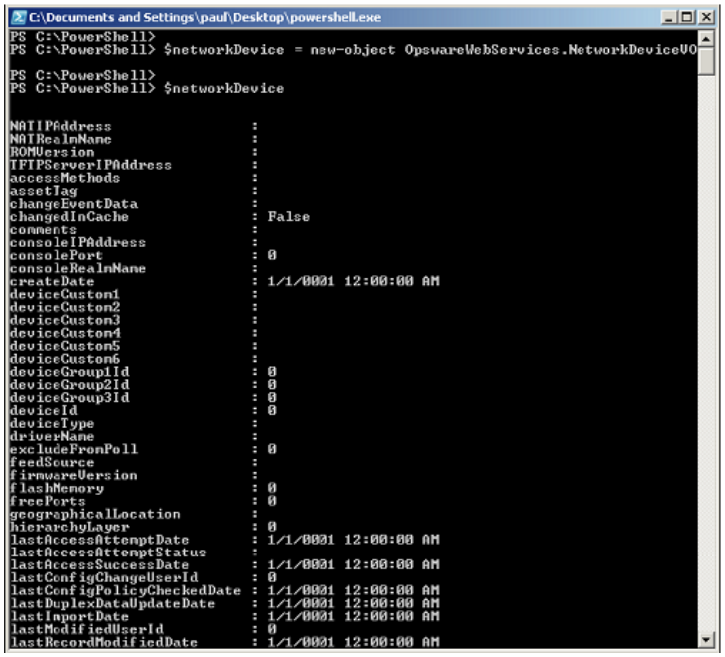
2. 使用 .NET Reflection 加载 .NET 配置集，并检查所加载的类型。这将显示所有可用于 Windows PowerShell 环境的 SA 类型。请参见下图。

加载 .NET 配置集并检查类型



3. 创建 NetworkDeviceVO 实例。这是一个初期 NetworkDeviceVO，其中显示了 PowerShell 环境中可用于脚本、报告等功能的所有网络设备特性。请参见下图。

创建 NetworkDeviceVO 实例



Java RMI 客户端

Java 远程调用 (RMI) 客户端可以从具有 SA 核心的网络访问权的服务器调用 SA API 方法。运行客户端的服务器不必是 SA 核心或托管服务器。当客户端连接到核心时，它会指定 SA 用户名和密码，这与登录到 SA 客户端的最终用户非常相似。用户所属的组确定了客户端可用的 SA 资源和任务。

本主题的目标读者包括熟悉 SA 基础知识和 Java 编程语言的软件开发人员。

Java RMI 客户端安装

在开发 SA API 的 Java RMI 客户端之前，请执行以下步骤：

1. 在开发环境中安装 SA 核心。请不要使用生产核心。
2. 获取一台要在其中构建和运行 Java RMI 客户端的开发服务器。
3. 在开发服务器上，安装 Java SE 7 SDK。
4. 验证开发服务器是否已通过网络连接到运行 OCC 组件的 SA 核心服务器。
5. 将 opswclient.jar 文件从 SA 核心服务器下载到开发服务器中。opswclient.jar 文件包含 SA API 的 Java RMI 存根。在编译和运行 Java RMI 客户端时，应在 classpath 选项中包含 opswclient.jar。
6. 要下载 opswclient.jar，请执行下列操作之一：
   a. 指定以下 URL，其中 occ_host 是运行 OCC 组件的核心服务器：

      ```
      https://occ_host/twister/opswclient.jar
      ```

   b. 转到以下目录：/opt/opsware/twist/extlib/client.

也需要 spinclient-latest.jar 和 opsware_common-latest.jar 文件。这些文件可从 /opt/opsware/twist/lib/ 中运行的

SA 核心获得。

当编译和运行这些示例时，必须将 .jar 文件添加到 classpath 参数中。

Java RMI 示例

本节描述了一个名为 GetServerInfo 的简单 Java RMI 客户端。

GetServerInfo 客户端通过完整或部分主机名搜索托管服务器，主机名可作为命令行参数指定。对于找到的每个托管服务器，客户端将打印服务器的名称、管理 IP 地址和操作系统版本。

GetServerInfo 客户端执行以下步骤：

1. 连接到 SA：
```
// Set the JNDI provider
client.setJNDIProvider( "https" , host , ( short ) 1032 , null, newString[] {
OPSWARE_CA_CRT_PATH } , null ) ;
// Force a reconnection
client.getContext( true ) ;
// Set and authenticate the user
client.setAPIUser( new APIUserImpl(username , password)) ;
```
2. 获取对 ServerService 接口的引用：
```
serverSvc = (ServerService)OpswareClient.getService
(ServerService.class);
```
3. 在 ServerService 上调用方法：
```
ServerRef[] serverRefs = serverSvc.findServerRefs(filter);
...
ServerVO[] serverVOs = serverSvc.getServerVOs(serverRefs);
...
System.out.println(serverVOs[i].getName());
```

编译和运行 GetServerInfo 示例

在编译和运行该示例之前，请执行以下任务：

1. 获取 opsware_common-latest.jar、spinclient-latest.jar 和 opswclient.jar 文件，如 Java RMI 客户端安装中所示。
2. 下载包含演示程序 GetServerInfo.java 文件的 ZIP 文件。
3. 要编译此客户端，请为 classpath 参数指定 opsware_common-latest.jar、spinclient-latest.jar 和 opswclient.jar 文件：

```
javac -classpath :path/opswclient.jar:path/opsware_common-
latest.jar:path/spinclient-latest.jar GetServerInfo.java
```

4. 要运行客户端，请输入以下命令，其中 target 是 SA 所管理的服务器的完整或部分名称。注意: Windows 的 Java 类路径分隔符是 ";"。

   ```
   java -classpath .:path/opswclient.jar:path/opsware_common-
   latest.jar:path/spinclient-latest.jar \GetServerInfo [options]    target
   ```

   在以下示例中，GetServerInfo 连接到主机 c44(运行 OCC 核心组件的主机)和端口 443 上的 SA。程序显示主机名包含字符串 opsw 的托管服务器的信息。

   ```
   java -classpath .:/home/jdoe/opswclient.jar:/home/jdoe/opsware_common-
   latest.jar:/home/jdoe/spinclient-latest.jar \GetServerInfo --host
   c44.dev.example.com --port 443 opsw
   ```

5. 对 SA 用户名和密码提示消息做出响应。SA 用户必须有权读取与命令行上指定的 target 匹配的服务器。

Web 服务客户端

Web 服务客户端概述

SA API 支持 Web 服务，这是基于 SOAP(简单对象访问协议)和 WSDL(Web 服务定义语言)等开放行业标准构建的编程环境。您可以使用如 Perl 和 C# 等各种编程语言 (将在本节后面介绍)，或通过支持 Web 服务的开发环境 (例如 Microsoft Visual Studio .NET) 来创建 Web 服务客户端。

本主题的目标读者包括熟悉 SA 基础知识和 Web 服务开发的软件开发人员。

本发行版中提供的编程语言绑定

本 SA 发行版包括用于 C# 的 Web 服务客户端存根。以 Perl 语言编写的 Web 服务客户端不需要客户端存根。

本发行版不包括用于 Java 或 Python 的 Web 服务客户端存根。但是，Java 客户端可以通过 RMI 访问 SA API，Python 客户端可以通过 Pytwist 访问 SA API，如前面章节中所述。

服务位置和 WSDL 的 URL

客户端可以从具有以下语法的 URL 访问 Web 服务，其中 host 是运行 OCC 核心组件的服务器，port 是 HTTPS 代理的端口。(默认代理端口为 443。)packageName 对应于服务所属的 Java 库。

```
https://host:port/osapi/packageName/WebServiceName
```

可从具有以下语法的 URL 访问 WSDL 文件：

```
https://host:port/osapi/packageName/WebServiceName?WSDL
```

例如，以下 URL 指向 FolderService 的位置和 WSDL：

```
https://occ.c38.example.com:443/osapi/com/opsware/folder/FolderService
```

```
https://occ.c39.example.com:443/osapi/com/opsware/folder/FolderService?wsdl
```

SOAP 绑定样式为 RPC(远程过程调用)，传输协议为 HTTPS。

Web 服务客户端安全性

与 SA API 的其他客户端一样，Web 服务客户端必须经过身份验证和授权才能在 SA 中执行操作。SA 核心中的客户端与 Web 服务组件之间的通信将加密。访问操作限制为通过 OCC 核心组件的 HTTPS 代理端口的 HTTPS 客户端。(默认端口为 443。)

重载操作

SA API 具有重载操作，但是 WSDL 2.0 规范不支持重载。在 SA API 中，重载操作以单个操作的形式由 Web 服务提供。

Java 接口支持

SA API 使用 Java 接口，但是 Web 服务不支持接口。解决该问题的方法是，使用 WSDL 文件将接口映射到 `xsd:anyType`。对于以面向对象的编程语言(如 C#)编码的客户端，如果 API 方法返回一个接口，则必须将返回类型强制转换为具体类。接口数组将转换为 `Object[]`。特定类型的数组成员将通过序列化/反序列化保留。有关 C# 代码的示例，请参见处理接口返回类型 (第 175 页)。

不支持的数据类型

SA API 使用以下数据类型，但是 SOAP 不支持这些数据类型：

```
java.util.Properties
com.opsware.common.ModifiableMap
com.opsware.acm.ValueSet
com.opsware.swmgmt.PolicyOverrideFilter
```

Web 服务中省略的方法

以下 SA API 方法使用不受支持的数据类型作为参数或返回类型。因此，Web 服务中未以操作形式提供这些方法。

```
com.opsware.custattr.CustomAttribute.getCustAttrs
com.opsware.custattr.CustomAttribute.setCustAttrs
```

com.opsware.custattr.CustomField.getCustomFields
com.opsware.custattr.CustomField.setCustomFields
com.opsware.pkg.Patch.getPolicyOverrideRefs

对 java.util.Map 的部分支持

Axis 会将 java.util.Map 转换为 apachesoap:Map，即一组“键-值”对。对于 .NET，无法执行该转换。例如，C# 客户端将接收到空的“键-值”对数组。但是，该转换可在 Perl 语言编写的 Soap::Lite 中正常工作。因此，使用 java.util.Map 的 SA API 方法将在 Web 服务中以操作形式提供。

以下方法使用 java.util.Map 作为参数或返回类型：

com.opsware.acm.GroupConfigurable.getApplicationInstances
com.opsware.acm.ServerConfigurable.getCustAttrsWithRC
com.opsware.compliance.sco.CMLSnapshot.getValueSet
com.opsware.compliance.sco.CMLSnapshot.setValueSet
com.opsware.compliance.sco.SnapshotResultService.remediateCMLSnapshot
com.opsware.custattr.VirtualColumnVO.getConfigInfo
com.opsware.custattr.VirtualColumnVO.setConfigInfo

使用不受支持的数据类型的 VO 方法

以下 VO 方法使用不受支持的数据类型作为参数或返回类型：

com.opsware.acm.ApplicationInstanceVO.getValueset
com.opsware.acm.ApplicationInstanceVO.setValueset
com.opsware.acm.ConfigurableVO.getValueset
com.opsware.acm.ConfigurableVO.setValueset
com.opsware.virtualization.VirtualConfigNode.getProperties
com.opsware.virtualization.VirtualConfigNode.setProperties
com.opsware.virtualization.VirtualServerConfig.getProperties
com.opsware.virtualization.VirtualServerConfig.setProperties

创建或更新 VO 时调用 setDirtyAtrributes

Web 服务客户端必须首先调用 setDirtyAttributes，然后才能对服务调用 create 或 update 方法。setDirtyAttributes 方法将显式标记需由 create 或 update 调用设置的 VO 的特性(字段)。由 setDirtyAttributes 指定的特性名称区分大小写。

例如，为修改 FolderVO 对象的 description 特性，下列代码将首先调用 setDirtyAttributes，然后调用 update：

```
// fs is FolderService
FolderVO folderVO = fs.getFolderVO(folderRef);
folderVO.setDescription("credit card processing");
folderVO.setDirtyAttributes(new String[]{"description"});
fs.update(folderRef, folderVO, true, true);
```

Axis 从 XML 反序列化 XML 对象的方式要求必须对 Web 服务客户端调用 setDirtyAttributes。如果不调用 setDirtyAttributes，Axis 将对所有 VO 特性(包括只读特性)调用 setter，从而导致 ReadOnlyException。

与 SA Web 服务 API 2.2 的兼容性

SA Web 服务 API 2.2 与本指南中描述的 SA API 不兼容。方法签名、服务、WSDL 和端口绑定不相同。如果要创建新的 Web 服务客户端，请务必使用 SA API 而不是 SA Web 服务 API 2.2。

Perl Web 服务客户端

本节包含用于创建将访问 SA API 的 Perl Web 服务客户端的分步式说明和示例代码。

Perl 客户端所需的软件

开发环境中必须具备以下 Perl 模块：

- Crypt-SSLeay-0.57
- IO-Socket-SSL-1.31
- Net-SSLeay-1.35
- HTML-Parser-3.64
- MIME-Base64-3.08
- URI-1.40

- libwww-perl-5.833
- SOAP-Lite-0.710

基于您的 Perl 版本，可能需要这些模块的新版本。

> **警告：** 如果 "500 SSL negotiation failed:" 错误仍然存在，则 OpenSSL 需要更新到版本 1.0.1 或更高版本。对于 RHEL 系列，OpenSSL 需要更新到版本 1.0.1e-30 或更高版本。

运行 Perl 演示程序

要运行演示程序，请执行以下步骤：

1. 从支持网站中，获取与 **All Manuals Download SA 10.5** 文件夹一起绑定的 **SA_Platform_Developer_Guide_examples.zip** 文件。
2. 从以下位置获取演示程序 uapisample.pl 文件：SA_Platform_Developer_Guide_examples.zip\SA_Platform_Developer_Guide_examples\api_examples\web_services\perl。
3. 编辑 uapisample.pl 文件，更改 host、username、password 以及诸如 serverID 等对象 ID 的硬编码值。
4. 运行 uapisample.pl。
5. 如果收到“Certificate Verify Failed”错误，您必须从样本文件中取消备注下列行，并将有效路径提供给证书文件：

   ```
   #$ENV{HTTPS_CA_FILE} = "path_to/opsware-ca.crt";
   ```

   您可以在 SA 核心的以下路径中找到该证书文件：

   ```
   /var/opt/opsware/crypto/twist/opsware-ca.crt
   ```

Perl 示例代码

下面的代码片段摘自 uapisample.pl(包含在之前下载的 ZIP 文件中的一个 Perl 程序)。

设置服务 URI

```
# Construct the URI for the service.
#
my $username = "integration";
my $password = "integration";
my $protocol = "https";
my $host = "occ.c38.dev.example.com";
my $port = "443";
my $contextUri = "osapi/com/opsware/";
my $folderServiceName = "folder/FolderService";
my $folderUri = "http://www.example.com/" .$contextUri .
$folderServiceName;
# Create a proxy to the FolderService.
#
my $folderProxy = $protocol .":".$username .":".$password ."@" .
$host .":".$port ."/" .$contextUri .$folderServiceName;
```

启动新服务

```
my $folderPort = SOAP::Lite
-> uri($folderUri)
-> proxy($folderProxy);
```

调用服务方法

```
my $root = $folderPort->getRoot()->result();
print 'Got root folder:' .$root->{'name'} ."\n";
# Alternative:
my $root = $folderPort->SOAP::getRoot();
print 'Got root folder:' .$root->{'name'} ."\n";
```

获取 VO

```
$rootVO = $folderPort->getFolderVO(SOAP::Data->name('self')
->value(\SOAP::Data->name('id')->type('long')->value(0)))
->result();
# The preceding call to getFolderVO does not pass a FolderRef
# parameter.If a method such as FolderService.remove accepts a
# FolderRef parameter, use the following code:
#
my $folderToBeRemoved = SOAP::Data->name('self')
->attr({ 'xmlns:ns_fs' => 'http://folder.example.com/FolderService'}) -
>type('ns_fs:FolderRef')->value(\SOAP::Data->name('id')->type('long') -
>value(123456));
$folderPort->remove($folderToBeRemoved);
# To see the Perl representation of the returned VO, you can use
# the Dumper method.This will help you understand how to
# construct the dirty attributes of a VO for a create or update
# method.
#
use Data::Dumper;
print Dumper($folderVO);
```

获取数组

```
# Construct $folder, the FolderRef before getting the array.
#
my $folder = SOAP::Data->name('self') ->attr({ 'xmlns:ns_fs' => 'http://
folder.example.com'}) ->type('ns_fs:FolderRef')->value(\SOAP::Data-
>name('id')->type('long') ->value($root->{'id'}));
# The getChildren method returns an array of FNodeReference
# objects.
#
my $children = $folderPort->getChildren($folder, SOAP::Data->name('type')-
>type('string')->value(''))->result();
foreach $child (@{$children}){
print 'Get child:' .$child->{'name'} ."\n";
}
```

构造对象数组

```
# For a function that takes an object array as a parameter,
# such the getVOs method, take the following approach:
#First, construct the Array object elements individually
# and put them in an array.
#
my @refs = [];
foreach my $ref (@{$myRefs}){
       # Assume myRefs was returned from a previous
               # Web Services call.
       my $object = SOAP::Data->name('FacilityRef')
               ->value(\SOAP::Data->name('id')
                       ->type('long')
                       ->value($ref->{'id'}
               )
       )->attr({ 'xmlns:facility' => 'http://locality.example.com'})
       ->type('facility:FacilityRef');
   push @refs, $object;
}
# Second, construct an Array Object and put the array in it.
#
my $selves = SOAP::Data->name("selves" =>\SOAP::Data->name("element" => @refs)-
>type("facility:FacilityRef"))
               ->attr({ 'xmlns:facility' => 'http://locality.example.com'})
               ->type("facility:ArrayOfFacilityRef");
```

更新或创建 VO

```
# This example updates the description attribute of a ServerVO.
#
my $serverID = 40038;
my $server = SOAP::Data->name('self')->value(\SOAP::Data->name('id')-
```

```
>type('long')->value($serverID));
# Don’t forget to set dirtyAttributes for the attributes
# you want to update.You also need dirtyAttributes for
# create methods that pass a VO.
#
my @dirtyAttrs = ('description');
my $serverVO = SOAP::Data->name('vo') ->attr({ 'xmlns:ns_ss' => 'http://
server.example.com'}) ->value(\SOAP::Data->value( SOAP::Data-
>name('description')->value('PERL_UPDATE_DESC')->type('string'), SOAP::Data-
>name('logChange')->value('false')->type('boolean'), SOAP::Data-
>name('dirtyAttributes' => \SOAP::Data->name("element" => @dirtyAttrs)-
>type("string")) ->type("ns_ss:ArrayOf_soapenc_string"), ));
my $force = SOAP::Data->name('force')->value('true')->type('boolean');
my $refetch = SOAP::Data->name('refetch')->value('true')->type('boolean');
# Call the update method.
#
print 'Invoking method serverWSPort.update...', "\n";
my $updatedServerVO = $serverWSPort->update(
               $server,
               $serverVO,
               $force,
               $refetch)->result();
print "New description:", $updatedServerVO->{'description'}, "\n";
```

处理 SOAP 错误

```
# Make sure that you turn off on_fault subroutine in the
# "use SOAP::Lite ..." statement.
#
# The fault member of a SOAP return will be set if the Web
# Service call throws an exception.
#The following code tries to get a folder that does not exist:
#
my $testVO = $folderPort->getFolderVO(SOAP::Data->name('self') -
>value(\SOAP::Data->name('id')->type('long')->value(123456)));
if($testVO->fault){
       print $testVO->faultstring ."\n";
       # This will print the error msg.
       print "ExceptionName:" . getExceptionName($testVO) ."\n"; # A
NotFoundException should be displayed here
       # The code that deals with the error goes here....
}
...
#The following subroutine extracts the exception name from the
# returned faultdetail.
#
sub getExceptionName {
       my $fault = shift; #get the fault object
       if($fault->faultdetail->{'fault'}){
```

```
                return ref($fault->faultdetail->{'fault'});
        }
}
...
#As shown in the preceding code, it’s easier to handle SOAP
# faults if you execute functions like this:
#
# my $data = $port->function(...);
# Not like this:
#$port->SOAP::function(...);
# $port->function(...)->result;
```

构造 Web 服务的 Perl 对象

在调用 Web 服务操作之前，Perl 客户端必须设置输入参数所需的数据结构。API 文档 (javadoc) 以及服务的 WSDL 文件中提供了设置数据结构所需的信息。本节中的 Perl 代码示例显示了如何构造 getServerVO 操作的输入参数。代码后的分步式说明显示了从 API 文档和 WSDL 文件中获取输入参数信息的位置。

用于调用 getServerVO 的源代码

以下 Perl 代码将设置输入参数 self，然后调用 getServerVO 操作。该调用将检索 ID 为 12345 的托管服务器的 VO(值对象)。

```
# Create a top-level SOAP::Data object
#
$self = SOAP::Data->name(’self’)
# The namespace corresponds to the schema of the data type
# of the SOAP:Data object.The name chosen (ns_ss) is
# arbitrary.
#
$self->attr({’xmlns:ns_ss =>
’http://server.example.com/ServerService’});
# Specify the type (ServerRef) for the parameter self, using the
# name of the namespace from the preceding statement.

#
$self->type(’ns_ss:ServerRef’);
# Create the value for the parameter.The value is a pointer
# to a SOAP::Data object.The number 12345 is the SA ID of a managed server.
#
my $id = SOAP::Data->name(’id’)->type(’long’)->value(12345);
# From the self object, point to the value.
#
$self->value(\$id);
# Finally, call getServerVO:
```

```
#
my $data = $serverPort->getServerVO($self);
if($data->fault){
       # Handle exceptions here ...
}
else{
       my $serverVO = $data->result;
}
...
```

getServerVO 设置信息的位置

要获取编写 getServerVO 调用的代码所需的信息，请执行以下步骤：

1. 在浏览器中，访问位于以下 URL 的 API 文档 (javadoc)：

   ```
   https://occ_host:1032/twister/docs/index.html
   ```

   其中 occ_host 是运行命令中心组件的核心服务器的 IP 地址或主机名。(有关在 Twister 中调用方法的说明，请参见 API 文档和 Twister (第 35 页)。)

2. 检查 API 文档，确定方法的输入参数和返回值。

   getServerVO 方法在接口 com.opsware.server.ServerService 中定义。在以下方法签名中，注意 getServerVO 将接受 ServerRef 作为参数，并返回 ServerVO：

   ```
   public ServerVO getServerVO(ServerRef self)
          throws java.rmi.RemoteException,
                  NotFoundException,
                  AuthorizationException
   ```

3. 在浏览器中，指定以下 URL 打开 ServerService 的 WSDL 文件：

   ```
   https://occ_host/osapi/com/opsware/server/ServerService?wsdl
   ```

4. 在 WSDL 文件中，找到 ServerService 的命名空间：

   ```
   <schema targetNamespace="http://server.example.com"
   xmlns="http://www.w3.org/2001/XMLSchema">
   ```

   以下 Perl 语句(来自前面列出的代码)指定了命名空间：

   ```
   $self->attr({'xmlns:ns_ss =>
   'http://server.example.com/ServerService'});
   ```

5. 在 WSDL 文件中，找到 getServerVO 操作，并注意输入消息名称 getServerVORequest。

```
<wsdl:operation name="getServerVO" parameterOrder="self">
       <wsdl:input message="impl:getServerVORequest" name="getServerVORequest"/>
       <wsdl:output message="impl:getServerVOResponse" name="getServerVOResponse"/
>
       <wsdl:fault message="impl:NotFoundException" name="NotFoundException"/>
       <wsdl:fault message="impl:AuthorizationException"
name="AuthorizationException"/>
</wsdl:operation>
```

6. 在 WSDL 文件中，找到 getServerVORequest 消息：

   ```
   <wsdl:message name="getServerVORequest">
          <wsdl:part name="self" type="impl:ServerRef"/>
   </wsdl:message>
   ```

   getServerVORequest 消息元素定义了 getServerVO 的输入参数的名称 (self) 和类型 (ServerRef)。以下 Perl 语句指定了 ServerRef：

   ```
   $self->type('ns_ss:ServerRef');
   ```

7. 在 WSDL 文件中，找到 ServerRef 的 complexType：

   ```
   <complexType name="ServerRef">
          <complexContent>
                 <extension base="tns1:ObjRef">
                        <sequence>
                               <element name="secureResourceTypeName" nillable="true"
   type="soapenc:string"/>
                               </sequence>
                        </extension>
                 </complexContent>
          </complexType>
   ```

   请注意 ServerRef 扩展了 ObjRef。

8. 在 WSDL 文件中，找到 ObjRef 的 complexType：

   ```
   <complexType abstract="true" name="ObjRef">
          <sequence>
                 <element name="id" type="xsd:long"/>
                 <element name="idAsLong" nillable="true" type="soapenc:long"/>
                 <element name="name" nillable="true" type="soapenc:string"/>
          </sequence>
   </complexType>
   ```

   在 ObjRef 中，注意名称 (id) 和类型 (long)。这些数据类型在以下 Perl 语句中指定：

   ```
   my $id = SOAP::Data->name('id')->type('long')->value(12345);
   ```

C# Web 服务客户端

本节包含用于创建将访问 SA API 的 C# Web 服务客户端的分步式说明和示例代码。

C# 客户端所需的软件

要开发 C# Web 服务客户端，您的开发环境必须具备以下软件：

- Microsoft .NET Framework SDK 1.1 版
- SA API 的 C# 客户端存根

获取 C# 客户端存根

SA 为每个服务提供一个存根文件，例如 FolderService.cs。所有存根都具有相同命名空间：OpswareWebServices。除存根之外，SA 还提供了 shared.cs 文件，该文件包含 ServerRef 等共享类。

要获取包含 C# 存根的 ZIP 文件，请指定以下 URL，其中 occ_host 是运行 OCC 组件的核心服务器：

```
https://occ_host:1032/twister/opswcsharpclient.zip
```

在服务和对象中定义的常量未在 C# 存根中进行定义。要获取有关常量的信息，请使用 API 文档(javadoc)，如常量字段值 (第 36 页)中所述。

构建 C# 演示程序

要构建该演示程序，请执行以下操作:

1. 从支持网站中，获取与 **All Manuals Download SA 10.5** 文件夹一起绑定的 **SA_Platform_Developer_Guide_examples.zip** 文件。

   SA_Platform_Developer_Guide_examples.zip 包含 SA_Platform_Developer_Guide_

examples\api_examples\web_services\csharp 位置中的下列演示程序文件:

- App.config - 应用程序设置
- WebServicesDemo.cs - 用于调用服务方法的客户端代码
- MyCertificateValidation.cs - 证书验证类

2. 创建以下目录：
   C:\wsapi
3. 从 Visual Studio 2008 起始页中，选择“New Project”，并创建一个具有以下值的项目：
   - 项目类型：Visual C# 项目
   - 模板：控制台应用程序
   - Name:WSAPIDemo
   - 位置：C:\wsapi

   此操作将创建一个包含某些文件的新目录 C:\wsapi\WSAPIDemo。
4. 在新项目中，从对象列表中删除默认文件 Program 和 AssemblyInfo.cs。
5. 将您在步骤 1 中获取的文件复制到 C:\wsapi\WSAPIDemo 目录。
6. 从获取 C# 客户端存根 (第 172 页)中指定的 URL 下载客户端存根。
7. 将 C# 客户端存根复制到 C:\wsapi\WSAPIDemo 目录。
8. 将在上述两个步骤中复制的文件添加到 WSAPIDemo 项目：
   - 在 Visual Studio 中，从“Project”菜单中选择“Add Existing Item”。
   - 浏览至目录 C:\wsapi\WSAPIDemo，并选择所有的演示文件(.cs 和 .config)。
9. 向 System.Web.Services.dll 添加引用：
   - 在 Visual Studio 中，从“Project”菜单中选择“Add Reference”。
   - 在 .NET 标记下，浏览到具有以下名称的组件：System.Web.Services.dll。
   - 依次单击 System.Web.Services.dll、“Select”和“OK”。
10. 如果在创建项目时使用了不同模板，则可能需要将引用添加到 System、System.XML 和 System.Data。检查项目引用以确定是否需要添加这些引用。
11. 在 App.config 文件中，更改 username、password、host 以及诸如 serverID 等硬编码对象 ID 的值。
12. 在 Visual Studio 中，从“Build”菜单中选择“Build WSAPIDemo”。

> **警告：** 如果目标核心正在运行 TLSv1.x 的最低协议版本，则 PowerShell 版本(绑定的基础 NET Framework 版本)必须支持它。有关详细信息，请参见

> https://msdn.microsoft.com/en-us/library/system.net.securityprotocoltype(v=vs.110).aspx
>
> 此外，它必须显式从 C# 应用程序中启用。
>
> 代码示例:System.Net.ServicePointManager.SecurityProtocol |=
> System.Net.SecurityProtocolType.Tls12 | System.Net.SecurityProtocolType.Tls11;"

运行 C# 演示程序

要运行演示程序，请执行以下操作:

1. 打开 Visual Studio 2008 命令提示符：

   “开始”>“所有程序”>“Microsoft Visual Studio 2008”>
   “Visual Studio Tools”>“Visual Studio 2008 Command Prompt”
2. 将目录更改为：

   ```
   C:\wsapi\WSAPIDemo\bin\Debug
   ```
3. 输入以下命令：

   ```
   WSAPIDemo.exe
   ```

C# 示例代码

下面的代码片段摘自 WebServicesDemo.cs (包含在之前下载的 ZIP 文件中的一个 C# 程序)。

设置证书处理方式

```
# This setup is required just once for the client.
#
ServicePointManager.CertificatePolicy = new MyCertificateValidation();
```

分配 URL 前缀

```
# This is the URL prefix for all services.
#
wsdlUrlPrefix = protocol + "://"+ host + ":"+ port + "/" + contextUri + "/";
```

启动服务

```
FolderService fs = new FolderService();
fs.Url = wsdlUrlPrefix + "com.opsware.folder/FolderService";
```

调用服务方法

```
FolderRef root = fs.getRoot();
FolderVO vo = fs.getFolderVO(root);
```

处理接口返回类型

```
            # In the API, FolderVO.getMembers returns an array of
# FNodeReference interfaces, but Web Services does not support
# interfaces.In the C# stub, the return type of
# FolderVO.members is Object[].If a returned Object type will
# be used as a parameter that must be a specific type, then you
# must cast it to that type.For example, the following code
# casts elements of the returned array to FolderRef as
# appropriate.
#
Object[] members = vo.members;
for(int i=0;i<members.Length;i++)
{
Console.WriteLine("Got object:" + members[i].GetType().FullName + " --> " +
((ObjRef)members[i]).name);
       if(members[i] is FolderRef) {
       Console.WriteLine("I am a FolderRef:" +
       ((FolderRef)members[i]).name);
       }
}
```

更新或创建 VO

```
            # When updating a VO, the changed attributes must be set in
# dirtyAttributes.(The VO passed to a create method has
# the same requirement.)
#
# Note:If you update a VO that was returned from a service
# method invocation, such as getFolderVO, then you must
# set the logChange attribute of the VO to false:
# vo.logChange = false;
#
# The following code changes the name of a folder.
#
Console.WriteLine("Changing name from " + vo.name +
" to yo_csharp.");
vo.name = "yo_csharp";
vo.dirtyAttributes = new String[]{"name"};
# Manually set dirty fields being changed.
#
```

```
vo = fs.update(folder, vo, true, true);
Console.WriteLine("Folder name changed to:" + vo.name);
```

处理异常

```
            # .NET converts Web Services faults into SoapExceptions
# without trying to deserialize them into application
# exceptions first.As a result, your code cannot catch
# application exceptions.As a workaround, the C# stubs
# provided by SA include SOAPExceptionParser,
# a class that enables you to get information from
# SOAPExceptions.The following code shows how to get the
# exception name and error message by calling the getDetail
# method of SOAPExceptionParser.
#
try{
// Try to get a non-existent folder here.
} catch(SoapException e){
       SoapExceptionDetail detail =
       SoapExceptionParser.getDetail(e);
       Console.WriteLine("SoapExceptionDetail.name:" +
       detail.exceptionName);
       Console.WriteLine("SoapExceptionDetail.msg:" +
       detail.message);
...
}
```

使用 C# 时的密码安全性

FolderService 方法将从 App.config 文件读取用户和密码对。下面显示了该方法的示例。

```
            User user = new User();
user.username = "user";
user.password = "password";
FolderService fs = new FolderService();
fs.Url = wsdlUrlPrefix + "com.opsware.folder/FolderService";
fs.user = user;
```

如果您不希望在 App.config 文件中以明文形式存储密码，可以使用 SecureUser 类对密码进行加密。SecureUser 类使用 .NET 2.0 中的 C# SecureString。密码将以加密形式存储在 SecureString 中。此外，getPassword() 仅在内部可见。SecureUser 是一个静态类，因此您只需一次性设置用户名和密码，或在每次切换用户时进行设置。

为了向后兼容，每个服务首先会从 SecureUser 检索用户名和密码，然后从其用户成员变量检索，最后从 App.config 中检索。SecureUser 将获取密码的 String 或 SecureString。在任一种情况下，客户端都负责清理传递给 SecureUser.setUser() 方法的密码变量。

在某一时刻，密码需在内存中转换为常规 C# 字符串，该字符串将一直存在于内存中，直到下一次执行垃圾回收操作。如果使用 SecureUser，将只能确保内部密码存储是安全的。

以下示例显示了如何安全地设置用户名和密码。

```
            SecureString passwd = new SecureString();
passwd.AppendChar('p');
passwd.AppendChar('a');
passwd.AppendChar('s');
passwd.AppendChar('s');
passwd.AppendChar('w');
passwd.AppendChar('d');
SecureUser.setUser("username", passwd); // that's it, no need to set up user
for each service.
passwd.Dispose(); // resets passwd and frees up memory so no copy remains from
caller.
```

可插入检查

SA 审核和修正功能允许您定义和监控 SA 托管服务器的符合性信息。由于符合性标准在不断发展，因此 SA 允许您创建专门的自定义检查和策略，以及扩展已在 SA 中提供的检查和策略。可插入检查是审核规则，属于一个或多个审核策略。可以使用 SA 客户端在命令行环境中创建可插入检查、上载该检查，然后将它添加到审核策略。

本节的目标读者包括熟悉 XML 以及 SA 的审核和修正功能的软件开发人员。

可插入检查安装

在开发可插入检查之前:

1. 在开发环境中安装 SA 核心。请不要使用生产核心。
2. 在已安装代理的服务器上，安装 OCLI 1.0。有关 OCLI 1.0 的信息，请参见使用。

可插入检查教程

本教程演示了如何创建名为 HelloWorld Check 的可插入检查。该简单检查将验证 /var/tmp/helloworld 文件是否存在于 Unix 托管服务器上。如果该文件不存在，则此可插入检查的修正脚本将创建该文件。

要开发 HelloWorld Check，请执行以下操作：

1. 按照可插入检查安装中的说明执行操作。在本教程中，安装了 OCLI 1.0 的服务器将充当开发服务器。
2. HelloWorld Check 示例代码包含在含有 API 代码示例的 ZIP 文件中。
3. 压缩在之前步骤中下载的文件，并验证 pluggable_checks/helloworld 目录中是否包含以下文件：
   - config.xml
   - gethelloworld.py
   - sethelloworld.py

   HelloWorld Check 由下列三个文件组成。config.xml 文件是配置文件。gethelloworld.py Python 脚本执行审核。sethelloworld.py Python 脚本执行修正。在下面的步骤中，会将这些文件打包成一个 ZIP 文件，然后将该 ZIP 文件导入 SA。
4. 在开发服务器上，将解压缩后的 helloworld 文件复制到工作目录，例如：

   ```
   cd /home/jdoe/dev
   mkdir helloworld
   cd helloworld
   cp unzip_dest/pluggable_checks/helloworld/* .
   ```
5. 获取全局唯一 ID (GUID)。每个可插入检查都需要一个 GUID。可以使用下列方法之一获取有效的 GUID：
   - 登录网站，如下列网站：
     http://kruithof.xs4all.nl/uuid/uuidgen
   - 从以下网站下载免费的 Windows 工具 guidgen：
     http://www.microsoft.com/downloads/details.aspx?FamilyID=94551F58-484F-4A8C-BB39-ADB270833AFC&displaylang=en

如果您以编程方式创建 GUID，则代码应遵循 RFC4122 (http://www.ietf.org/rfc/rfc4122.txt)。

6. 使用文本编辑器在 config.xml 文件中插入 GUID，例如：

```
<checkGUID>6c7ed38c-d8d6-11db-8314-0800200c9a66</checkGUID>
```

这是本教程中需要在 config.xml 文件中修改的唯一元素。

7. 在文本编辑器中，在 config.xml 中保存对 GUID 的更改。

将文本编辑器保持打开状态。在本教程中，将检查 config.xml 中的各个元素，了解它们是如何映射到 HelloWorld Check 的 Python 脚本和 SA 客户端显示字段的。

8. 在 config.xml 文件中，注意下列与 HelloWorld Check 中的审核 (get) 和修正 (set) 脚本相关的元素：

```
<!-- The name of the script that performs the check.-->
<checkGetScriptName>gethelloworld.py</checkGetScriptName>

<!-- The name of the script that remediates the audit.-->
<checkSetScriptName>sethelloworld.py</checkSetScriptName>

<!-- The exit code of the gethelloworld.py script will be checked.-->
<checkReturnType>EXITCODE</checkReturnType>

<!-- A string argument is passed to gethelloworld.py.-->
<checkGetArgumentType>STRING</checkGetArgumentType>

<!-- The default argument for gethelloworld.py is the name of the file the
script is checking for.-->
<checkGetArgumentDefaultValue>/var/tmp/helloworld
</checkGetArgumentDefaultValue>

<!-- If the helloworld file exists, the exit code of gethelloworld.py is 0.
-->
<checkSuccessExitCodeValue>0</checkSuccessExitCodeValue>
<!-- If the helloworld file does not exist, the exit code of
gethelloworld.py is 1.-->
<checkSuccessExitCodeValue>1</checkSuccessExitCodeValue>
```

9. 检查 gethelloworld.py 脚本，该脚本通过检查 /var/tmp/helloworld 是否存在来执行审核。在本教程中，无需编辑该脚本。在本教程的后面 (步骤 30)，当您在 SA 客户端中运行审核时，该脚本会在托管服务器上执行。

/var/tmp/helloworld 字符串是脚本的默认参数，由 config.xml 中 <checkGetArgumentDefaultValue> 的值表示。该脚本的退出代码 (result) 对应于为 <checkSuccessExitCodes> 指定的值。

下面是 gethelloworld.py 脚本的源代码：

```
import sys
import os
import string
if __name__ == "__main__":
       if len(sys.argv) != 2:
               sys.stderr.write("No argument found!Please enter a
                       file name!\n")
               sys.exit(220)
filename = sys.argv[1]
if os.path.isfile(filename) or os.path.isdir(filename):
       result = 0
else:
       result = 1
sys.stderr.write("Debugging:Found result %s\n"
% result)
sys.stdout.write("%s\n" % result)
sys.exit(result)
```

10. 然后，检查修正脚本 sethelloworld.py，该脚本将创建 /var/tmp/helloworld 文件。如果确定要在步骤 35 中修正审核，该脚本将在托管服务器上运行。在本教程中，将不更改该脚本。

sethelloworld.py 的源代码如下：

```
import sys
import os
import string
if __name__ == "__main__":
       if len(sys.argv) != 2:

                sys.stderr.write("No argument found!
                       Please enter a file name!\n")
               sys.exit(220)
filename = sys.argv[1]
if os.path.isfile(filename) or os.path.isdir(filename):
       # Do nothing because the file already exists.
       pass
else:
       try:
               fd = open(filename, "w")
```

```
                fd.write(" ")
                fd.close()
        except:
                sys.stderr.write("Could not open file %s for
                        writing!\n" % filename)
                sys.exit(220)
# Exit successfully with a 0 exit code.
sys.stderr.write("Successfully created file\n")
sys.exit(0)
```

11. 将 HelloWorld Check 打包。

    要将 HelloWorld 可插入检查打包，请将工作目录内容存档到一个 ZIP 文件中，例如：

    ```
    cd /home/jdoe/dev/helloworld
    zip ../helloworld.zip *
    ```

12. 通过输入下列 unzip 命令，验证 ZIP 文件是否包含两个 Python 脚本和 config.xml 文件：

    ```
    unzip -t ../helloworld.zip
    testing: config.xml OK
    testing: gethelloworld.py OK
    testing: sethelloworld.py OK
    No errors detected in compressed data of ../helloworld.zip.
    ```

13. 使用 OCLI 1.0 的 ouplоad 命令将可插入检查导入 SA：

    ```
    ouplоad -C"Customer Independent" \
    -t"Server Configuration Check" \
    --forceoverwrite --old -O"SunOS 5.8" ../helloworld.zip
    ```

    > **备注：** 对于所有 Unix 和 Linux 检查，平台选项 (-O) 为 SunOS 5.8。对于 Windows 检查，平台选项为 Windows 2003。

    如果 ouplоad 运行失败，请确保您已安装正确版本的 OCLI 1.0、正确设置 PATH 环境变量，并已将 login 文件包括到环境中。有关这些要求的详细信息，请参见使用中的 OCLI 1.0。

14. 打开 SA 客户端。

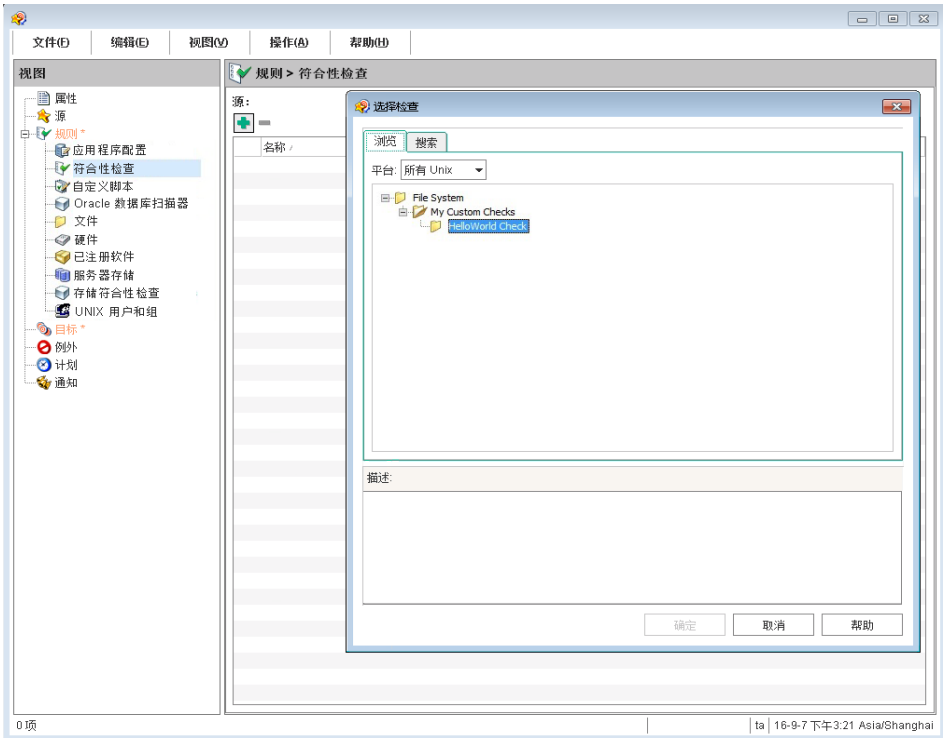
    在接下来的几个步骤中，您将创建新审核，将其添加到使用 ouplоad 命令导入的 HelloWorld Check 中。

15. 在“工具”菜单中，选择“更新缓存”。

16. 从“导航”页面中，选择“库”**>**“按类型”**>**“审核和修正”**>**“审核”**>**“Unix”。

17. 在“操作”菜单中，选择“新建”。
18. 在“审核”窗口“属性”窗格的“名称”字段中，输入“HelloWorld Audit”。
19. 在“视图”窗格中，选择“规则”>“符合性检查”。
20. 单击“添加”按钮，然后单击“文件系统”。
    “内容”窗格将在“可用于审核”下列出 HelloWorld Check，如下图所示。

文件系统规则中的 HelloWorld Check



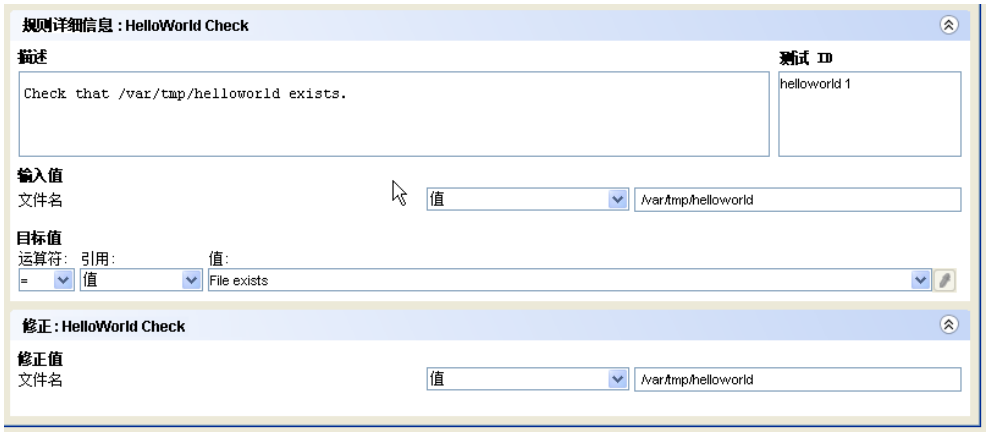
21. 在 config.xml 文件中，注意下列与所示信息相关的元素：

```
<!-- The check name is the rule name shown in the SA Client.-->
<checkName>HelloWorld Check</checkName>

<!-- The category corresponds to the rule hierarchy displayed by the SA
Client.-->
<checkCategory>File System|My Custom Checks</checkCategory>
```

在 SA 客户端的“审核”窗口的“可用于审核”下，选择“HelloWorld Check”，然后单击加号。

“内容”窗格将列出 HelloWorld Check 的详细信息，如下图所示。

HelloWorld Check 规则详细信息



22. 在 config.xml 文件中，检查下列与 HelloWorld Check 规则详细信息 (第 184 页)下显示的信息相关的元素:

```
<!-- The following value appears under Description in the Rule Details of
the SA Client.-->
<checkDefaultDescription>
Check that /var/tmp/helloworld exists.
</checkDefaultDescription>

<!-- The following element corresponds to the Test ID in the SA Client.-->
<checkTestID>helloworld 1</checkTestID>

<!-- This label is under Input Values in the SA Client.-->
<checkGetArgumentDefaultLabel>File Name
</checkGetArgumentDefaultLabel>

<!-- The default argument to the gethelloworld.py script also appears
under Input Values in the SA Client.-->
<checkGetArgumentDefaultValue>/var/tmp/helloworld
</checkGetArgumentDefaultValue>
```

23. 在 SA 客户端的“视图”窗格中选择“目标”。
24. 在以下步骤中，将向 HelloWorld Audit 中添加目标服务器。在其后的步骤中，gethelloworld.py 和 sethelloworld.py 脚本将在目标服务器上运行。
25. 在“内容”窗格中，单击“添加”。
26. 在“选择服务器”窗口中，向下钻取到服务器，然后单击“确定”。
    在“审核”窗口中，选择“文件”>“保存”。
27. 此时，HelloWorld Audit 将包含 HelloWorld Check(规则)，并与目标服务器关联。

28. 在“审核”窗口中，从“操作”菜单选择“运行审核”。
29. 在“运行审核”任务的各个窗口中完成操作。
30. 在“运行审核”窗口中，单击“启动作业”。
    此操作将在目标服务器上启动运行 gethelloworld.py 脚本的作业。
31. 在该作业完成后，单击“查看结果”。
32. 在“审核结果”窗口的“视图”窗格中，选择“策略规则(1)”。
33. 在“审核结果”窗口的“内容”窗格中，打开“HelloWorld Check”。

    此时将显示“差异详细信息”窗口，如下图所示。

    HelloWorld Check 差异详细信息

34. 在 config.xml 文件中，注意下列与 HelloWorld Check 差异详细信息 (第 186 页)中所示信息相关的元素:

```
<!-- The following value appears as the Policy Value in the Difference
Details window.-->
<checkSuccessExitCodeDefaultDisplayName>
File exists</checkSuccessExitCodeDefaultDisplayName>


<!-- The next value appears as the Actual Value in the same window.-->
<checkSuccessExitCodeDefaultDisplayName>
File does not exist</checkSuccessExitCodeDefaultDisplayName>
```

35. 如果要在目标服务器上创建 /var/tmp/helloworld，则在“差异”窗口中单击“修正”。

    该操作将运行 sethelloworld.py 脚本。有关详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

审核和修正

Sarbanes-Oxley (SoX)、IT 基础设施库 (ITIL) 和 ISO20000 要求服务器配置必须符合规定。SA 的审核和修正功能提供了一组结构清晰的策略，帮助您解决符合性问题。而通过图形界面，您可以轻松针对指定服务器选择和运行审核，以及查看服务器与各种专业标准的符合情况。

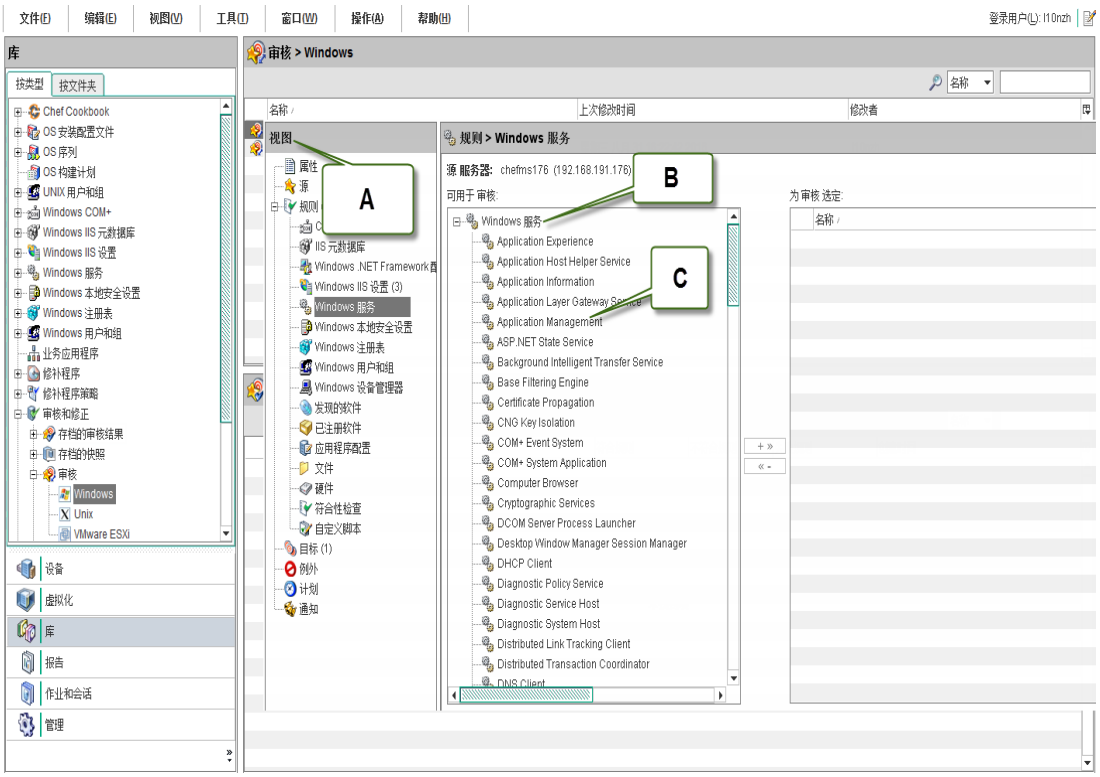
审核和修正功能还简化了系统管理任务。例如，您可以监控一类运行由自己团队所构建应用程序(例如数据库服务器或中间件应用程序)的服务器。当配置和监控运行该类应用程序的服务器时，您需要一个清单来跟踪配置的理想状态。该清单可包括文件、目录和网络共享权限。

您可以创建用来定义这些配置的审核，然后在安装应用程序后审核服务器。审核结果将确认应用程序是否已安装，以及是否已根据条件成功配置。如果配置不符合条件，可以创建临时审核，用于解决问题。如果审核结果表明发生了错误，则可以修正服务器，以实现可满足要求的配置。为了确保所做配置变更能够在生产环境中正常工作，可以将审核设置为按可配置的计划运行，并在完成时发出通知。

下图中显示了用于选择审核的窗口，其中包括以下标注:

- **标注 A**：在“视图”面板中列出的任何类别可能具有 SA 无法修改的功能，或者可修改的可插入检查。
- **标注 B**：指向用于处理 Windows 服务的 SA 功能。
- **标注 C**：列出用于处理 Windows 服务的可插入检查。

Windows 服务审核规则



每项检查评估一个规则。可以将多项检查捆绑到一个策略中。

SA 审核和修正功能附带了许多预置检查。通过选择所需检查，可以运行大部分审核。随着开发人员设计、编码、测试以及通过 HPE Live Network 向系统添加更多的检查，审核选项将持续增多。这些检查作为完整策略导入。

然而，由于每个业务都具有独特挑战和独有资源，因此您可能需要根据 SA 审核和修正框架中一组尚不存在的审核条件来确定符合性。为此，系统提供了一种方法来创建自定义可插入检查。

审核和修正功能将根据特定规则来评估 SA 所管理的服务器的符合性状态。此功能还可以修正与规则中定义的期望配置状态不一致的服务器。这些规则包括各种服务器参数、注册表值、文件权限、应用程序配置、文件存在性、COM+ 对象，等等。

在 Windows 环境中，Web 服务器规则也使用应用程序配置指定，基于 Microsoft Internet Information Services (IIS) Web 服务器配置文件 UrlScan.ini。SA 可以比较指定配置文件的部分或全部值，从文件选择所需元素并确保这些值或配置文件条目存在。有关详细信息，请参见应用程序配置。

SA 包含许多预先设计的审核规则。每个规则都定义了服务器或服务器组的期望配置状态。一些规则基于值，提供了比较运算符(、、==、!=、contains 等)、一个或一组值以及一个或多个检查，用来拼写出评估审核的项目状态所用的基础代码。比较数据将确定是否符合条件。如果检查支持修正操作，则规则也可能包含修正值。

规则由单个检查组成。通过以可插入检查形式来使用自定义内容对象，您可以创建新功能。为方便起见，还可以将多个相关的可插入检查捆绑到审核策略中。

创建可插入检查

可插入检查是下载到一个或多个托管服务器并由审核和修正框架执行的代码。可以使用检查来扩展本地审核和修正属性，以及提供其他专用功能。每个可插入检查都包括一个自定义的 config.xml 文件以及至少一个脚本，用于对照该 config.xml 文件中指定的值来比较审核功能。可插入检查还可包括一个脚本，用于将审核的服务器中的指定变量设置为 config.xml 文件中指定的值。您可以在 Python、Visual Basic 脚本 (VBS)、BAT 或 shell 脚本中编写可插入检查。可插入检查将打包为 zip 存档。

大多数 CIS 检查是 CIS 基准的直接转换。有关详细信息，请访问 http://www.cisecurity.org。

大多数类型的检查可划分为以下类别之一：

- Windows 注册表检查
- Unix 服务检查
- 用户检查，这些检查可使用密码或影射文件信息

可插入检查准则

为了简化服务器维护操作，请遵循以下准则：

- 在创建新的可插入检查时，应特别留意名称。名称应描述检查目的，并用下划线代替空格。例如，`Users_Without_Password_Expiration` 就一目了然。在服务器已获取数百个或更多检查的情况下，这可帮助您快速找到某个检查。
- 编写通用检查。这样，只需更改几行代码就能轻松创建更多执行类型相同的检查。例如，对于大多数 CIS2k3 Windows 服务检查，只需更改一行代码即可为新服务创建新检查。
- 在命名审核 (get) 和修正 (set) 脚本时，请从目录名称中删除空格或下划线，并相应地加上前缀 get 或 set。例如，`getUsersWithoutPasswordExpiration.sh` 就是一个很好的审核文件名称。即使认为您的自定义检查不会由其他人使用，也请遵循该准则。
- 注意错误检查。请记住，脚本失败时，意外的返回值可能会将审核报告为不符合条件。请捕获意外的错误或异常，并将相关信息写入到 stdout 或 stderr 以简化故障排除操作。
- 在可能的情况下，将大多数检查转换为简单的二进制 True 或 False 结构。

- 务必始终尝试同时处理特定基准情形及其对应情形。例如，您可以在创建“Enable Service X”可插入检查的同时，轻松地创建“Disable Service X”可插入检查，并重用大多数代码。如果您决定以后测试相反条件，这将十分有用。
- 尽可能使用框架所定义的标准退出代码。这些退出代码为：

```
EXIT_FAILURE=220
EXIT_ERR_USAGE=221
EXIT_ERR_INVALID_OS=222
```

- 在布尔类型检查中返回已禁用或已启用时，返回 0 表示已禁用，返回 1 表示已启用。
- 将所有可插入检查打包为 ZIP 存档。单个文件系统目录中包含表所列出的文件。

可插入检查内容

文件名	描述
config.xml	(必需)用于定义该可插入检查如何执行、返回及最终报告符合或不符合条件的 XML 配置文件。
getName.{py | sh | BAT | vbs}	(必需)以 Python、VBS、BAT 或 shell 编写的审核脚本，用于评估审核对象，并根据 config.xml 中的定义返回文本和退出代码。
setName.{py | sh | BAT | vbs}	(可选)以 Python、VBS、BAT 或 shell 编写的修正脚本，用于修正审核脚本所检查的条件。
其他代码、脚本或库	(可选)由审核或修正脚本使用的帮助程序和补充脚本。

审核和修正脚本的文件名不必以 get 和 set 开头，但是该命名约定可简化文件维护工作。

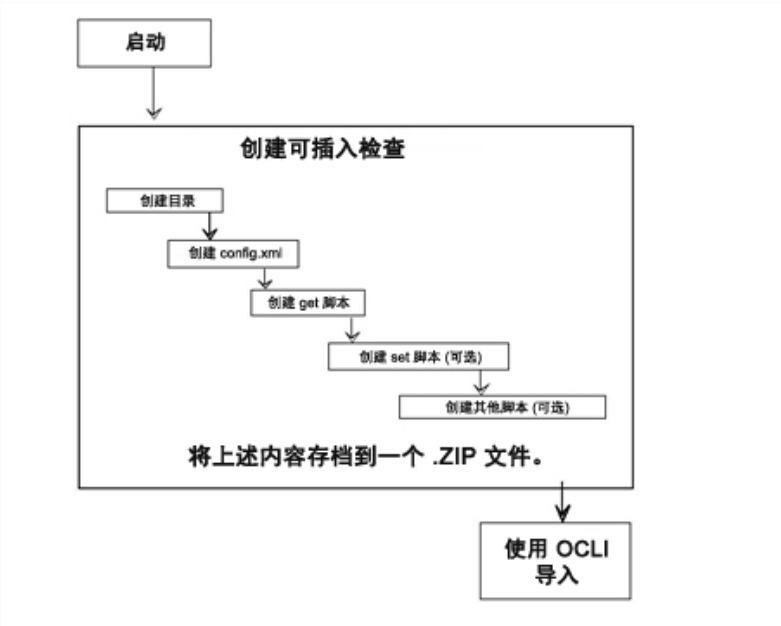
以下示例显示了可插入检查的目录结构：

```
./check_name/
./check_name/config.xml
./check_name/getcheckname.py
./check_name/setcheckname.py
```

可插入检查的开发流程

下表概述了在命令行环境中执行的开发流程。

开发流程



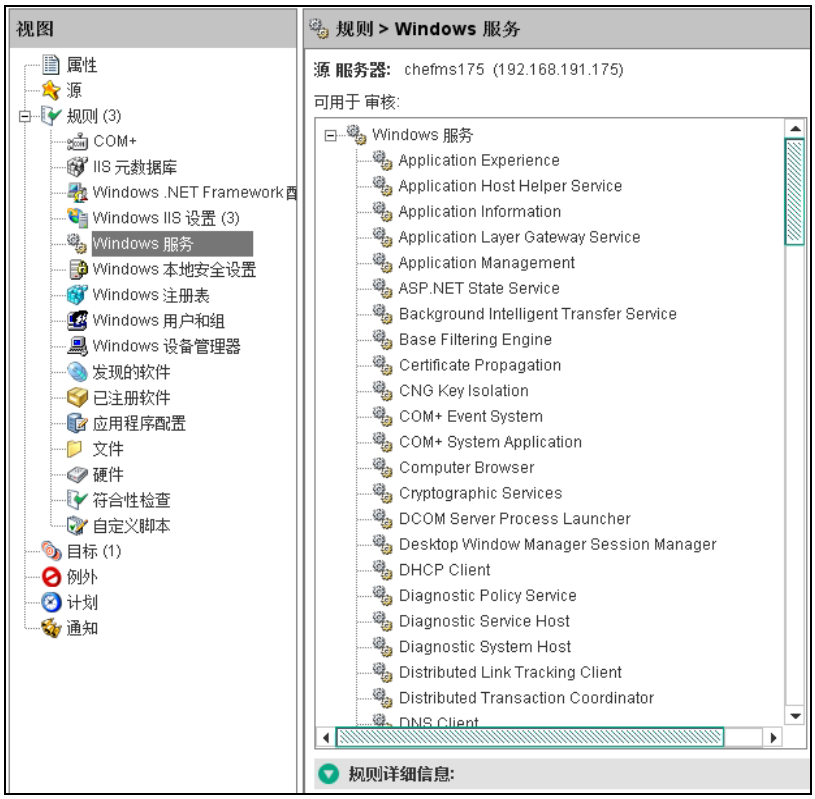
可插入检查配置 (config.xml)

config.xml 文件是可插入检查的规范文件，其中包含一些元素，用于控制该检查在 SA 客户端中的显示方式、默认值、要比较的值类型以及检查类别。例如，config.xml 文件中的以下元素确定 SA 客户端中的可插入检查的规则类别：

```
<checkCategory>Windows Services</checkCategory>
```

每个标准类别由其相应图标表示。标准类别包括硬件、软件、操作系统、用户和组、文件系统等，如下图所示。

规则层次结构中的可插入检查类别



以下列表显示了 config.xml 文件的模板：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE checkConfiguration SYSTEM "check.dtd">
<checkConfiguration version="1.0">
<checkName>$CHECKNAME</checkName>
<checkGUID>$CHECKGUID</checkGUID>
<checkDefaultDescription>$CHECKDESCRIPTION</checkDefaultDescription>
<checkRemediationDefaultDescription> $CHECKREMEDIATIONDESCRIPTION </
checkRemediationDefaultDescription>
<checkGetScriptName>$GETSCRIPTNAME</checkGetScriptName>
<checkGetScriptType>PY</checkGetScriptType><!-- Or SH for shell, BAT for Bat,
VBS for Visual Basic -->
<checkSetScriptName>$SETSCRIPTNAME</checkSetScriptName><!-- Optional -->
<checkSetScriptType>PY</checkSetScriptType><!-- Optional -->
<checkVersion>32b.0-1.0</checkVersion>
<checkReturnType>$RETURNTYPE</checkReturnType> <!-- EXITCODE, STRING, or
NUMBER -->
<checkTestIDs>
<checkTestID>$CHECKTESTID</checkTestID> <!-- Optional -->
</checkTestIDs>
<checkPlatformTypes>
<checkPlatform>$PLATFORMTYPE</checkPlatform> <!-- Currently Unix or Windows --
>
</checkPlatformTypes>
```

```
<checkCategories>
<checkCategory>$CATEGORY</checkCategory> <!-- Top-level GUI category -->
</checkCategories>
<checkGetArguments> <!-- All arguments are optional -->
<checkGetArgument>
<checkGetArgumentType>$GETARGTYPE</checkGetArgumentType> <!-- STRING or NUMBER
-->
       <checkGetArgumentDefaultLabel>$GETDEFAULTLABEL</
checkGetArgumentDefaultLabel>
               <checkGetArgumentDefaultDescription>$GETDEFAULTDESCRIPTION</
checkGetArgumentDefaultDescription>
               <checkGetArgumentDefaultValue>$GETDEFAULTVALUE</
checkGetArgumentDefaultValue>
       </checkGetArgument>
</checkGetArguments>
<checkSetArguments> <!-- Also optional -->
<checkSetArgument>
<checkSetArgumentType>$SETARGTYPE</checkSetArgumentType>
       <checkSetArgumentDefaultLabel>$SETDEFAULTLABEL</
checkSetArgumentDefaultLabel>
               <checkSetArgumentDefaultDescription>$SETDEFAULTDESCRIPTION</
checkSetArgumentDefaultDescription>
       <checkSetArgumentDefaultValue>$SETDEFAULTVALUE</
checkSetArgumentDefaultValue>
</checkSetArgument>
</checkSetArguments>
<checkSuccessExitCodes> <!-- Only for EXITCODE type checks, generally at least
two entries -->
       <checkSuccessExitCode>
<checkSuccessExitCodeValue>$EXITCODEVALUE</checkSuccessExitCodeValue>
               <checkSuccessExitCodeDefaultDescription>$EXITCODEDESCRIPTION
       </checkSuccessExitCodeDefaultDescription>
               <checkSuccessExitCodeDefaultDisplayName>$EXITCODEDISPLAYNAME
       </checkSuccessExitCodeDefaultDisplayName>
</checkSuccessExitCode>
</checkSuccessExitCodes>
</checkConfiguration>
```

有关更多详细信息，请参见config.xml 文件的文档类型定义 (DTD)。

审核 (get) 脚本

您可以设计审核脚本(也称为 get 脚本)，以从托管服务器中获取值。根据 config.xml 文件中的指定，可使用一些可选参数来执行该脚本。如果脚本运行 EXITCODE 检查，则会将脚

本结果与 config.xml 文件中指定的退出代码进行比较。对于 STRING 和 NUMBER 返回类型的检查，会将结果与写入 STDOUT 的内容进行比较。

审核脚本具有一组预定义的返回代码。可以在检查的 config.xml 文件中定义其他返回代码。

审核脚本可显示参考消息。在对审核脚本失败进行故障排除时，这些消息将很有用。请查看下面的 Python 审核脚本示例：

修正 (set) 脚本

您可以设计修正脚本(也称为 set 脚本)，以制定托管服务器更改，该更改完成时审核脚本将返回成功。根据检查的 config.xml 文件中的指定，可使用一些可选参数来执行该脚本。

这些 set 脚本是可选的，并且可以具有与其对应 get 脚本非常相似的特性，也可以具有与对应 get 脚本完全不同(并且更长)的特性。

从 shell 的角度来看，除了所使用的返回代码之外，脚本本身并无特殊之处。大多数检查会显示一些调试输出或信息消息。除了发生脚本失败时(这些消息可用于执行故障排除)，用户通常不会看到这些信息。

标准实践是始终在 set 脚本中至少包含一个参数。此外，请记住修改 config.xml 文件，以便在向现有检查添加 set 脚本时，该文件能够正常显示在 SA 客户端中。

确保在修正脚本退出时返回退出代码 0，这表示成功。所有其他退出代码将表示修正操作失败。

请查看下面的 Python set 脚本示例。

```
import sys
import os
import string
if __name__ == "__main__":
#If there are set arguments they will be loaded into
# sys.argv
# Enter the desired set code here.Stdout may be used for
# debugging.
#Uses exitcode 0 for success, and all other values for
# failure.
# enter condition where set script if successful. for this
# example, use ‘if 1’
if 1:
       sys.exit(0)
else:
       sys.exit(-1)
```

可插入检查中的其他代码

除 get 或 set 脚本以外，可插入检查还可包含其他代码。可以向检查中添加库、可执行文件或其他脚本，以使其 set 或 get 脚本能够在执行时利用这些代码。

您还可以在 ZIP 文件中包含其他代码。

压缩可插入检查

在创建 config.xml 文件之后，审核 (get) 脚本和可选的修正 (set) 脚本将创建一个包含这些文件的 ZIP 存档。下面的 shell 历史记录显示了 UNIX 环境中的创建过程。

```
# ls
check_name
# cd check_name
# zip ../checkname.zip *
adding: config.xml
adding: getcheckname.py
adding: setcheckname.py
# unzip -t ../checkname.zip
testing: config.xml OK
testing: getcheckname.py OK
testing: setcheckname.py OK
No errors detected in compressed data of ../checkname.zip.
```

导入可插入检查

可使用 OCLI 1.0 实用程序将可插入检查导入 SA 核心或网状网络。SA 内容实用程序中说明了该实用程序。下面的 shell 历史记录提供了适用于 Linux 的导入过程示例：

```
# cp checkname.zip /var/tmp/checks
# cd /var/tmp/checks
# cp opsware_32.a.692.0-upload/disk001/packages/Linux/3AS/ocli-32a.2.0.5-
linux-3AS .
# chmod 755 ocli-32a.2.0.5-linux-3AS
# ./ocli-32a.2.0.5-linux-3AS
# ../ocli/login.sh
# export PATH=/opt/opsware/bin:$PATH
# oupload -C"Customer Independent" -t"Server Configuration Check" --
forceoverwrite --old -O"SunOS 5.8" your_Pluggable_check.zip
```

`ouploa`d 命令使用 "SunOS 5.8" 将检查定义为属于 SA 客户端中的通用 Unix 类别。要指定 Windows 类别的检查，请使用“Windows 2003”。

创建审核策略

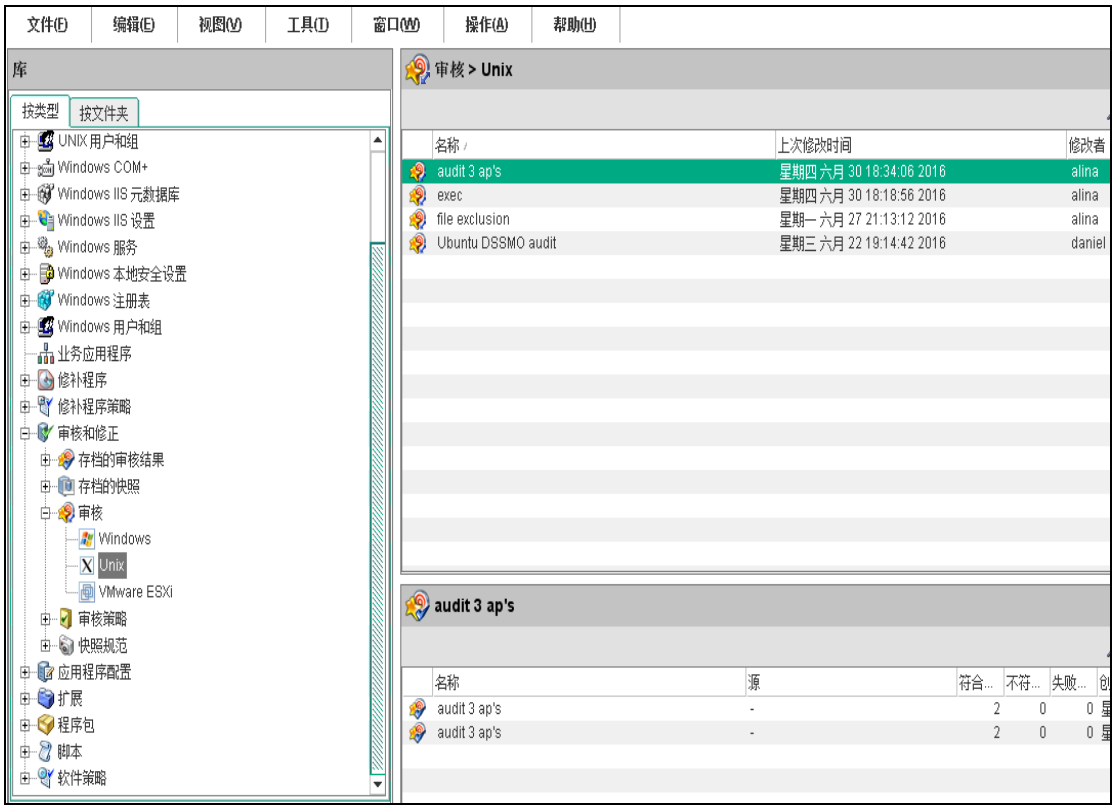
下图显示了审核策略创建过程:

创建审核策略的过程

创建审核策略

审核策略由规则组成。每个规则由一个或多个检查组成，其中可以包括用户创建的可插入检查。在 SA 客户端中显示、创建和编辑审核策略及规则。下图显示在模型系统上可用的审核规则的列表。

审核规则列表



有关创建审核策略的详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

导出审核策略

要将新审核策略移动到其他 SA 核心，请使用 DCML 交换工具 (DET) 命令行实用程序将其从一个核心导出，然后导入另一个核心。可通过此工具使用现有核心的内容(如策略)来填充新安装的 SA 核心。有关此过程的详细说明，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

config.xml 文件的文档类型定义 (DTD)

该文件管理 SA 客户端的显示名称和描述、默认值、要对检查代码返回的值执行的比较、显示这些值的 SA 客户端类别，等等。

默认 config.xml 文件中的两个元素 checkGetArguments 和 checkSetArguments 用于在执行时向脚本传递数据值。如果您的可编程检查不需要任何参数，请从 config.xml 文件中删除这些元素。

config.xml 的以下 DTD 由 SA 动态生成：

```
<!ELEMENT checkConfiguration (checkName, checkGUID, checkDefaultDescription,
checkRemediationDefaultDescription?, checkGetScriptName?,
checkGetScriptType?, checkSetScriptName?, checkSetScriptType?, checkVersion,
checkAllowRemediationOnFailure?, checkReturnType, checkTestIDs?,
checkPlatformTypes, checkExclusivePlatforms?, checkExcludePlatforms?,
checkCategories, checkGetArguments?, checkSetArguments?,
checkComparisonDefaults?, checkCompareValidValues?, checkSuccessExitCodes?)>
<!ATTLIST checkConfiguration version CDATA #REQUIRED>
<!ELEMENT checkName (#PCDATA)>
<!ELEMENT checkGUID (#PCDATA)>
<!ELEMENT checkDefaultDescription (#PCDATA)>
<!ELEMENT checkRemediationDefaultDescription (#PCDATA)>
<!ELEMENT checkGetScriptName (#PCDATA)>
<!ELEMENT checkGetScriptType (#PCDATA)>
<!ELEMENT checkSetScriptName (#PCDATA)>
<!ELEMENT checkSetScriptType (#PCDATA)>
<!ELEMENT checkVersion (#PCDATA)>
<!ELEMENT checkAllowRemediationOnFailure (#PCDATA)>
<!ELEMENT checkReturnType (#PCDATA)>
<!ELEMENT checkTestIDs (checkTestID+)>
<!ELEMENT checkTestID (#PCDATA)>
<!ELEMENT checkPlatformTypes (checkPlatform+)>
<!ELEMENT checkPlatform (#PCDATA)>
<!ELEMENT checkExclusivePlatforms (checkExclusivePlatform+)>
<!ELEMENT checkExclusivePlatform (#PCDATA)>
<!ELEMENT checkExcludePlatforms (checkExcludePlatform+)>
<!ELEMENT checkExcludePlatform (#PCDATA)>
<!ELEMENT checkCategories (checkCategory+)>
<!ELEMENT checkCategory (#PCDATA)>
<!ELEMENT checkGetArguments (checkGetArgument+)>
<!ELEMENT checkGetArgument (checkGetArgumentType,
checkGetArgumentDefaultLabel, checkGetArgumentDefaultDescription,
```

```
checkGetArgumentDefaultValue?, checkGetArgumentValidValues?)>
<!ELEMENT checkGetArgumentType (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultLabel (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultDescription (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultValue (#PCDATA)>
<!ELEMENT checkGetArgumentValidValues (checkGetArgumentValidValue+)>
<!ELEMENT checkGetArgumentValidValue (checkGetArgumentValidValueItem,
checkGetArgumentValidValueDisplayName)>
<!ELEMENT checkGetArgumentValidValueItem (#PCDATA)>
<!ELEMENT checkGetArgumentValidValueDisplayName (#PCDATA)>
<!ELEMENT checkSetArguments (checkSetArgument+)>
<!ELEMENT checkSetArgument (checkSetArgumentType,
checkSetArgumentDefaultLabel, checkSetArgumentDefaultDescription,
checkSetArgumentDefaultValue?, checkSetArgumentValidValues?)>
<!ATTLIST checkSetArgument populateFromRule CDATA #IMPLIED>
<!ELEMENT checkSetArgumentType (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultLabel (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultDescription (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultValue (#PCDATA)>
<!ELEMENT checkSetArgumentValidValues (checkSetArgumentValidValue+)>
<!ELEMENT checkSetArgumentValidValue (checkSetArgumentValidValueItem,
checkSetArgumentValidValueDisplayName)>
<!ELEMENT checkSetArgumentValidValueItem (#PCDATA)>
<!ELEMENT checkSetArgumentValidValueDisplayName (#PCDATA)>
<!ELEMENT checkComparisonDefaults (checkComparisonDefaultOperator?,
checkComparisonDefaultValues)>
<!ELEMENT checkComparisonDefaultOperator (#PCDATA)>
<!ATTLIST checkComparisonDefaultOperator not CDATA #IMPLIED>
<!ATTLIST checkComparisonDefaultOperator caseInsensitive CDATA #IMPLIED>
<!ELEMENT checkComparisonDefaultValues (checkComparisonDefaultValue+)>
<!ELEMENT checkComparisonDefaultValue (checkComparisonDefaultValueItem,
checkComparisonDefaultValueDisplayName)>
<!ELEMENT checkComparisonDefaultValueItem (#PCDATA)>
<!ELEMENT checkComparisonDefaultValueDisplayName (#PCDATA)>
<!ELEMENT checkCompareValidValues (checkCompareValidValue+)>
<!ELEMENT checkCompareValidValue (checkCompareValidValueItem,
checkCompareValidValueDisplayName)>
<!ELEMENT checkCompareValidValueItem (#PCDATA)>
<!ELEMENT checkCompareValidValueDisplayName (#PCDATA)>
<!ELEMENT checkSuccessExitCodes (checkSuccessExitCode+)>
<!ELEMENT checkSuccessExitCode (checkSuccessExitCodeValue,
checkSuccessExitCodeDefaultDescription,
checkSuccessExitCodeDefaultDisplayName)>
<!ELEMENT checkSuccessExitCodeValue (#PCDATA)>
<!ELEMENT checkSuccessExitCodeDefaultDescription (#PCDATA)>
<!ELEMENT checkSuccessExitCodeDefaultDisplayName (#PCDATA)>
```

下表描述了 config.xml DTD 的元素。

DTD 元素和特性

元素	特性
checkConfiguration version	设置为 1.0。仅在审核和修正框架需要时才进行更改。
checkName	检查/规则在 SA 客户端中的英文显示名称。
checkGUID	标准 GUID (例如， 9500A4AE-EE9E-4383-87F2-BAD7DDC26C59) 可以使用“guidgen”Windows 实用程序生成、从网站下载，或通过其他方式获取。 GUID 必须唯一，否则无法将可插入检查上载到核心。在使用其唯一 GUID 上载检查之后，不能更改该 GUID。否则，在删除原始项之前，重新上载操作将失败并显示“数据库唯一约束错误”。检查由 GUID 进行唯一标识，但是在上载过程中则由其名称(zip 文件名)标识。
checkDefaultDescription	在 SA 客户端描述框中显示。接受硬回车和 HTML。对于 HTML，需使用 < 和 > 转换 HTML 标记。
checkRemediationDefaultDescription	在 SA 客户端检查/规则的“修正”部分下显示。
checkGetScriptName	get 脚本的文件名，例如 getUsersWithoutPasswordExpiration.sh。
checkGetScriptType	代码类型确定了要运行的解释程序。get 和 set 脚本可以是以下类型：SH、VBS、PY、BAT。
checkSetScriptName	修正脚本的文件名。
checkSetScriptType	代码类型确定了要运行的解释程序。set(修正)脚本可以是 SH、VBS、PY 和 BA 类型。
checkVersion	该元素基于 SA 和框架内部版本号，例如 32b.0-1.0。

DTD 元素和特性(续)

元素	特性
checkAllowRemediationOnFailure	某些脚本可能会在 get 阶段失败，但是您可以通过修正脚本更正该情况。这样，即使在脚本失败时，也可以执行修正。例如，如果未定义注册表项的不存在状态，您可以在 set 代码中创建和设置它。
checkReturnType	允许的值包括 EXITCODE、STRING 或 NUMBER: EXITCODE - 通过 Wscript.Quit()、exit、return 等的标准脚本返回。 NUMBER - 审核和修正框架将从 stdout 抓取数据，并将其解释为数字类型。 STRING - 审核和修正框架将从 stdout 抓取数据，并将其解释为字符串类型。
checkTestIDs	测试 ID 的列表。
checkTestID	用于显示 CIS、MSFT、NSA 或其他策略的标准命名，例如 CIS-RHEL 8.4。这是显示在 SA 客户端中的自由格式字段，因此命名应一致，以便与 TON 内容对应。
checkPlatformTypes	检查的有效平台类型列表。
checkPlatform	WINDOWS | UNIX(或同时作为单独的元素)
checkExclusivePlatforms	专有平台的列表。目前，默认情况下审核和修正会按 Windows 或 Unix 划分内容，但是各种实际标准以及各操作系统中存在的限制和/或差异导致无法始终实现该理想状态。您可以将审核和修正限制为从 spin 检索的平台 ID 所指定的任何平台。 此参数可以引用 SA 支持的平台文档中列出的一个受支持操作系统。

DTD 元素和特性(续)

元素	特性
checkExclusivePlatform	单个平台 ID。
checkExcludePlatforms	排除的平台的列表。如果 PlatformType 声明了 UNIX，您可以提供要从 UNIX 集中排除的平台 ID (所有 Linux + 所有 Unix)。
checkExcludePlatform	单个平台 ID。
checkCategory	这是用于显示检查的 SA 客户端类别。目前，一个检查只能显示在一个类别中。如果类别不存在，将在上载检查时创建类别。应尽可能地对现有检查使用以下标准类别： 事件日志记录: 文件系统 操作系统 操作系统 | 域控制器 (子类别) 操作系统 | 网络 (子类别) 注册表 服务 用户和组
checkGetArgument (checkGetArgumentType, checkGetArgumentDefaultLabel, checkGetArgumentDefaultDescription, checkGetArgumentDefaultValue?, checkGetArgumentValidValues?)>	指定 get 脚本的参数。
checkGetArgumentType	NUMBER | STRING
checkGetArgumentDefaultLabel	输入框或下拉选项旁的 SA 客户端标记。
checkGetArgumentDefaultDescription	包含详细说明信息的悬浮文本。
checkGetArgumentDefaultValue	此 get 参数的默认值。
checkGetArgumentValidValue (checkGetArgumentValidValueItem, checkGetArgumentValidValueDisplayName	checkGetArgumentValidValueItem (#PCDATA)> checkGetArgumentValidValueDisplayName (#PCDATA)>
checkGetArgumentValidValues (checkGetArgumentValidValue+)	(可选)用于限制参数，例如限制为 0/禁用和 1/启用。

DTD 元素和特性(续)

元素	特性
checkSetArguments (checkSetArgument+)	checkSetArgument (checkSetArgumentType, checkSetArgumentDefaultLabel, checkSetArgumentDefaultDescription, checkSetArgumentDefaultValue?, checkSetArgumentValidValues?) setArgument 元素与 GetArguments 相同，但用于修正/set 脚本(如果存在)。 例外是： checkSetArgument populateFromRule - 指定该 set 参数默认值是否应用规则数据进行填充，这与在 config.xml 中提供任何默认值的方式形成对比。通常，该元素始终设置为 true。
checkSetArgumentType	NUMBER | STRING
checkSetArgumentDefaultLabel	输入框或下拉选项旁的 SA 客户端标记。
checkSetArgumentDefaultDescription	包含详细说明信息的悬浮文本。
checkSetArgumentDefaultValue	此 set 参数的默认值。
checkSetArgumentValidValues (checkSetArgumentValidValue+)	
checkSetArgumentValidValue (checkSetArgumentValidValue Item, checkSetArgumentValidValue DisplayName)>	checkSetArgumentValidValueItem (#PCDATA)> checkSetArgumentValidValueDisplay Name (#PCDATA)> checkSetArgumentValidValueItem (#PCDATA)> checkSetArgumentValidValueDisplay Name (#PCDATA)>
checkSetArgumentValidValue Item	(可选)用于限制参数，例如限制为 0/禁用和 1/启用。
checkSetArgumentValidValueDisplayName	
<!ELEMENT checkComparisonDefaults (checkComparisonDefaultOperator?, checkComparisonDefaultValues)>	checkComparisonDefaultOperator not — negation of operator specified, TRUE | FALSE

DTD 元素和特性(续)

元素	特性
	checkComparisonDefaultOperator caseInsensitive — only valid for STRING types.
<!ELEMENT checkComparisonDefaultOperator (#PCDATA)>	比较运算符的默认值列表。用于 TON 构建框架之外的字段或开发。
checkComparisonDefaultValues (checkComparisonDefaultValue+)	checkComparisonDefaultValue (checkComparisonDefaultValueItem, checkComparisonDefaultValueDisplayName).
checkComparisonDefaultValueIte	默认传递给代码的值。
checkComparisonDefaultValueDisplayName	值在 SA 客户端中的显示名称。
checkCompareValidValues (checkCompareValidValue+)> checkCompareValidValue (checkCompareValidValueItem, checkCompareValidValueDisplayName)> checkCompareValidValueItem (#PCDATA)> checkCompareValidValueDisplayName (#PCDATA)>	
checkSuccessExitCodes (checkSuccessExitCode+) checkSuccessExitCode (checkSuccessExitCodeValue, checkSuccessExitCodeDefaultDescription, checkSuccessExitCodeDefaultDisplayName)>	对于 EXITCODE checkReturnType，必须定义有效值才能正确执行脚本操作，这通常同时包括符合条件和不符合条件的预期值。如果返回不同于此处所指定值的任何值，将视为脚本失败，这将在 SA 客户端和报告中以不同的形式显示。
checkSuccessExitCodeValue	脚本完成值，例如 0(通常表示已禁用)。
checkSuccessExitCodeDefaultDescription	DisplayName/Value 的悬浮文本。
checkSuccessExitCodeDefaultDisplayName	向用户显示的值或文本，例如“已禁用”。

搜索筛选器语法

筛选器语法

搜索筛选器是一个用于 findServerRefs 等方法的参数。搜索筛选器中的表达式支持您根据对象特性值获取 SA 对象(例如服务器和文件夹)引用。搜索筛选器的正式语法如下：

```
<filter>                ::=     (<expression-junction>)+
<expression-junction>   ::=     <expression-list-open> <junction>
(<expression>)+ <expression-list-close>
<expression>            ::=     <expression-open> <attribute> <general-delimiter> <operator>
<general-delimiter> <value-list> <expression-close>

<attribute>             ::=     <resource_field>
<vo_member>             ::=     <text>
<resource_field>        ::=     <text>
<value-list>            ::=     (<double-quote> <text> <double-quote>)* |
(<number>)*
<text>                  ::=     [a-z] [A-Z] [0-9]
<number>                ::=     [0-9] [.]

<junction>              ::=      <union-junction> | <intersect-junction>
<union-junction>        ::=     ‘|’
<intersect-junction     ::=     ‘&’
<expression-list-open>  ::=     ‘(‘
<expression-list-close> ::=     ‘)’
<expression-open>       ::=     ‘(‘ | ‘{‘
<expression-close>      ::=     ‘(‘ | ‘}’
<general-delimiter>     ::=     <whitespace>
<whitespace>            ::=     ‘ ‘
<double-quote>          ::=     ‘”’
<escape-character>      ::=     ‘\’
<operator>              ::=     <equal_to> |...| <contains_or_above>

Valid operators for the preceding line:

<equal_to>              ::=     ‘=’ | ‘EQUAL_TO’
<not_equal_to>          ::=     ‘!=’ | ‘<>’ | ‘NOT_EQUAL_TO’
<in>                    ::=     ‘=’ | ‘IN’
<not_in>                ::=     ‘!=’ | ‘<>’ | ‘NOT_IN’
<greater_than>          ::=     ‘>’ | ‘GREATER_THAN’
<less_than>             ::=     ‘<‘ | ‘LESS_THAN’
<greater_than_or_equal> ::=     ‘>=’ | ‘GREATER_THAN_OR_EQUAL’
```

```
<less_than_or_equal>    ::=     ‘<=’ | ‘LESS_THAN_OR_EQUAL’
<begins_with>           ::=     ‘=*’ | ‘BEGINS_WITH’
<ends_with>             ::=     ‘*=’ | ‘ENDS_WITH’
<contains>              ::=    ‘*=*’ | ‘CONTAINS’
<not_contains>          ::=    ‘*<>*’ | ‘NOT_CONTAINS’
<in_or_below>           ::=    ‘IN_OR_BELOW’
<in_or_above>           ::=    ‘IN_OR_ABOVE’
<between>               ::=    ‘BETWEEN’
<not_between>           ::=    ‘NOT_BETWEEN’
<not_begins_with>       ::=    ‘NOT_BEGINS_WITH’
<not_ends_with>         ::=    ‘NOT_ENDS_WITH’
<is_today>              ::=    ‘IS_TODAY’
<is_not_today>          ::=    ‘IS_NOT_TODAY’
<within_last_days>      ::=    ‘WITHIN_LAST_DAYS’
<within_last_months>    ::=    ‘WITHIN_LAST_MONTHS’
<within_next_days>      ::=    ‘WITHIN_NEXT_DAYS’
<within_next_months>    ::=    ‘WITHIN_NEXT_MONTHS’
<not_within_last_days>  ::=    ‘NOT_WITHIN_LAST_DAYS’
<not_within_last_months>::=    ‘NOT_WITHIN_LAST_MONTHS’
<not_within_next_days>  ::=    ‘NOT_WITHIN_NEXT_DAYS’
<not_within_next_months>::=    ‘NOT_WITHIN_NEXT_MONTHS’
<contains_or_below>     ::=    ‘CONTAINS_OR_BELOW’
<contains_or_above>     ::=    ‘CONTAINS_OR_ABOVE’
```

重建 Apache HTTP 服务器和 PHP

此主题描述在 SA 中如何重建 Apache HTTP 服务器和 PHP 并替换它们。SA 中包含一个 Apache HTTP 服务器和 PHP。因此，只有在需要使用其他 Apache HTTP 服务器版本，或者需要将其他库或模块编译到 PHP 中时，才需阅读本信息。

SA 使用 Apache HTTP 服务器和 PHP 进行 Web 自动化平台扩展 (APX)。有关详细信息，请参见 自动化平台扩展 (APX) (第 92 页)。

扩展 APX HTTP 环境

本节描述如何通过重建 Apache HTTP 服务器和 PHP 来扩展 APX HTTP 环境。

必须在所有核心升级之后执行这些任务。

如果您具有多主控网状网络，则必须对所有核心中的每个切分执行这些任务。有关切分组件捆绑包的详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

重建 PHP

1. 执行以下任务重建 PHP。

   从 http://www.php.net/ 下载 PHP 源代码。
2. 将源代码放置在安装了 apxproxy 的服务器上的目录下，通常为 /opt/opsware/apxproxy 目录下。
3. 输入以下命令，替换版本号(如果您下载的是不同版本的 PHP)。

```
mkdir /build ; cp php-4.4.8.tar.gz /build; cd /build
gzip -dc php-4.4.8.tar.gz | tar xvf -
cd php-4.4.8
./configure --prefix=/opt/opsware/apxphp
--with-pear=/opt/opsware/apxphp/lib/pear
--with-config-file-path=/opt/opsware/apxphp/lib
--with-apxs2=/opt/opsware/apxhttpd/bin/apxs <any other options you>
make clean
make
```

4. 备份 libphp4.so 的旧副本：

   ```
   cp /opt/opsware/apxhttpd/modules/libphp4.so
   /opt/opsware/apxhttpd/modules/libphp4.so.backup
   ```

5. 将新的 libphp4.so 文件复制到 apxhhtps 目录：

   ```
   cp libs/libphp4.so /opt/opsware/apxhttpd/modules/libphp4.so
   ```

6. 确保 tool.list 中存在完整的引用库:

   ```
   ldd ./libs/libphp4.so
   ```

   对于输出中的每个条目，确保该文件存在于 /etc/opt/opsware/ogfs/tool.list 中。

   如果条目不存在，请添加它。

7. 备份 apxphp 文件夹：

   ```
   mv /opt/opsware/apxphp /opt/opsware/apxphp.orig
   ```

8. 安装 PHP:

   ```
   make install
   ```

9. 重新加载和链接 OGFS，确保您添加到 /etc/opt/opsware/ogfs/tools.list 中的任何内容都显示在 OGFS 中：

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/
   tools/reload
   ```

10. 重新启动 apxproxy：

    ```
    /etc/opt/opsware/startup/apxproxy restart
    ```

重建 Apache

执行以下任务重建 Apache HTTP 服务器。

1. 从 http://httpd.apache.org/ 下载 Apache HTTP 服务器的源代码。
2. 将源代码放置在切分组件捆绑包所在的服务器上的目录中。有关切分组件捆绑包的详细信息，请参见 HPE SSO 门户上的“Server Automation 管理指南”。

3. 输入以下命令，替换版本号(如果您下载的是不同版本的 httpd)。

   ```
   mkdir /build; cp httpd-2.2.8.tar.gz /build; cd /build
   gzip -dc httpd-2.2.8.tar.gz | tar xf -
   cd httpd-2.2.8
   ./configure --prefix=/opt/opsware/apxhttpd <any other options you want>.
   ```

   SA 当前使用：

   ```
   --enable-mods-shared="actions alias auth_basic auth_digest authn_file authz_
   user cgi deflate dir dumpio env expires headers ident logio log_config mime
   negotiation rewrite userdir vhost_alias imagemap status"
   --disable-dav
   --with-port=8021
   --with-expat=builtin
   --without-pgsql
   ```

   (仅适用于 SunOS)输入以下命令：

   ```
   perl -pi -e 's/#define HAVE_GETADDRINFO 1/#undef HAVE_GETADDRINFO/g'
   ./srclib/apr/include/arch/unix/apr_private.h
   make
   ```

4. 生成 apxhttp 目录的备份：

   ```
   mv /opt/opsware/apxhttpd /opt/opsware/apxhttpd.orig
   ```

5. 安装 Apache：

   ```
   make install
   ```

6. 将新文件重新加载和链接到 OGFS 中：

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/tools/reload
   ```

7. 模块目录中的 HTTPD 和 .so 文件可能会引用外部库。这些库必须对 OGFS 可见或在 OGFS 中闪烁。

   登录 OGFS，并在 /opt/opsware/apxhttpd/bin/httpd 以及 /opt/opsware/apxhttpd/modules 中的任何 .so 上运行 LDD，确保其中列出的所有文件都存在于 OGFS 中。如果不存在，则将文件添加到 /etc/opt/opsware/ogfs/tool.list (在 OGFS 外部)，然后重新运行步骤 6，直到所有文件都对 /opt/opsware/apxhttpd/bin/httpd 可用。

8. 此时，必须重建 PHP。请参见 重建 PHP (第 210 页)。

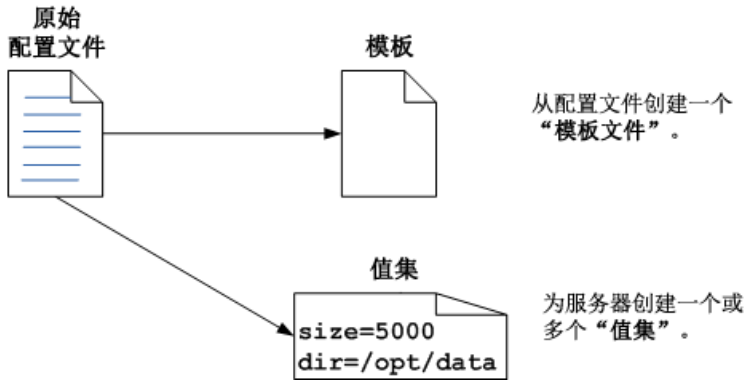
应用程序配置

本节将帮助您了解高级别的应用程序配置。它将说明对应用程序配置进行设置和管理所需执行的步骤。

- 有关如何执行这些任务的详细说明，请参见应用程序配置任务 (第 245 页)。
- 有关应用程序配置的背景信息，请参见应用程序配置概念 (第 217 页)。

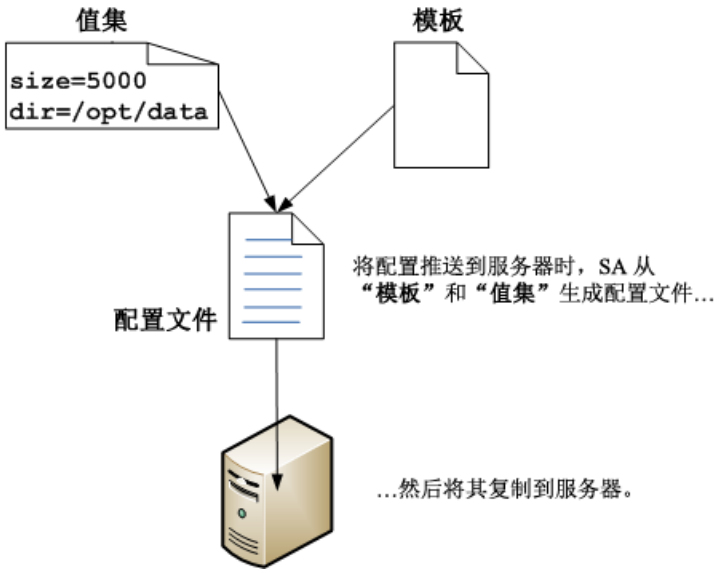
要创建和使用应用程序配置，必须首先创建“模板”文件和“值集”，如下图所示。模板是配置文件的模型。值集是数据值，这些值将与模板合并，用以创建要推送到服务器的配置文件实例。有关详细说明，请参见下图:

应用程序配置创建



在创建模板和值集之后，将应用程序配置**附加**到一台或多台服务器并将配置**推送**到服务器，如下图所示。有关详细说明，请参见下图:

应用程序配置推送图



要创建和使用应用程序配置，请执行以下操作：

1. **确定要管理的配置文件**。选择要管理的应用程序或系统配置文件。例如，对于 Apache Web 服务器，您可以管理 http.conf、password.conf、obj.conf、mimetypes 和 magnus.conf 文件。
2. **创建每个配置文件的模板文件**。对于要管理的每个配置文件，请基于此配置文件创建模板文件。模板是原始配置文件的模型，其中包含随服务器变化的值的占位符。此模板可指定配置文件中哪些值为固定值，哪些值为变量(随服务器的不同而不同)。另请参考以下主题:
   - 创建应用程序配置 (第 246 页)
   - 使用可视化编辑器创建配置模板 (第 249 页)
   - 对于 XML 配置文件，请创建 XML 或 XML-DTD 模板文件。有关详细信息，请参见 管理 XML 配置文件 (第 282 页)。
   - 对于其他配置文件，请使用配置建模语言 (CML) 创建模板文件。有关详细信息，请参见CML 引用 (第 352 页)和CML 教程 1 - 为简单 Web 应用程序服务器创建应用程序配置 (第 306 页)。
   - 配置模板和脚本模板 (第 218 页)。
3. **创建每个脚本的模板文件**。在某些情况中，可能需要在更新配置文件之前或之后运行脚本。您必须使用 CML 从脚本创建模板文件。 如果无需针对配置文件运行任何脚

本，则跳过此步骤。另请参见：
- 从脚本创建模板 (第 266 页)。
- 配置模板和脚本模板 (第 218 页)。
- 使用应用程序配置运行脚本 (第 234 页)。

4. **将模板文件导入 SA 库**。在从配置文件和脚本创建所有模板之后，可将模板导入 SA 库中。或者也可以在 SA 客户端中直接创建这些模板。有关详细信息，请参见 导入和验证模板文件 (第 263 页)。
5. **创建应用程序配置对象。**应用程序配置对象仅仅是容纳一个或多个模板的容器。另请参见：
   - 创建应用程序配置 (第 246 页)。
   - 应用程序配置管理 (第 218 页)。
6. **将模板添加到应用程序配置对象**。创建并将模板导入 SA 库中之后，可将这些模板添加到应用程序配置对象。此外，还可指定文件的安装顺序以及脚本(如果有)的执行时间。有关详细信息，请参见 将模板添加到应用程序配置或从应用程序配置删除模板 (第 264 页)。
7. **将应用程序配置附加到服务器**。创建和配置应用程序配置之后，可将其附加到要使用它的服务器。这将创建应用程序配置的一个实例。服务器上可以具有此应用程序配置的多个实例。每个实例可以具有不同的用途。例如，一个实例可用于配置应用程序的临时版本，另一个实例可用于配置应用程序的生产版本。或者，如果服务器上运行应用程序的三个实例，则可以通过应用程序配置的三个实例来配置应用程序的每个实例。有关详细信息，请参见以下主题:
   - 将应用程序配置附加到服务器或设备组 (第 268 页)。
   - 软件策略中的应用程序配置 (第 244 页)。
8. **创建服务器的值集**。设置将放置于模板的值，以生成将推送到服务器的实际配置文件。可以在多个级别设置默认值，如果没有在较低级别覆盖这些值，则将继承这些默认值。有关详细信息，请参见以下主题:
   - 在值集编辑器中设置值 (第 224 页)
   - 值集 (第 221 页)
9. **将实际配置文件与配置模板进行比较**。 (可选)可轻松将配置模板与服务器上的实际配置文件进行比较，查看是否存在任何差异。这会显示服务器上配置文件的内容以及向服务器推送应用程序配置时将复制到服务器的值。有关详细信息，请参见以下主题:
   - 将配置模板与目标配置文件进行比较 - 预览 (第 278 页)
   - 比较配置模板 (第 279 页)

10. **推送配置更改**。要更新服务器上的配置文件，需将应用程序配置“推送”到这些服务器。在将更改推送到服务器之后，才会更改服务器上的实际配置文件。可将应用程序配置更改推送到单台服务器或服务器组。有关详细信息，请参见以下主题：
    - 推送应用程序配置 (第 272 页)
    - 将应用程序配置推送到服务器 (第 236 页)
11. **审核配置、监控符合性和修正**。要审核配置、监控符合性以及修正已更改的配置，请参见应用程序配置符合性 (第 237 页)和 HPE SSO 门户上的“Server Automation 管理指南”。

应用程序配置概念

使用 Server Automation，可以从一个中心位置管理配置文件 (包括 XML 配置文件)，并在数据中心的多台服务器中轻松传播更改和更新。您可以管理单个配置文件(如 UNIX 系统中的 /etc/hosts 文件)，也可以管理与某个应用程序关联的多个复杂配置文件，例如与某个大型业务应用程序(如 WebLogic 或 Websphere)关联的配置文件。

配置文件是服务器上需要您控制其内容的任何文件。这包括应用程序配置文件以及系统配置文件，例如：

- 可控制应用程序服务器、Web 服务器、数据库或其他应用程序行为的文件。例如，您可以管理跨多个不同服务器运行的 Apache Web 服务器的 httpd.conf 文件。可附加此文件的应用程序配置并在每台服务器上为该文件的每个实例输入特定值。
- 服务器上的文件，例如 UNIX 服务器上的 /etc/hosts、/etc/fstab、/etc/passwd 或 /etc/groups 文件。
- 服务器上用于配置服务器的任何其他文件。

此外，您可以使用应用程序配置执行以下操作：

- 在服务器上放置配置文件之前或之后**运行脚本**例如，可使用脚本，以便在将配置推送到服务器之后重新启动应用程序。有关详细信息，请参见使用应用程序配置运行脚本 (第 234 页)。
- 在**软件策略**中使用应用程序配置，以作为应用程序部署和持续管理的一部分。有关详细信息，请参见软件策略中的应用程序配置 (第 244 页)。
- **审核**以确定服务器是否符合所需的应用程序配置文件内容。有关详细信息，请参见审核应用程序配置 (第 243 页)。

- 根据所定义的应用程序配置，对服务器进行**符合性检查**。有关详细信息，请参见应用程序配置符合性 (第 237 页)。
- **恢复**先前的配置文件。有关详细信息，请参见将配置文件恢复到之前的状态 (第 274 页)。

应用程序配置管理

要管理服务器上的配置文件，需要在 SA 库中创建应用程序配置对象，并至少创建一个配置模板。应用程序配置对象只是一个容纳模板和可选脚本的容器。

在将应用程序配置推送到一台或多台服务器时，SA 会执行以下操作：

- 从配置模板和指定值集生成配置文件。
- 将生成的配置文件复制到服务器。

下图显示了一个示例应用程序配置，其中包含一个名为 "exports.tpl" 的模板以及名为 "post-exports.sh" 的 UNIX Shell 脚本文件。“.tpl”文件扩展名指示这些文件为模板。

包含配置模板和 Shell 脚本的应用程序配置

配置模板和脚本模板

配置模板是配置文件的模型，它用变量代替随服务器变化的数据值。此外，配置模板还包含有关如何使用模板重新创建配置文件以及如何在推送操作过程中使用值集重新创建配置文件的说明。配置模板可定义配置文件的固定部分和变量部分。模板还包含解析配置文件和重新创建配置文件以推送到托管服务器的说明。

可以使用 SA 配置建模语言 (CML) 或 XML 创建配置模板文件。针对所有基于 XML 的配置文件使用 XML，针对所有其他配置文件使用 CML。有关 CML 的完整信息，请参见CML 引用 (第 352 页)。有关 XML 配置文件的信息，请参见管理 XML 配置文件 (第 282 页)。

您可以定义将用于生成最终配置文件的值。在 SA 数据库中，这些值被存储为值集。配置模板中的 CML 或 XML 将值集中的值与模板文件合并，以生成目标配置文件。有关详细信息，请参见值集 (第 221 页)。

您还可以使用配置模板从配置文件导入数据并构建值集。有关详细信息，请参见 导入和验证模板文件 (第 263 页)。

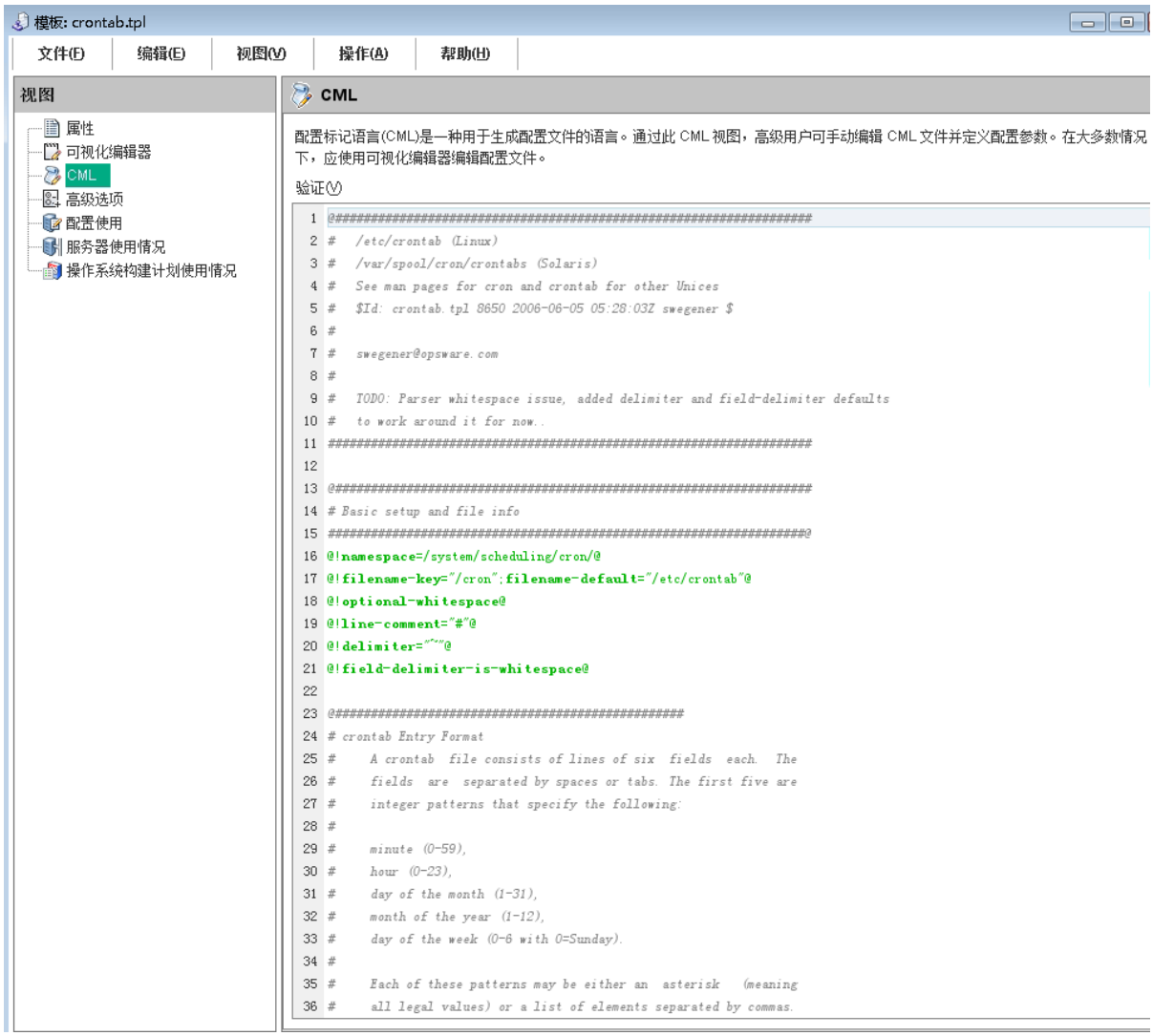
要管理配置文件，请为要管理的每个文件创建配置模板并将其添加到应用程序配置。有关详细信息，请参见 创建应用程序配置 (第 246 页)。

CML 配置模板

配置建模语言 (CML) 提供了创建配置文件模板的一种方式，可将该模板与值集合并用以生成实际的配置文件。下图显示了采用配置建模语言 (CML) 创建的示例配置模板文件。

有关详细信息，请参见CML 教程 1 - 为简单 Web 应用程序服务器创建应用程序配置 (第 306 页)、CML 教程 2 - 创建 Web 服务器配置文件的模板 (第 316 页)和 CML 引用 (第 352 页)。

以下是使用 CML 编写的配置模板示例。



XML 和 XML-DTD 配置模板

您还可以在托管服务器上管理 XML 配置文件。通过使用 XML 配置模板，可以对 XML 配置文件值进行建模并根据实际 XML 检查这些值，然后将更改推送到目标文件。您可以对使用或不使用 DTD 的 XML 配置文件进行建模。

XML 配置模板的功能与 CML 模板类似，但是前者使用注释中定义的某些应用程序配置选项。此外，通过 XML 配置模板，还可以使用标记来自定义文件在 SA 客户端中的显示方式。

有关详细信息，请参见 管理 XML 配置文件 (第 282 页)。

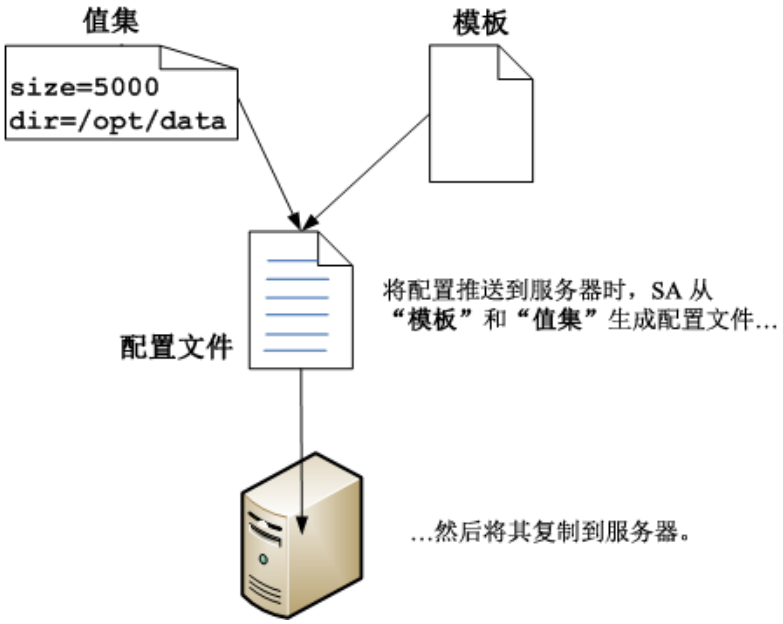
脚本模板

您可以将脚本添加到应用程序配置，这些脚本将在向目标服务器复制配置值之前或之后执行。要使应用程序配置对象中包含脚本，必须将脚本复制到 CML 模板，然后将脚本模板导入应用程序配置对象中。

有关详细信息，请参见使用应用程序配置运行脚本 (第 234 页)和从脚本创建模板 (第 266 页)。

值集

值集是一组数据值，可与模板文件合并用以生成目标配置文件。生成的配置文件可推送到服务器。



简单地说，可以为特定服务器定义“服务器实例”级别的值集。将“服务器实例”值集中的值与配置模板合并，以生成将推送到服务器的实际配置文件。

值集级别和值集继承

为各台服务器设置服务器实例级别的值适用于较小数量的服务器，但对于大量服务器，可以使用值集继承来设置较高级别的默认值，这些值可应用于托管服务器的较大子集并可由较低级别继承。每个级别将继承其上级别的值，除非此级别明确阻止继承或设置将覆盖继承值的值。

下表列出了所有值集继承级别、在较低级别继承或覆盖值的方式以及在每个级别设置值的方式。以下描述了在每个级别设置值的详细信息。

值集级别和值集继承

级别	级别的描述	值的设置方式
应用程序	此级别定义应用于应用程序配置本身的值。除非被以下任意级别覆盖，否则它将应用于已附加应用程序配置的所有服务器。	打开“应用程序配置”对象。选择“内容”>“应用程序值”。选择一个模板。选择“文件值”视图。请参见 在应用程序级别设置值 (第 228 页)。
设施	此级别定义设施的值。除非被以下任意级别覆盖，否则它将应用于指定设施中已附加应用程序配置的所有服务器。	打开“应用程序配置”对象。选择“内容”>“设施值”><设施>。选择一个模板。选择“文件值”视图。请参见 在设施级别设置值 (第 229 页)。
客户	此级别定义客户的值。除非被以下任意级别覆盖，否则它将应用于指定客户中已附加应用程序配置的所有服务器。	打开“应用程序配置”对象。选择“内容”>“客户值”><客户>。选择一个模板。选择“文件值”视图。请参见 在客户级别设置值 (第 229 页)。
组	此级别定义设备组的值。除非被以下任意级别覆盖，否则它将应用于指定设备组中已附加应用程序配置的所有服务器。	打开设备组。选择“配置的应用程序”><应用程序配置名称>。选择一个模板。请参见 在组级别设置值 (第 231 页)。
组实例	此级别定义应用程序配置的特定实例值。除非被以下任意级别覆盖，否则它将应用于指定设备组中已附加实例的所有服务器。	打开设备组。选择“配置的应用程序”><应用程序配置名称>><实例名称>。选择一个模板。请参见 在组实例级别设置值 (第 232 页)。
服务器	此级别定义服务器的值。除非被以下任意级别覆盖，否则它将应用于指定服务器上应用程序配置的所有实例。	打开服务器。选择“管理策略”选项卡。选择“配置的应用程序”><应用程序配置名称>。选择一个模板。请参见 在服务器级别设置值 (第 233 页)。
服务器实例	此级别定义服务器上应用程序配置的特定实例值。它仅应用	打开服务器。选择“管理策略”选项卡。选择“配置的应用程序”><应用程序配

值集级别和值集继承(续)

级别	级别的描述	值的设置方式
	于指定服务器上的指定应用程序配置实例并覆盖所有以上级别。	。选择一个模板。请参见 在服务器实例级别设置值 (第 234 页)。

下表说明了应用程序配置值的继承方式。每行中的值表示该级别的值集。最下面一行显示将推送到服务器的实际值。

表 2：应用程序配置值的继承

级别	每个级别的值集						
应用程序	2	1	Z	Y	X	W	V
设施		U	T	S	R	Q	P
客户			O	N	M	L	K
组				J	I	H	G
组实例					F	E	D
服务器						C	B
服务器实例							A
继承结果	2	U	O	J	F	C	A

阻止继承

通过在值集编辑器中为任意变量选择“阻止继承”，可以在任何级别阻止继承。将不会继承阻止的级别之前的所有值集。但仍然继承阻止的级别之后的级别中的值。较低级别必须设置此值，否则此变量将不具有任何值。

1. 在任意级别打开值集编辑器。
2. 在任意变量的值列中，选择一个值。将在编辑框的最右侧显示一个下拉列表和“...”按钮。
3. 选择下拉列表或“...”按钮。将显示变量值的多个选项。
4. 选择“阻止继承”。
5. 选择“文件”>“保存”或“保存更改”按钮。

有关详细信息，请参见 在值集编辑器中设置值 (第 224 页)。

值集编辑器

值集编辑器显示模板中定义的变量并支持您设置这些变量的值。在任意继承级别选择应用程序配置的值集时会显示此值集编辑器。此部分将描述如何使用值集编辑器。

在值集编辑器中设置值

只要设置的值符合变量的数据类型，您就可以在值集编辑器中为变量设置任何值。在所需变量旁边的“值”列中键入值。

此外，也可以按以下任意方式设置值：

- 选择编辑框最右侧的下拉列表并选择以下值之一：
  - 空字符串 - 指定将长度为零的字符串放置于值集中。
  - 阻止继承 - 指定不会继承较高级别的值。较低级别必须设置此值，否则此变量将不具有任何值。
  - 代理版本、验证域、机箱 ID、客户 ID、客户名称等 - 指定将托管服务器的选定信息放置于值集中。
- 选择编辑框右侧的“...”按钮可显示以下选项：
  - 没有值 - 将值集中的值保留为空。
  - 阻止继承 - 指定不会继承较高级别的值。较低级别必须设置此值，否则此变量将不具有任何值。
  - 任何值 - 输入任何值。
  - 对象特性 - 从以上列表中选择其中一个值。
  - 自定义特性 - 输入自定义特性。
  - 部署自动化值 - 从部署自动化管理的应用程序输入值。.
- 右键单击编辑框可显示编辑菜单。

在值集编辑器中设置字段

无论在哪个级别设置值，值集编辑器都会为正在编辑的值集显示以下所述的字段。

服务器级别的值集编辑器 (第 232 页)图显示了附加有名为 "WAS-app-config" 的应用程序配置的服务器。其中显示了服务实例级别的值集，您可以对其进行编辑。有关以下字段的示例请查看此图像。

- **模板**：列出应用程序配置对象中包含的模板文件并允许您选择要查看和修改的模板。选择要查看的模板。
- **文件名**：指定托管服务器(配置模板的目标)上的配置文件名称。如果未设置此文件名，则将继承此文件名。如果应用程序配置层次结构中没有设置任何文件名，则将使用配置模板中列出的文件名。
  如果服务器上的应用程序具有多个实例，则使用此字段来指示每个目标配置文件的完整路径名称。
- **编码**：指定目标配置文件的字符编码。默认为托管服务器上使用的编码。(请注意，SA 客户端中不支持使用 UTF-16 编码。)
- **保留格式**：指定是否保留目标配置文件的空格、注释和顺序。SA 将试图尽可能保留目标配置文件的格式，但是可能无法保留所有注释和格式。如果模板使用 `@!partial-template@` CML 标记，则此选项必选。
  如果没有为模板打开此选项，则将从文件删除所有注释和格式，并使用模板中的默认顺序和空格。

  > **备注：**请注意，对于基于 XML 的模板，保留格式将不能保留 XML 标记内的空格或特性顺序。保留格式将保留所有除标记内的空格和特性顺序之外的所有空格和顺序。推送后，标记内的额外空格将消失，其中特性的顺序也可能会更改。消除空格和更改特性顺序不会影响 XML 的含义。

- **保留值**：指定当值集中没有对应的值时是否保留托管服务器上目标配置文件中包含的值。默认情况下，关闭此选项。

  通过“保留值”，您可以请求保留服务器的配置文件中尚未存储在 SA 数据库中的任何值。如果不打算将所有当前值从配置文件导入 SA 中或者用户或程序有时在 SA 外部修改配置文件，则此选项将十分有用。它允许您在 SA 外部管理某些配置文件更改。
- **显示所继承的值**：显示生成的值集，其中包括从较高级别继承的值。关闭此选项时，将仅显示当前级别的值集。此视图为只读，且仅在服务器实例级别查看值集时才提供。

值集编辑器中的列

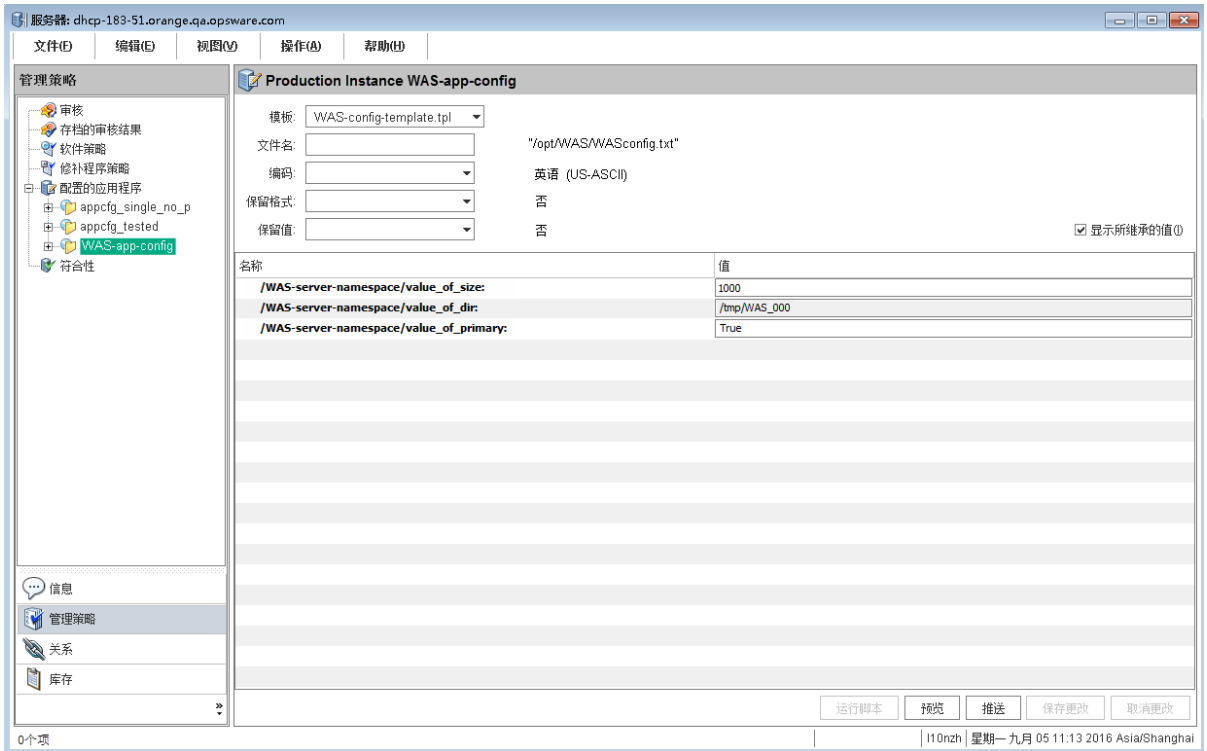
在编辑值集时，SA 客户端会显示有关配置模板值的以下信息。

服务器级别的值集编辑器 (第 232 页)图显示了附加有名为 "WAS-app-config" 的应用程序配置实例的服务器。其中显示了服务实例级别的值集，您可以对其进行编辑。以下描述了列“名称”、“值”以及“继承自”。

- **名称**：列出模板文件中的变量名称。此名称可以是简单类型、简单类型的列表，也可以是多维列表。多维列表将显示在其父列表下方。所需元素均显示为粗体。可双击以显示或隐藏多维列表。要将其他条目添加到列表类型值，请右键单击父列表，然后选择“添加项目”。配置模板中设置了必填字段。必填字段不能为空，否则将无法预览或推送应用程序配置。
- **值**：列出所显示的值集中的值。您可以输入文本值或从服务器设置中选择特性，如客户名称、客户 ID、机箱 ID、设备 ID 等。如果您将某个设置留空，则会从父级或上级(如果父级或上级配置了值)继承该设置。要将值设置为值的自定义特性，请单击“...”按钮以使用“设置值”对话框。
- **继承自**：指示从哪里继承值。仅在服务器实例级别查看并已选择“显示所继承的值”选项时，才会显示此列。

  如果已设置“保留值”选项，则服务器上的配置文件将成为继承层次结构的最外层级别。也就是说，如果值集中不存在任何值，就会保留此文件中的值。

服务器实例级别的值集编辑器



从现有配置文件导入值集

虽然可以手动设置值集中的值，但是也可以将值从托管服务器上的现有配置文件导入值集中。

要将值从现有配置文件导入值集中，请执行以下操作：

1. 在要导入值的级别显示值集编辑器。导入选项仅在(服务器)实例级别可用。请参见以下部分了解如何在每个级别显示值集编辑器。
2. 在值集编辑器中，请确保“文件名”字段显示包含要导入值的配置文件的绝对文件名。
3. 右键单击“值”列表并选择“导入值”。配置模板的所有值都将被替换为实际配置文件的值。

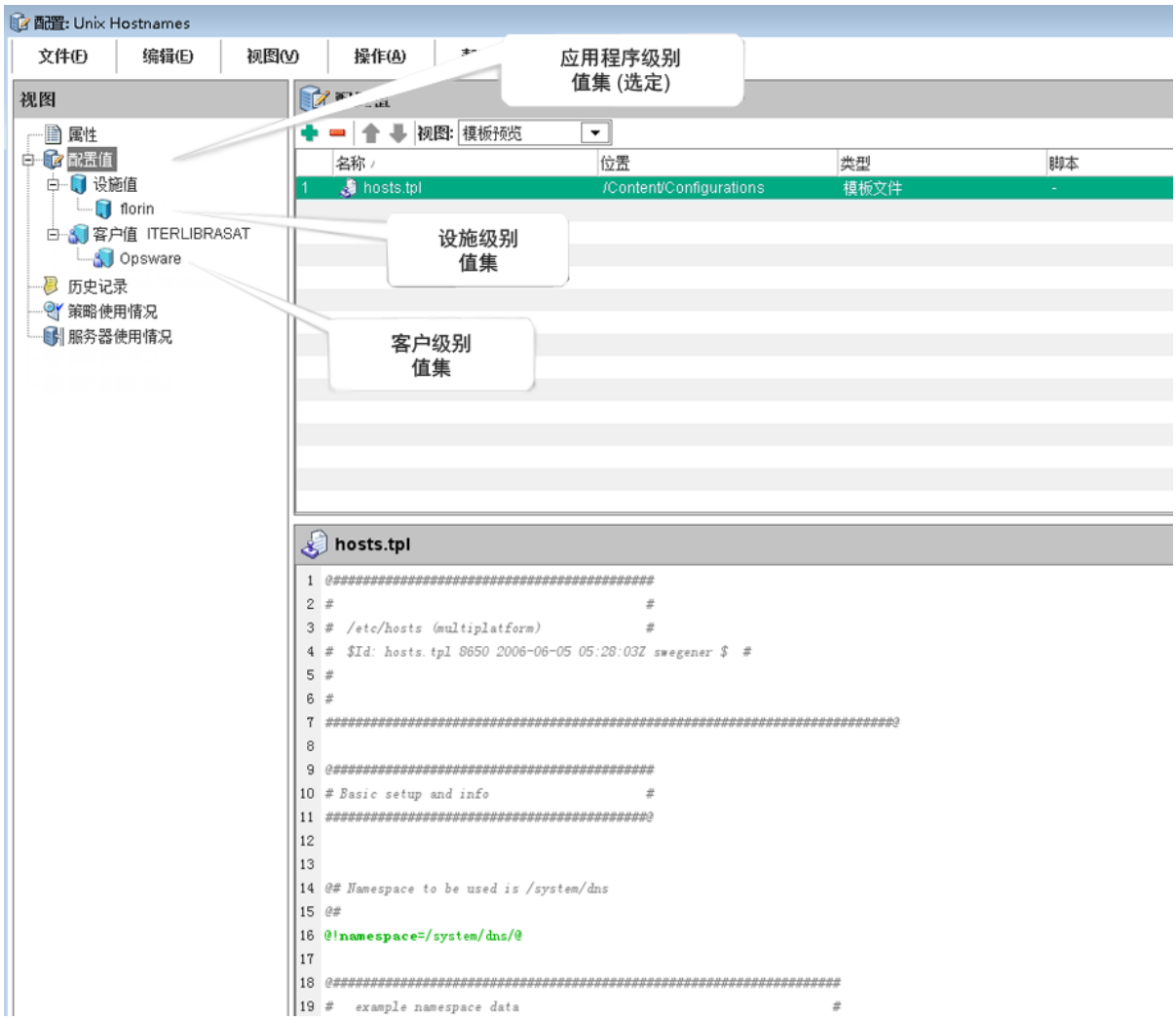
   选择“导入值”菜单项将读取托管服务器上的现有配置文件、解析值并将其保存在选定级别。导入值后，可修改任意值并将更改推送回服务器(如果需要)。
4. 选择“保存更改”。

应用程序、设施和客户级别的值集编辑器

在应用程序、设施和客户级别的值集编辑器 - 已选择应用程序级别 (第 227 页)图显示了应用程序、设施和客户级别的值集编辑器。仅在打开应用程序配置对象时，此视图才可用。

- 选择“配置值”将显示应用程序级别的值集编辑器。
- 在“设施值”下选择一个设施将显示设施级别的值集编辑器。
- 在“客户值”下选择一个客户将显示客户级别的值集编辑器。

在应用程序、设施和客户级别的值集编辑器 - 已选择应用程序级别



在应用程序级别设置值

应用程序级别定义应用于应用程序配置本身的值。除非被以下任意级别覆盖，否则它将应用于已附加应用程序配置的所有服务器。

要在应用程序级别设置值，请执行以下操作：

请参见应用程序、设施和客户级别的值集编辑器 (第 227 页)中的示例，其显示了应用程序、设施和客户级别的值集编辑器。

1. 在 SA 客户端中打开应用程序配置对象。
2. 选择“配置值”节点。
3. 在应用程序配置中选择一个模板。

4. 从“视图：”下拉列表中选择“文件值”。此时将在右下侧显示值集编辑器，在此您可以在应用程序级别设置默认值。
5. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。
6. 可选择从“视图：”下拉列表选择“文件预览”以查看生成的文件。
7. 选择“文件”>“保存”以保存更改。

在设施级别设置值

设施级别定义设施的值。除非被以下任意级别覆盖，否则它将应用于指定设施中已附加应用程序配置的所有服务器。

要在设施级别设置值，请执行以下操作：

请参见应用程序、设施和客户级别的值集编辑器 (第 227 页)。<MadCap:xref href="#app_config_756947499_1518244"> </MadCap:xref>中的示例，其显示了应用程序、设施和客户级别的值集编辑器。

1. 在 SA 客户端中打开应用程序配置对象。
2. 打开“配置值”视图可显示“设施值”节点。
3. 打开“设施值”节点并选择一个设施。
4. 在应用程序配置中选择一个模板。
5. 从“视图”下拉列表中选择“文件值”。此时将在右下侧显示值集编辑器。
6. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。
7. 可选择从“视图”下拉列表选择“文件预览”以查看生成的值。
8. 选择“文件”>“保存”以保存更改。

在客户级别设置值

客户级别定义客户的值。除非被以下任意级别覆盖，否则它将应用于指定客户中已附加应用程序配置的所有服务器。

要在客户级别设置值，请执行以下操作：

请参见应用程序、设施和客户级别的值集编辑器 (第 227 页)中的示例，其显示了应用程序、设施和客户级别的值集编辑器。

1. 在 SA 客户端中打开应用程序配置对象。
2. 打开“配置值”视图可显示“客户值”节点。
3. 打开“客户值”节点并选择一个客户。
4. 在应用程序配置中选择一个模板。
5. 从“视图”下拉列表中选择“文件值”。此时将在右下侧显示值集编辑器。
6. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。另请参见在值集编辑器中设置值 (第 224 页)。
7. 可选择从“视图”下拉列表选择“文件预览”以查看生成的值。
8. 选择“文件”>“保存”以保存更改。

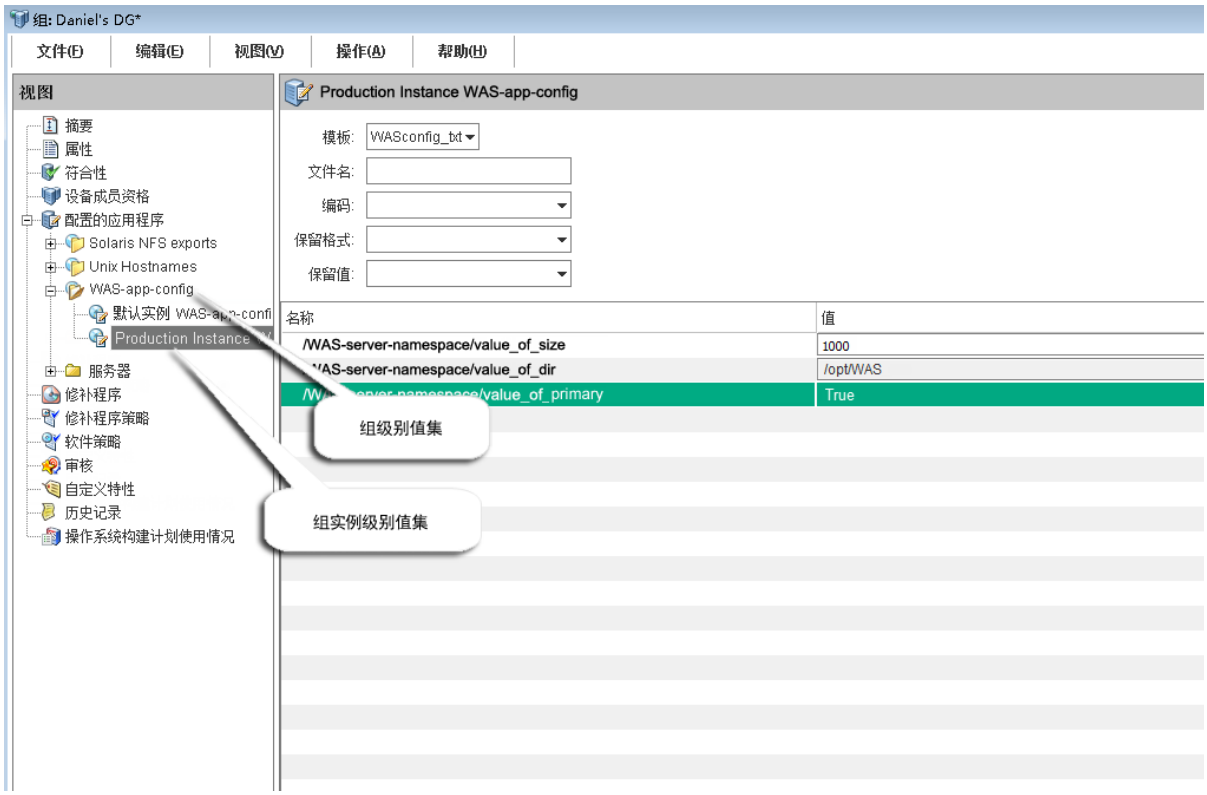
组级别的值集编辑器

组级别定义设备组的值。除非被以下任意级别覆盖，否则它将应用于指定设备组中的所有服务器(假设设备组已附加应用程序配置)。

组级别和组实例级别的值集编辑器 (第 230 页)显示了组和组实例级别的值集编辑器。仅在设备组附加了应用程序配置之后，组和组实例级别才可用。仅在打开已附加应用程序配置的设备组时，此视图才可用。有关详细信息，请参见将应用程序配置附加到服务器或设备组 (第 268 页)。

- 选择名为“WAS-app-config”的应用程序配置时将显示组级别的值集编辑器。
- 选择其中一个应用程序配置实例时将显示组实例级别的值集编辑器。此示例显示了名为“Production Instance WAS-appconfig”和“Staging Instance WAS-appconfig”的两个应用程序配置实例。此值集编辑器显示“Production Instance WAS-appconfig”的值集。

组级别和组实例级别的值集编辑器



在组级别设置值

组级别定义设备组的值。除非被以下任意级别覆盖，否则它将应用于指定设备组中的所有服务器。

在组级别设置值之前，必须将应用程序配置附加到设备组。请参见组级别和组实例级别的值集编辑器 (第 230 页)图中的示例，其显示了组和组实例级别的值集编辑器。

要在组级别设置值，请执行以下操作：

1. 在 SA 客户端中打开设备组。必须将应用程序配置附加到设备组。
2. 在左侧的“视图”窗格中打开“配置的应用程序”节点。
3. 在“配置的应用程序”节点下选择所需的应用程序配置节点。此时将在右侧显示值集编辑器。
4. 在“模板:”下拉列表中选择所需的模板文件。
5. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。
6. 选择“保存更改”按钮。

在组实例级别设置值

组实例级别为附加到设备组的应用程序配置的实例定义值。除非被以下任意级别覆盖，否则它将应用于指定设备组中的所有服务器。

在组实例级别设置值之前，必须将应用程序配置附加到设备组。请参见下图中的示例，其显示了组和组实例级别的值集编辑器。

要在组实例级别设置值，请执行以下操作：

1. 在 SA 客户端中打开设备组。必须将应用程序配置附加到设备组。请参见 将应用程序配置附加到服务器或设备组 (第 268 页)。
2. 在左侧的“视图”窗格中打开“配置的应用程序”节点。
3. 在“配置的应用程序”节点下打开所需的应用程序配置节点。
4. 选择所需的应用程序配置实例。例如，组级别和组实例级别的值集编辑器 (第 230 页)显示了选定的名为 "Production Instance WAS-appconfig" 的应用程序配置实例。这是名为 WAS-app-config 的应用程序配置对象的一个实例。此应用程序配置被附加到设备组并定义了此应用程序配置的两个实例。
5. 在“模板:”下拉列表中选择所需的模板文件。
6. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。另请参见在值集编辑器中设置值 (第 224 页)。
7. 选择“保存更改”按钮。

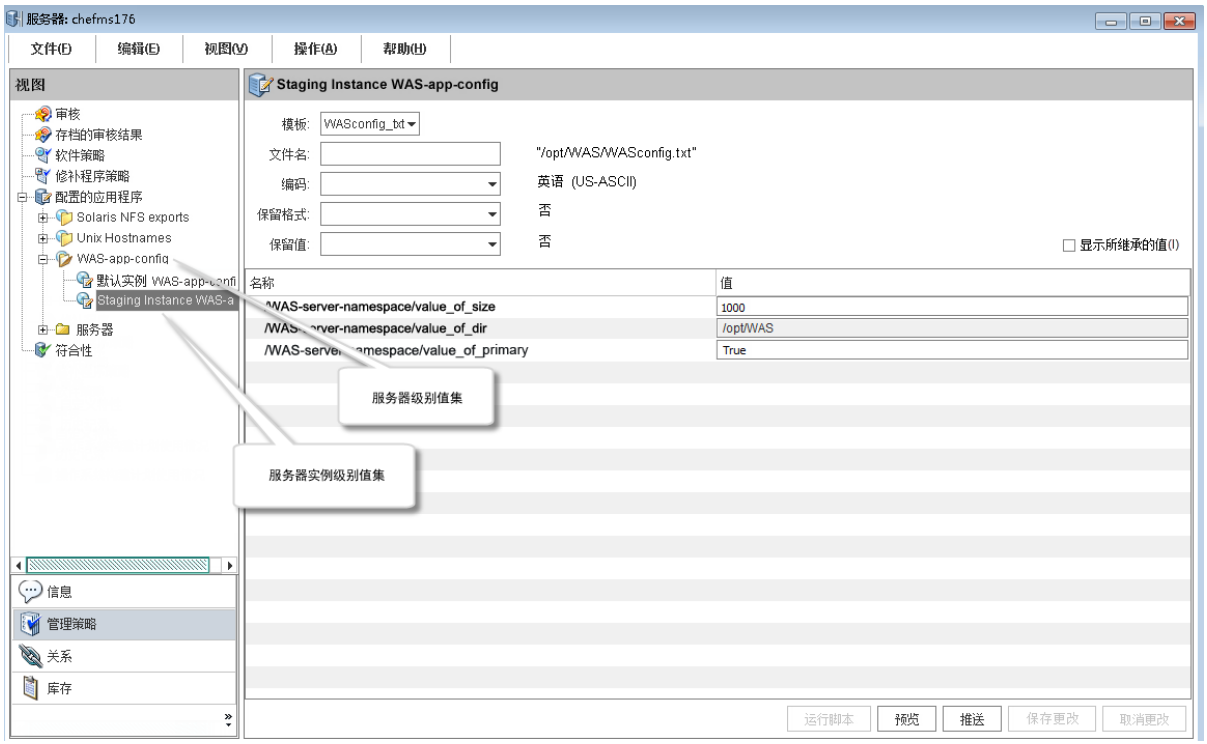
服务器级别的值集编辑器

服务器级别定义服务器的值。除非被以下任意级别覆盖，否则它将应用于指定服务器上应用程序配置的所有实例。

服务器级别和服务器实例级别的值集编辑器 (第 232 页)显示了服务器级别和服务器实例级别的值集编辑器。仅在打开已附加应用程序配置的服务器时，此视图才可用。

- 选择名为“WAS-app-config”的应用程序配置时将显示服务器级别的值集编辑器。
- 选择名为“Production Instance WAS-appconfig”或“Staging Instance WAS-appconfig”的实例均会显示服务器实例级别的值集编辑器。

服务器级别和服务器实例级别的值集编辑器



在服务器级别设置值

服务器级别定义服务器的值。除非被以下任意级别覆盖，否则它将应用于指定服务器上应用程序配置的所有实例。

在服务器级别设置值之前，必须将应用程序配置附加到服务器。请参见服务器级别和服务器实例级别的值集编辑器 (第 232 页)中的示例，其显示了服务器级别和服务器实例级别的值集编辑器。

要在服务器级别设置值，请执行以下操作：

1. 在 SA 客户端中打开服务器。必须将应用程序配置附加到服务器。有关详细信息，请参见将应用程序配置附加到服务器或设备组 (第 268 页)。
2. 选择左侧的“管理策略”选项卡。
3. 在左侧的“管理策略”窗格中打开“配置的应用程序”节点。
4. 在“配置的应用程序”节点下选择所需的应用程序配置节点。
5. 在“模板:”下拉列表中选择所需的模板文件。
6. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。
7. 选择“保存更改”。

在服务器实例级别设置值

服务器实例级别定义服务器上应用程序配置的特定实例值。它仅应用于指定服务器上的指定应用程序配置实例并覆盖所有以上级别。

在服务器实例级别设置值之前，必须将应用程序配置附加到服务器。请参见服务器级别和服务器实例级别的值集编辑器 (第 232 页)中的示例，其显示了服务器级别和服务器实例级别的值集编辑器。

要在服务器实例级别设置值，请执行以下操作：

1. 在 SA 客户端中打开服务器。必须将应用程序配置附加到服务器。有关详细信息，请参见将应用程序配置附加到服务器或设备组 (第 268 页)。
2. 选择左侧的“管理策略”选项卡。
3. 在左侧的“管理策略”窗格中打开“配置的应用程序”节点。
4. 在“配置的应用程序”节点下打开所需的应用程序配置节点。
5. 在所需的应用程序配置节点下选择所需的实例。
6. 在“模板”下拉列表中选择所需的模板文件。
7. 在文本框中输入值。通过右键单击菜单项可使用其他编辑功能。
8. 选择“保存更改”。

使用应用程序配置运行脚本

您可以将脚本添加到应用程序配置，这些脚本将在向目标服务器复制配置值之前或之后执行。

例如，您可能要添加**安装前脚本**(将停止应用程序)和**安装后脚本**(进行配置更改之后重新启动应用程序)。如果在推送或安装后脚本执行期间发生错误，则可以运行**出错后脚本**。

或者可能需要执行**数据操作脚本**来处理非文本的配置数据。在配置 IIS 服务器时，可以使用数据操作脚本将元数据库信息读取到平面文件中。在使用配置模板解析平面文件中的信息之后，可以运行安装后脚本将更新的信息写回到元数据库信息。

> **警告：** 在推送包含 JScript 或 VBScript 安装前、安装后或出错后脚本的应用程序配置时，即使脚本失败，推送也可能成功。在这些情况中，推送将忽略脚本错误。此应用程序配置不会检测脚本错误，并且能够完成推送，而不发生错误。如果计划使用这些

类型的脚本，则必须确保这些脚本没有错误，并且通过调用 WScript.Quit(<status>) 确保脚本可返回非零的退出状态。

应用程序配置脚本的类型

下表列出了可用于应用程序配置对象的脚本类型。脚本类型将指定调用脚本的时间。每种脚本类型都只能定义一个。如果定义了一个脚本，但是没有指定其中一种类型，则会将该脚本视为配置模板。即会将此脚本推送到服务器，但是不会执行该脚本。

脚本类型，指定脚本运行时间

脚本类型	描述
数据操作	在任何安装前脚本之前运行，用于解析非文本配置文件，以便 CML 模板可解析此配置文件。仅要扫描和导入由应用程序配置管理的现有文件时，数据操作脚本也十分有用。 如果此脚本失败，则不会将应用程序配置推送到服务器。
安装前	在发生实际推送之前运行。例如，安装前脚本可以停止应用程序或服务。 如果此脚本失败，则不会将应用程序配置推送到服务器。
安装后	在发生实际推送之后运行。例如，安装后脚本可以在推送之后重新启动服务。
出错后	仅在推送失败或安装后脚本失败时，才会运行。例如，出错后脚本可以恢复备份的文件。

要指定脚本类型，请执行以下操作：

1. 在 SA 客户端中，打开包含此模板的应用程序配置对象。
2. 选择“配置值”视图，以查看应用程序配置对象中包含的模板。
3. 选择模板并右键单击以显示菜单。
4. 选择脚本类型，如上表中所列。
5. 选择“文件”>“保存”菜单。

另请参见从脚本创建模板 (第 266 页)。

上表列出了可用于应用程序配置对象的脚本类型。此类型指定脚本的语法和执行环境。

脚本源类型

脚本源类型	描述
Windows .BAT 脚本	Windows 批处理命令文件。
Windows .JS 脚本	运行在 Windows 上的 Javascript 文件。
Windows .CMD 脚本	Windows 批处理命令文件。
Windows .VBS 脚本	Windows Visual Basic 脚本。
Windows .WSF 脚本	Windows 脚本文件。
Windows .PY 脚本	运行在 Windows 上的 Python 文件。
Unix .SH 脚本	Unix shell 脚本。
其他 Unix 脚本	在 Unix 上运行的任何其他脚本。

要指定脚本类型，请执行以下操作：

1. 在 SA 客户端中打开模板。
2. 选择“属性”视图。
3. 在“类型”字段中选择上表中列出的脚本类型。
4. 选择“文件”>“保存”菜单。

另请参见从脚本创建模板 (第 266 页)。

将应用程序配置推送到服务器

无论何时更改值集中的值并要将这些更改与目标服务器上的配置文件合并，都必须将应用程序配置推送到服务器。推送应用程序配置后，值集中的所有值都将替换目标托管服务器上配置文件中的值。如果目标服务器上不存在配置文件，则在推送时会在服务器上创建新的文件。此外，应用程序配置中的所有脚本基于脚本类型执行。

在推送应用程序配置时，将发生以下事件。

- SA 运行数据操作脚本(如果指定)。
- SA 从模板和值集生成目标配置文件。
- SA 备份现有配置文件。
- SA 运行安装前脚本(如果指定)。
- SA 将生成的配置文件复制到服务器。

- SA 执行安装后脚本(如果指定)。
- 如果已指定，且安装前脚本或安装后脚本失败或者复制操作失败，则 SA 将执行出错后脚本。

有关如何在“软件策略和审核”上下文中推送应用程序配置的信息，请参见软件策略中的应用程序配置 (第 244 页)。

有关使用应用程序配置中的脚本的详细信息，请参见从脚本创建模板 (第 266 页)和通过运行数据操作脚本管理非文本配置 (第 267 页)。

有关如何推送应用程序配置的说明，请参见推送应用程序配置 (第 272 页)。

> **备注：** 推送时的(列表和标量)合并序列取决于在应用程序配置继承层次结构中设置值的方式以及在应用程序配置 CML 模板中配置的序列合并模式。有关序列合并的详细信息，请参见序列聚合 (第 392 页)。

应用程序配置符合性

应用程序配置符合性可以帮助您确定附加到服务器(或服务器组)的应用程序配置的值是否与目标服务器上的配置文件值匹配。

如果目标配置文件值与应用程序配置中定义的值匹配，则将服务器视为符合。如果目标配置与应用程序配置中定义的值不匹配，则将服务器视为不符合。

SA 客户端显示应用程序配置的以下符合性状态：

- **符合：** 附加到服务器或设备组(或多台服务器和组)的应用程序配置中的所有值与目标服务器上的配置值匹配。由  图标表示。

  对于设备组，AppConfig 符合性基于组中所有服务器(以及任何子组中的服务器)的符合性状态。默认情况下，组符合性由默认阈值确定：如果组中 5% 以上的服务器具有“不符合”状态，则将整个组视为“不符合”。要更改此默认设置，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的“更改设备组的符合性设置”一节。
  - **不符合：** 应用程序配置中定义的值中至少有一个与目标服务器上配置文件中的值不匹配。由  图标表示。

    对于设备组，不符合性基于组中所有服务器(以及任何子组中的服务器)的符合性状态。默认情况下，组不符合性由默认阈值确定: 如果组中 5% 以上的服务器具有“不符

合”状态，则将整个组视为“不符合”。要更改此默认设置，请参见 HPE SSO 门户上的“Server Automation 用户指南”中的“更改设备组的符合性设置”一节。

- **已启动扫描**：当前正在计算应用程序配置符合性信息。由  图标表示。
- **需要扫描**：应用程序配置符合性信息未定义，可能由于从未运行符合性扫描(例如，在新安装程序上)，或已将自上次更改的服务器(或设备组中的服务器)上的配置信息报告给 SA 客户端。由  图标表示。
- **不适用**：应用程序配置符合性信息不适用，由短划线 (—) 表示。如果应用程序配置未附加到服务器，则会显示此信息。

您可以查看单台服务器或服务器组的应用程序配置符合性：

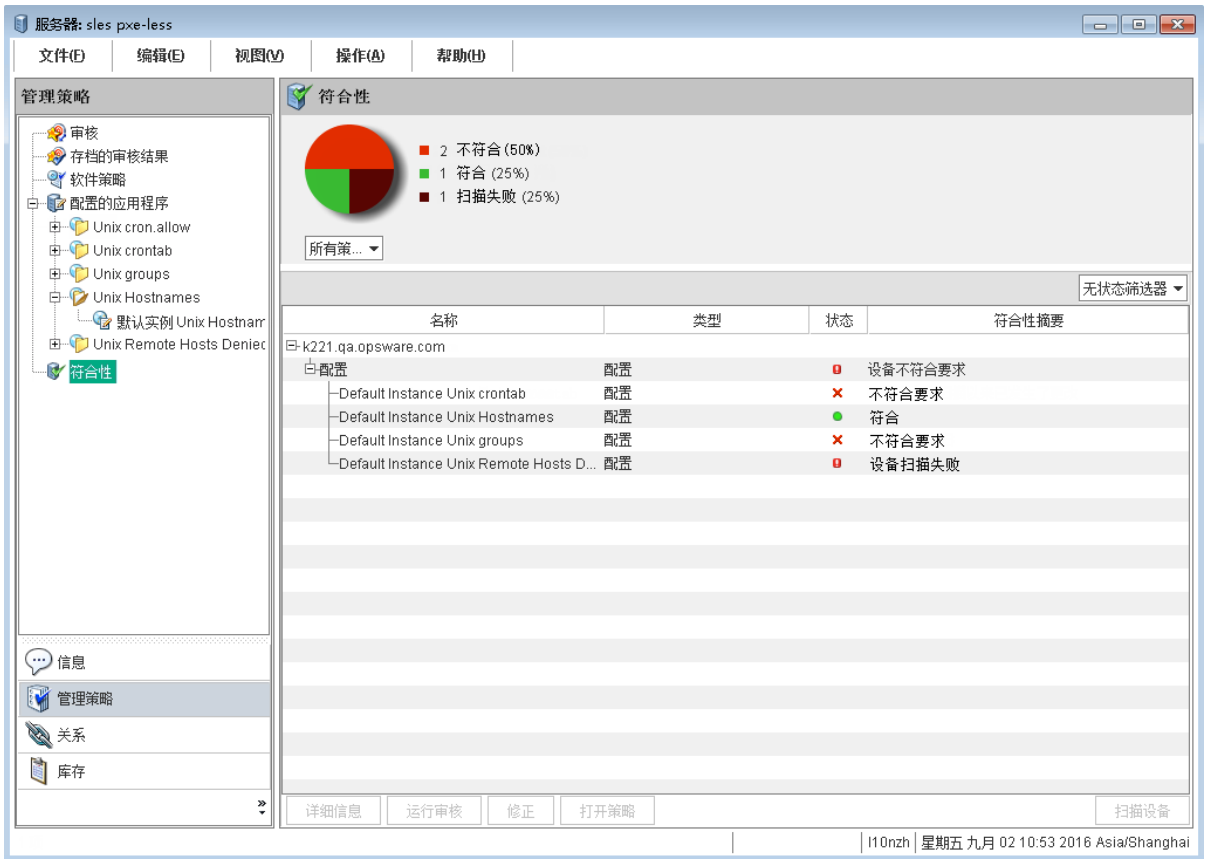
- 单台服务器的应用程序配置符合性 (第 238 页)
- 多台服务器的应用程序配置符合性 (第 239 页)

> **备注：**如果对应用程序配置进行任何更改(例如在值集编辑器中编辑其值)，则附加此配置的任何服务器或服务器组都将显示“需要扫描”的符合性状态。

单台服务器的应用程序配置符合性

对于单台服务器，符合性视图将显示附加到此服务器的所有应用程序配置的整体符合性。如果此服务器上附加了多个应用程序配置，则可以查看所有应用程序配置的聚合符合性状态以及每个配置的符合性状态。

下图显示了单台服务器的应用程序配置符合性。



如果发现应用程序配置和目标服务器上实际配置文件的任何差异，则下部窗格将显示不符合的类别。如果服务器附加了多个应用程序配置，并且应用程序配置的任意一个目标配置文件不同于应用程序配置，则服务器将处于不符合状态。

有关如何运行应用程序配置符合性扫描的详细信息，请参见在服务器中扫描应用程序配置符合性 (第 242 页)。

多台服务器的应用程序配置符合性

您可以查看多台服务器的应用程序配置符合性状态。在 SA 客户端导航窗格中选择“设备”，然后选择“设备组”或“服务器”。选择一个设备组或一组服务器，然后在“视图”菜单中选择“符合性”。此时将显示选定服务器的聚合符合性状态。

如果组中 5% 以下的服务器不符合要求，则会将附加到服务器组的应用程序配置视为符合。如果 5% 以上的服务器不符合要求，则会将聚合符合性视为不符合。通过在 SA 客户端中选择“管理”选项卡，然后选择“符合性设置”，您可以更改此百分比。

在“符合性”视图中，服务器组的详细信息窗格显示是否所有应用程序配置符合要求，但是不会展开显示单台服务器和应用程序配置的细分。

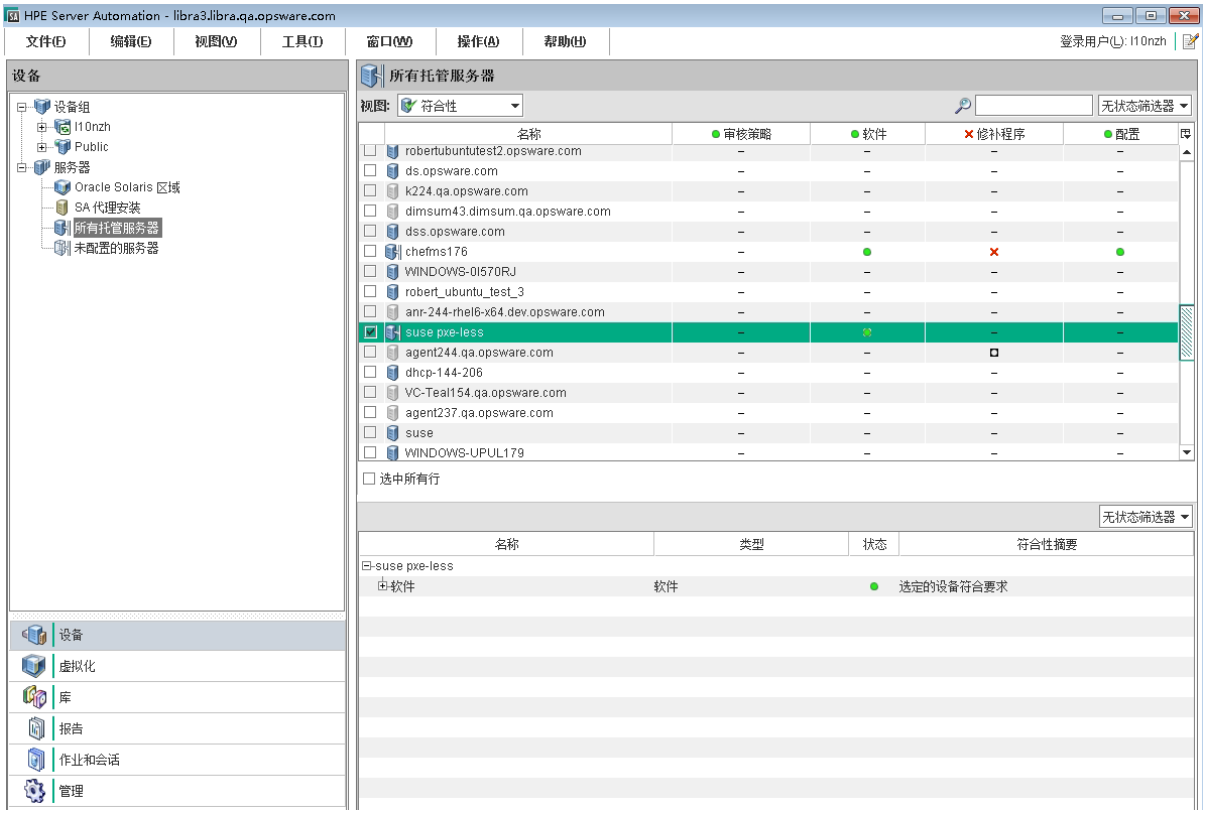
您可以使用以下方式查看服务器组应用程序配置符合性状态：

- 查看多台服务器的应用程序配置符合性 (第 240 页)。
- 查看多个设备组的应用程序配置符合性 (第 241 页)。
- 查看单个设备组的应用程序配置符合性 (第 241 页)。

查看多台服务器的应用程序配置符合性

要查看多台服务器的应用程序配置符合性，请执行以下操作：

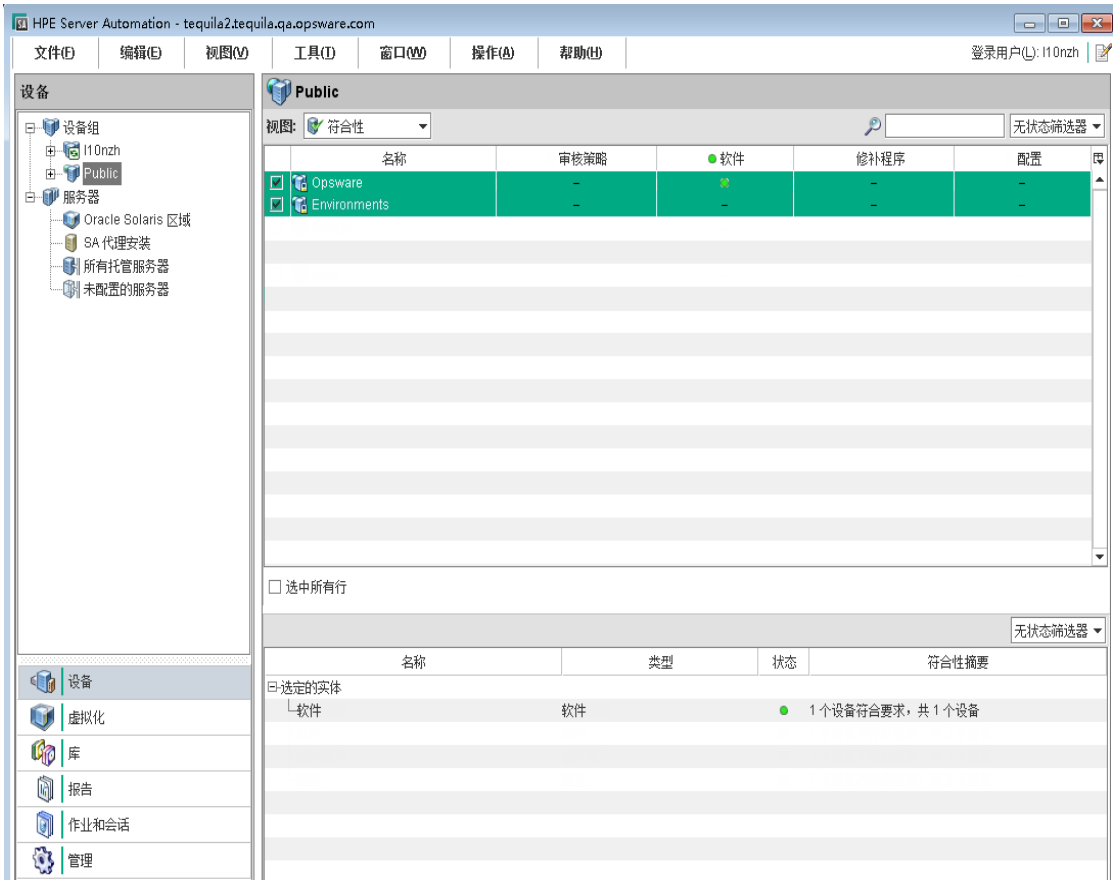
1. 从 SA 客户端的“导航”窗格中，选择“设备”>“服务器”>“所有托管服务器”。
2. 在“视图”下拉列表中，选择“符合性”。
3. 要查看多台服务器的符合性级别，请选中服务器旁边的复选框，将在底部详细信息窗格中显示选定服务器的符合性汇总，如下图所示。



查看多个设备组的应用程序配置符合性

要查看多个设备组的应用程序配置符合性，请执行以下操作：

1. 从 SA 客户端的“导航”窗格中，选择“设备”>“设备组”。
2. 选择设备组或包含设备组的文件夹。
3. 从“视图”下拉列表中，选择“符合性”。此时将显示所有组的符合性状态。
4. 要查看多个组的符合性级别，请选中服务器旁边的复选框，将在底部详细信息窗格中显示选定组的符合性摘要，如下图所示。

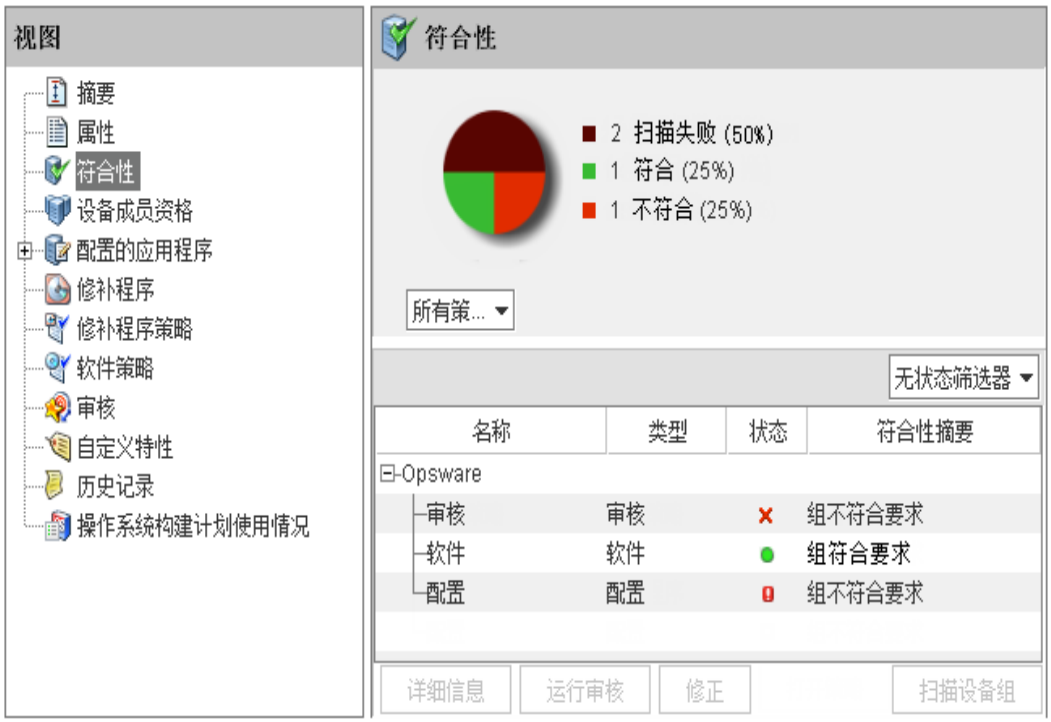


查看单个设备组的应用程序配置符合性

要查看一个设备组的应用程序配置符合性，请执行以下操作：

1. 从 SA 客户端的“导航”窗格中，选择“设备”>“设备组”。
2. 导航到所需设备组并选择该组。
3. 右键单击并选择“打开”，或者选择“操作”>“打开”。此时将显示此设备组。
4. 从“视图”窗格中选择“符合性”。此时将显示组中所有成员的每个策略类型的整体聚合符合性，而不会显示每台服务器的符合性状态，如下图所示。

设备组的应用程序配置符合性



在服务器中扫描应用程序配置符合性

在将应用程序配置推送到服务器之后，可以有意或偶尔更改或改变服务器上的此配置文件。或者在应用程序配置中定义的值可能会发生更改。当目标服务器上的配置文件的值与应用程序配置中定义的值不匹配时，会将配置文件视为不符合要求。

您可以在服务器上扫描配置符合性，确定服务器上的任何配置文件是否符合配置模板中存储的值。您可以计划定期进行扫描。

要在一台或多台服务器上扫描配置符合性，请执行以下操作：

1. 从 SA 客户端导航窗格中，选择“设备”。
2. 选择“设备组”或“所有托管服务器”。如果选择了“设备组”，则选择一个设备组将显示属于该组的服务器。
3. 在内容窗格中选择一台服务器。还可以选择多台服务器或设备组，然后对其全部扫描。
4. 在“操作”菜单中，选择“扫描”>“配置符合性”，或者选择“计划”>“配置符合性扫描”。
   - 如果选择“扫描”>“配置符合性”，则 SA 将扫描设备以确定符合性并在“扫描配置符合性”屏幕中显示状态。
   - 如果选择“计划”>“配置符合性扫描”，则 SA 将显示“计划作业”屏幕，其中您可以指定作业完成的时间以及其他作业参数。

审核应用程序配置

借助 SA，您可以审核服务器上的配置文件，确定这些文件是否符合组织的配置标准。您可以创建审核规则，用以指定如何定义服务器上的配置文件，并定期审核这些服务器以检查是否正确配置了配置文件。如果发现审核规则定义和目标配置文件值不匹配，可以通过修正这些服务器来修复此问题。

例如，要确保托管服务器上的 /etc/hosts 文件仅为特定 IP 地址定义某些主机名，您可以定义审核规则来指定可接受的主机名和 IP 地址对的列表。运行审核时，如果主机文件包含除规则中指定的值之外的任何值，则审核结果将显示错误，您可以修正此问题。

审核应用程序配置的一般过程会按照以下步骤执行操作：

1. **创建审核和审核规则**：要审核服务器上的配置文件，需首先创建审核。创建审核时，需指定配置文件所基于的源服务器(或快照或快照规范)。然后选择应用程序配置模板来构造规则。此规则将定义要在目标配置文件中检查的确切的值。对于每个审核规则，请指定目标服务器上配置文件的位置。
2. **选择目标服务器**：在审核中，选择要审核的目标服务器。可以选择单台服务器、多台服务器或服务器组。
3. **运行或计划审核**：可以计划运行一次审核，也可以定期运行审核。此外，还可以指定审核结果发送到的电子邮件地址。
4. **检查审核结果**：检查审核结果，确定目标服务器上的配置文件是否与审核规则中定义值匹配。如果存在差异，您可以比较规则和目标文件以查看差异，从而确定如何修正服务器。

5. **修正服务器**：要修复审核结果中找到的差异，可以修正服务器或者任意或全部规则，确保目标配置与规则匹配。

有关使用审核和快照的详细信息，请参阅《SA 10.50 管理指南》中的“审计和修正”一节。

软件策略中的应用程序配置

应用程序配置在用于软件策略中时是一个非常强大的工具。软件策略定义应用程序(包括要在服务器上安装的所有程序包、修补程序、脚本和其他对象)的理想状态以及在服务器上设置应用程序配置文件的方式。在托管服务器上安装软件策略时，SA 会将所有内容应用到策略的目标服务器，包括在应用程序配置中定义的所有值。

通过使用 SA 客户端中的符合性视图，可以查看从此策略安装的软件的符合性状态。例如，如果某个人从软件策略删除修补程序、在服务器上安装新的程序包，或者更改策略中定义的其中一个配置文件，则策略将在符合性视图中显示为不符合。要确保正确安装和配置应用程序，您可以修正服务器。

有关使用和创建软件策略的详细信息，请参见《SA 10.50 用户指南》中的“软件管理”一节。

要在软件策略中使用应用程序配置，请执行以下操作：

1. **定义应用程序**:在构建软件策略之前，应用程序专家收集组成此应用程序的所有必需的程序包和修补程序。此外，还收集用于定义和管理与应用程序关联的配置文件的配置模板。
2. **将程序包和修补程序导入 SA**：在定义软件策略的组件之后，将组成应用程序的所有程序包和修补程序导入 SA 库中，以便将其置于软件策略中。
3. **创建应用程序配置和设置值**:定义将用于生成配置文件的配置值。例如，如果创建软件策略以部署 Apache Web 服务器，则应用程序专家会使用值集编辑器为 httpd.conf 文件定义默认值。例如，将任何安装前或安装后脚本添加到应用程序配置，以便在软件策略修正期间推送应用程序配置之后重新启动 Apache 服务。
4. **测试应用程序配置**：在将应用程序配置添加到软件策略并将应用程序部署到服务器之前，最好将应用程序配置附加到服务器并确保在创建软件策略之前应用程序可以正常运行。可以预览向服务器的配置推送以验证配置的正确性。
5. **创建软件策略**：在定义和创建软件策略的所有组件并将其导入 SA 中之后，应用程序专家可以创建软件策略，用以指定要安装的软件以及安装其组件(包括所有修补程序、程序包和应用程序配置)的顺序。在将软件策略保存到 SA 库中之后，负责部署、测试和管理应用程序的系统管理员便可以访问此策略。

6. **将策略附加到服务器或服务器组**：在创建和保存软件策略之后，系统管理员可以将策略附加到设备组中的服务器或服务器组。
7. **修正服务器以安装软件**：通过修正服务器上的软件策略，系统管理员可以将软件部署到一台或多台服务器。修正过程可以确保以策略中指定的顺序将策略中定义的所有内容部署在目标服务器上。修正之前，管理员可在作业中预览应用程序配置。预览步骤提供在服务器实例级别为模板变量定义 appconfig 值的选项。
8. **测试应用程序和遍历更改**：系统管理员使用软件策略修正安装应用程序之后，在将应用程序投入生产环境之前，应测试应用程序以确保其可以正常运行并且包含正确的组件。此外，应检查其配置文件所影响的应用程序的每个部分，确保已正确配置该部分。
9. **使用应用程序**：在部署和使用此应用程序之后，系统管理员可以执行持续的管理和维护任务，例如运行软件符合性扫描，确定已部署此应用程序的服务器的符合性状态、修正不符合的服务器以及生成软件符合性报告。

有关使用软件策略的详细信息，请参见《SA 10.50 用户指南》。

应用程序配置任务

本节描述了执行应用程序配置任务的方式：

应用程序配置和模板入门：

- 创建应用程序配置 (第 246 页)
- 创建配置模板 (第 247 页)
- 使用可视化编辑器创建配置模板 (第 249 页)

编辑和管理应用程序配置模板：

- 使用可视化编辑器编辑配置模板 (第 255 页)
- 编辑模板中的 CML 或 XML (第 262 页)
- 导入和验证模板文件 (第 263 页)
- 查看配置模板源 (第 264 页)
- 将模板添加到应用程序配置或从应用程序配置删除模板 (第 264 页)
- 指定应用程序配置中的模板顺序 (第 265 页)

在应用程序配置中使用脚本：

- 从脚本创建模板 (第 266 页)
- 通过运行数据操作脚本管理非文本配置 (第 267 页)
- 手动运行数据操作脚本 (第 267 页)

附加和分离应用程序配置：

- 将应用程序配置附加到服务器或设备组 (第 268 页)
- 从服务器或设备组分离应用程序配置 (第 270 页)

推送和管理应用程序配置：

- 推送应用程序配置 (第 272 页)
- 计划应用程序配置推送 (第 274 页)
- 将配置文件恢复到之前的状态 (第 274 页)
- 搜索和筛选作业结果 (第 277 页)
- 将配置模板与目标配置文件进行比较 - 预览 (第 278 页)
- 比较配置模板 (第 279 页)
- 对配置文件元素名称进行本地化 (第 280 页)

创建应用程序配置

应用程序配置是可容纳配置模板文件和(可选)脚本(将应用程序配置推送到服务器时会执行的脚本)的容器。有关详细信息，请参见应用程序配置管理 (第 218 页)。

要创建应用程序配置，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开应用程序配置节点：
   - 打开“配置”节点。
   - 打开与应用程序配置相关的操作系统组。
   - 选择操作系统。

> **备注：**
> 应用程序配置可应用于多个操作系统。可在后面的步骤中进行指定。

3. 从菜单选择“操作”>“新建”打开一个新配置窗口，您可在其中指定配置属性和内容。
4. 在“属性”视图中，指定配置的名称和描述。此外，还需指定以下内容：
   - **位置：指定 SA 库中存储应用程序配置的位置。**
   - **版本**：版本可以是用于跟踪应用程序配置更改的任何字符串。版本不会自动增加。
   - **OS**：指定应用该应用程序配置的操作系统。
     - 仅装有指定操作系统的服务器才能使用此应用程序配置。
     - 此应用程序配置中只能包含适用于在此处定义的所有操作系统的模板；即适用于此配置的 OS 必须是适用于这些模板的 OS 的子集。
   - **可用性**：使用此设置可跟踪已经过测试并准备使用的应用程序配置以及尚未测试或已弃用的应用程序配置。此设置不会更改应用程序配置可以实现的功能。
   - **本地化文件**：选择“+”按钮可添加模板，以便生成本地语言的配置文件。有关详细信息，请参见对配置文件元素名称进行本地化 (第 280 页)。
5. 在“配置值”视图中，选择“操作”>“添加”菜单或 "+" 按钮将模板添加到应用程序配置中。
   - 使用“-”按钮可删除选定配置模板。
   - 使用向上和向下箭头可更改配置模板的安装顺序。
   - 选择“模板预览”视图可查看选定模板文件的内容。
   - 选择“文件值”视图可查看要置于选定模板文件以生成配置文件的值。
   - 选择“文件预览”视图可查看使用当前值集的选定配置文件。
6. 选择“文件”>“保存”以保存应用程序配置。

创建配置模板

配置模板与本地应用程序配置文件类似，但是它的变量部分可以通过使用配置建模语言 (CML) 和指令进行“模板化”，以便在配置文件和值集之间移动值。请参见配置模板和脚本模板 (第 218 页)。

此外，还必须采用 CML 模板格式编写要使用此应用程序配置执行的任何脚本。有关详细信息，请参见从脚本创建模板 (第 266 页)。

要创建配置模板，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并选择将使用模板文件的操作系统。请注意，模板可应用于多个操作系统。可在后面的步骤中进行指定。
3. 选择“操作”>“新建”菜单。此时将显示“模板”屏幕，您可以在其中定义模板的属性。
4. 选择“属性”视图并输入模板文件的名称和描述以及以下信息：
   - **位置**：指定 SA 库中用于存储模板的位置。
   - **版本**：版本可以是用于跟踪模板更改的任何字符串。版本不会自动增加。
   - **类型**：指定是模板文件、脚本还是本地化文件：
     - 模板文件是配置文件的模型。
     - 脚本可在向服务器推送配置文件之前或之后执行。有关详细信息，请参见 从脚本创建模板 (第 266 页)。
     - 本地化文件用于针对不同区域设置自定义的配置文件。有关详细信息，请参见 对配置文件元素名称进行本地化 (第 280 页)。
   - **分析器语法**：选择模板使用的语法类型：
     - 适用于除 XML 文件之外的所有文本配置文件以及所有脚本文件的 CML 语法。
     - 适用于采用 XML 编写的配置文件的 XML 语法。
     - 适用于同时采用 XML 和 DTD 编写的配置文件的 XML DTD 语法。
     - OS 配置的自定义特性语法
   - **OS**：指定应用该应用程序配置的操作系统。
     - 仅装有指定操作系统的服务器才能使用此模板。
     - 只有适用于一个或多个此处所列操作系统的应用程序配置才能使用此模板。
   - **可用性**：使用此设置可跟踪已经过测试并准备使用的模板以及尚未测试或已弃用的模板。此设置不会更改模板可以实现的功能。您可以将此字段值用作搜索条件。
   - **可审核**：要为模板文件启用审核时，可设置此字段。有关详细信息，请参见 审核应用程序配置 (第 243 页)。
5. 选择“内容”视图。
6. 在模板编辑器中直接输入 CML 或 XML 或 XML DTD 文本。有关编辑操作和语法突出显示的信息，请参见编辑模板中的 CML 或 XML (第 262 页)。有关 CML 和 XML 的详细信息，请参见 CML 引用 (第 352 页)和管理 XML 配置文件 (第 282 页)。
7. 选择“验证”可分析 CML 或 XML 语法并检查错误。
8. 选择“文件”>“保存”以保存模板。

9. 将模板添加到应用程序配置对象。请参见 将模板添加到应用程序配置或从应用程序配置删除模板 (第 264 页)。

使用可视化编辑器创建配置模板

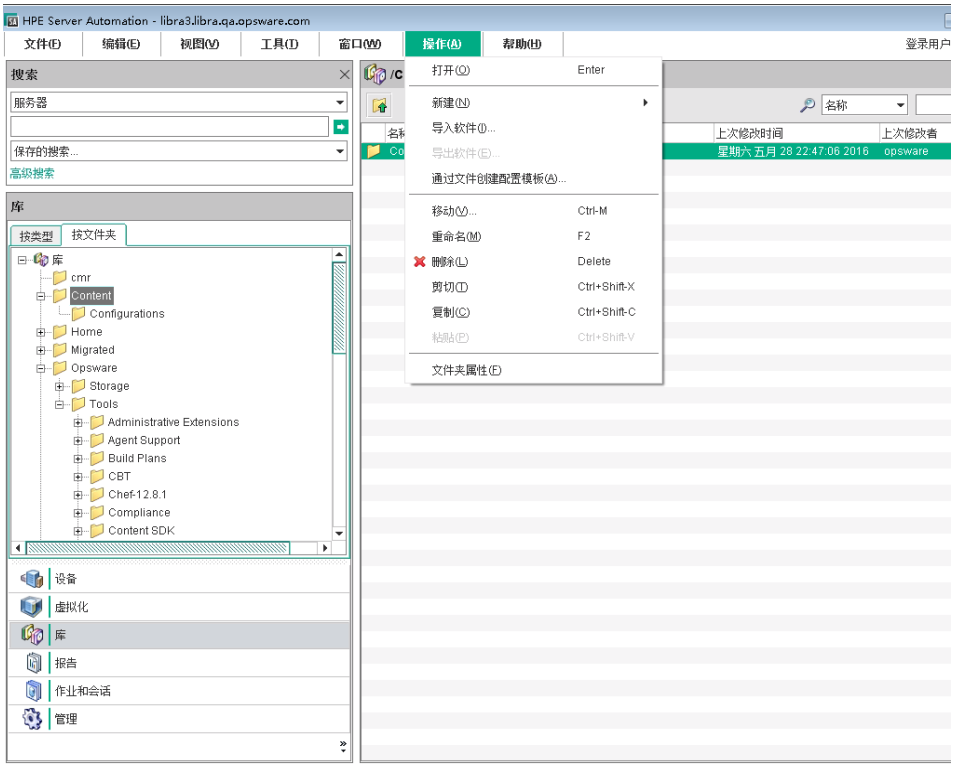
可视化编辑器模式支持您创建简单的应用程序模板，而无需了解 CML 或理解整个配置文件。如果要参数化小部分配置文件以将更改推送到服务器，此工具将十分有用。

1. 首先，可以从现有文件开始，使用可视化编辑器创建应用程序配置模板，例如：
   - SA 托管服务器上的工作配置文件
     请参见 从工作配置文件创建模板 (第 250 页)。
   - 从本地 PC 导入的配置文件示例
     请参见 导入并创建模板 (第 249 页)。
2. 然后，可视化编辑器模式会提供一个用户界面，用于编辑简单的应用程序配置模板，而无需使用 CML。
   - 只需标记在配置推送期间应替换的配置选项，就会显示此窗体视图。标记的选项将基于基础模板填充参数名称、显示名称和数据类型字段。此窗体视图允许您根据需要修改参数的详细信息。请参见 使用可视化编辑器编辑配置模板 (第 255 页)。
3. 在创建模板之后，可以选择立即生成应用程序配置实例。
   - 通过托管服务器上的工作配置文件，您可以查看兼容模板的列表、打开模板和立即生成应用程序配置实例并将其附加到服务器。配置实例可在值集编辑器中打开，在此编辑器中，您可以修改值并将更改推送到服务器。请参见管理配置文件 (第 258 页)。

> **备注：** 在 SA 中执行特定操作的功能由权限设置控制。此外，为了通过服务器文件系统访问可视化编辑器，您需要具有 OGFS 权限来读取此文件系统。要获取其他权限，请与 SA 管理员联系。有关详细信息，请参见《SA 10.5 管理指南》。

导入并创建模板

1. 从 SA 客户端导航窗格中，通过选择“库”>“按文件夹”或 >“按类型”访问要从 PC 中导入配置文件的文件夹，然后导航到所需的文件夹。
2. 从“操作”菜单中，选择“通过文件创建配置模板...”。



3. 查找并选择要导入的配置文件，然后单击“打开”。
4. 将在可视化编辑器模式中打开选定文件的模板。

请参见可视化编辑器界面 (第 252 页)和使用可视化编辑器编辑配置模板 (第 255 页)。

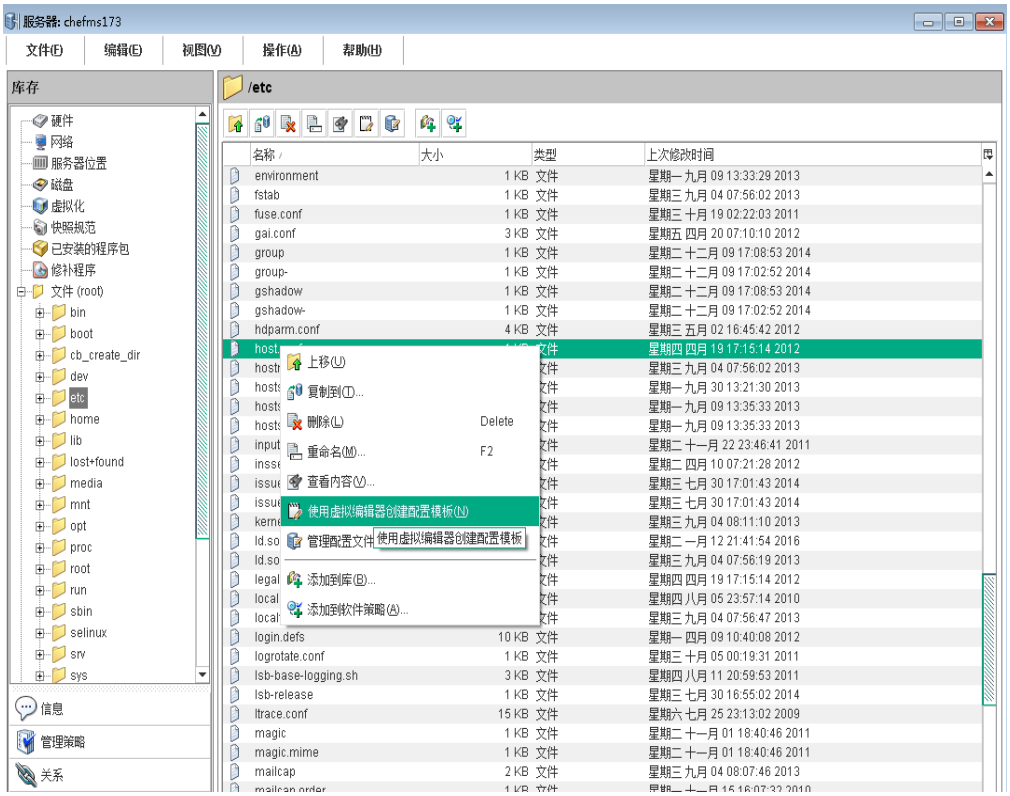
从工作配置文件创建模板

要使用可视化编辑器从工作配置文件创建配置模板，请执行以下操作：

1. 打开服务器浏览器：
   a. 在 SA 客户端导航窗格中，访问托管服务器或设备组的列表：
      i. 选择“设备”>“服务器”>“所有托管服务器”，以查看服务器列表。
      ii. 选择“设备”>“设备组”，以查看设备组列表。
   b. 在内容窗格中，选择要打开的服务器或设备组。
   c. 从“操作”菜单中，选择“打开”。
2. 导航到“服务器浏览器”>“库存”>“文件”。

3. 系统提示时，请选择服务器文件系统相应的根路径 (通常，针对 Unix 选择 root，针对 Windows 选择 Administrator。)。

4. 右键单击配置文件并选择“使用虚拟编辑器创建配置模板”。

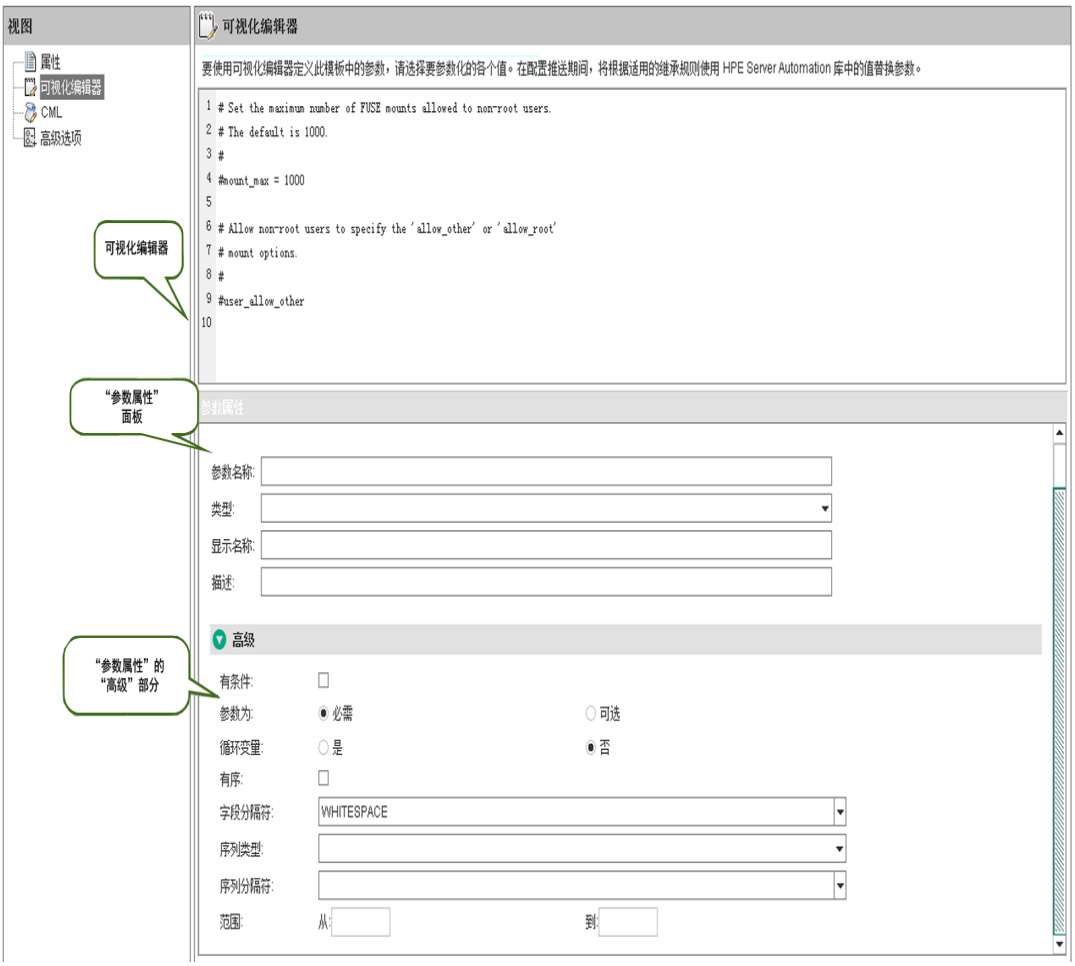


将在可视化编辑器模式中打开选定配置文件的模板。请参见 可视化编辑器界面 (第 252 页)。

可视化编辑器界面

可视化编辑器界面在拆分面板中显示基础应用程序配置模板的详细信息，其中，下方面板的可编辑窗体视图中显示选定参数的详细属性。首次进入“可视化编辑器”窗口时，会显示“参数属性”面板。

可视化编辑器界面



“参数属性”面板

在“可视化编辑器”中选择一个值，然后单击“参数属性”面板的“参数名称”字段时，将基于基础配置文件中的值填充“参数属性”面板中的字段。

可编辑位于用户友好的窗体视图中的参数属性字段。

“参数属性”窗体

“参数属性”的“高级”部分:使用“高级”部分可指定有关参数的其他选项，例如参数是必需(默认)还是可选、是循环变量、有序列表、序列(指定类型和分隔符)还是范围。为数字类型(例如整数、十进制或端口)指定范围。(数字类型可以是序列的一部分。)

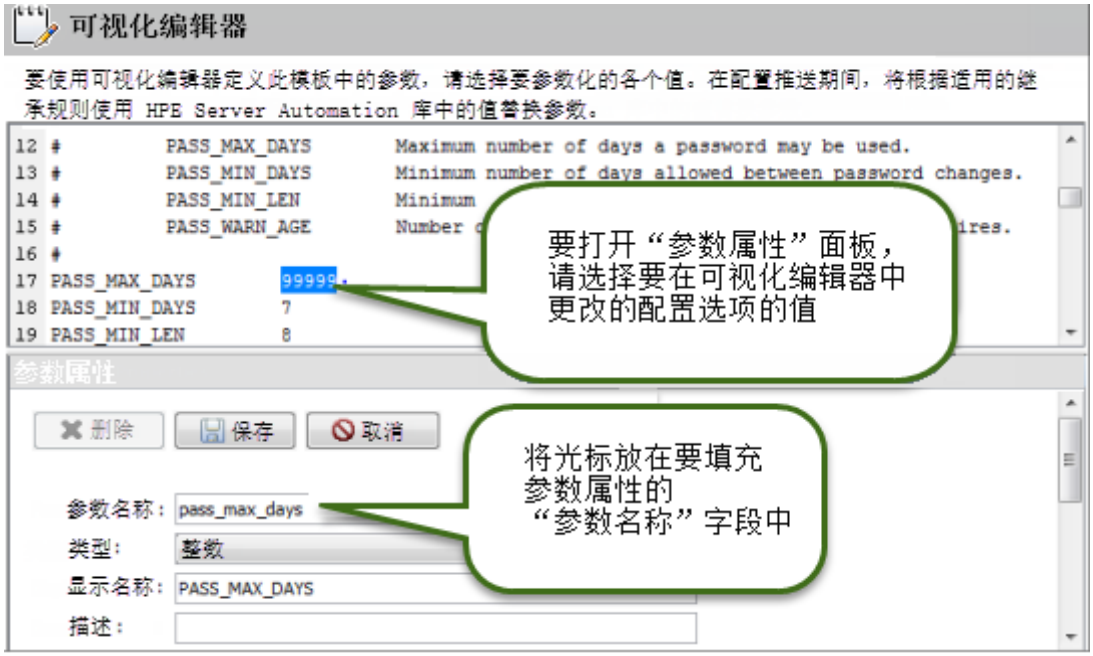
“参数属性”窗体的“高级”部分

“可视化编辑器”界面的运行方式:

在“可视化编辑器”中，选择在推送配置时要替换的配置选项的值，此时会显示此窗体视图。

标记的选项将基于基础配置文件中的数据填充“参数属性”面板中的“参数名称”、“显示名称”和“类型”(数据类型)字段。此窗体视图允许您根据需要修改参数的详细信息。

在可视化编辑器中编辑配置模板



有关使用可视化编辑器编辑配置模板的说明，请参见使用可视化编辑器编辑配置模板 (第 255 页)。

可视化编辑器中的“高级选项”

“高级选项”视图允许用户在模板中修改全局 CML 选项。要访问此视图，请在模板的“视图”窗格中选择“高级选项”。

默认情况下已填充这些值，无需用户提供输入；但是可以对其进行编辑:

- 定义三个必填字段:“文件名关键字”、“默认文件名”和“命名空间”
- 指定生成的模板是部分模板还是完整模板 (默认情况下，部分模板意味着仅会替换指定的值。)
- 推送超时值和其他 CML 格式化和分析选项。

> **备注:** 可视化编辑器限制：可视化编辑器仅支持部分 CML 内容。如果采用可视化编辑器不支持的某种方式直接修改 CML，则会显示一个警告消息，提示不兼容。您可以选择保留所做的更改并禁用可视化编辑器，也可以选择放弃更改并继续使用可视化编辑器。

使用可视化编辑器编辑配置模板

除对可视化编辑器流的基本说明以外，此部分还包含有关使用可视化编辑器编辑配置模板的提示信息。有关打开可视化编辑器的说明，请参见使用可视化编辑器创建配置模板 (第 249 页)。

要编辑配置模板，请执行以下操作：

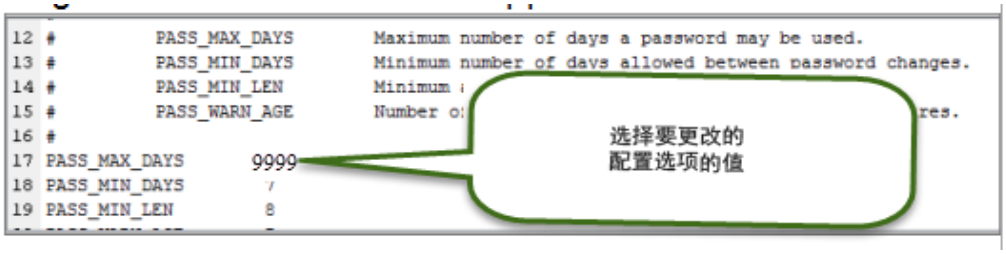
1. 打开可视化编辑器。
   请参见导入并创建模板 (第 249 页)或从工作配置文件创建模板 (第 250 页)。

2. 定义要推送的参数。
   有关示例，请参见创建简单参数 (第 256 页)或创建序列参数 (第 257 页)。
3. 查看“属性”选项卡，指定模板的名称和存储此模板的位置。
4. 单击“保存”以保存此模板。

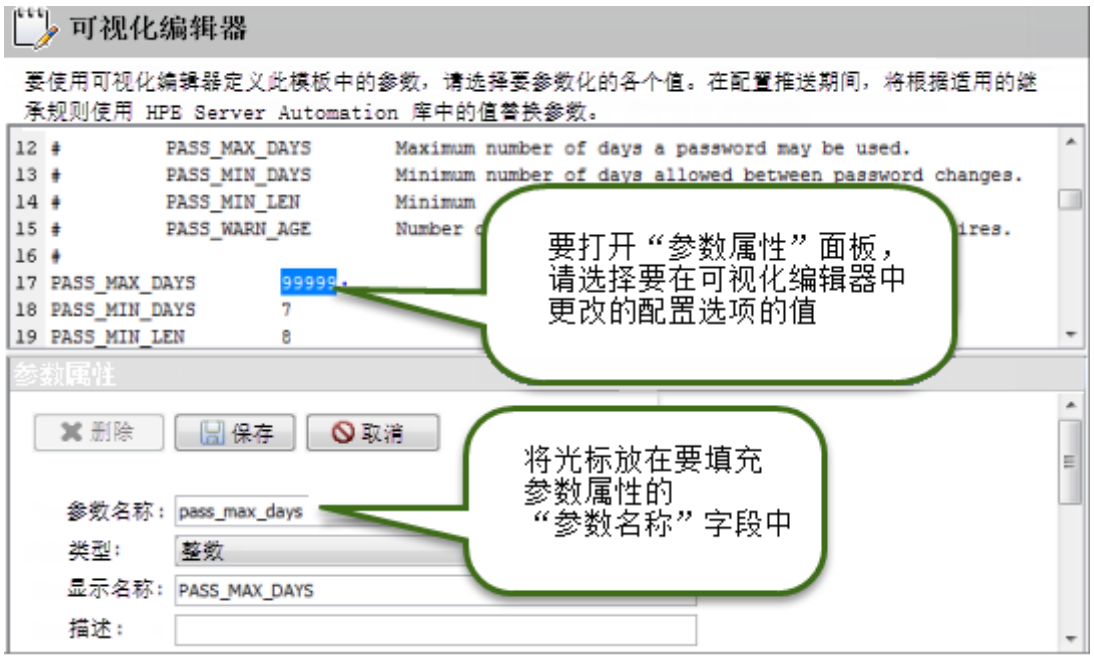
创建简单参数

要创建简单参数，请执行以下操作：

1. 选择在推送配置时要替换的配置选项，此时会显示此窗体视图。

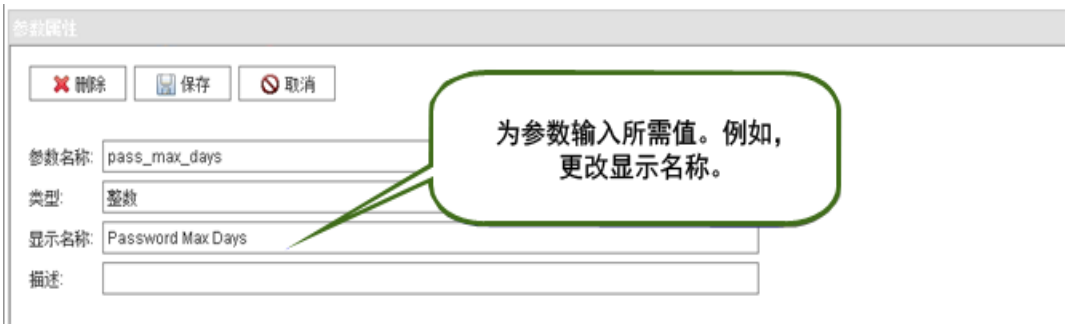


2. 在“参数属性”面板中，单击“参数名称”字段，标记的选项将基于基础模板填充“参数名称”、“显示名称”和“数据类型”字段。

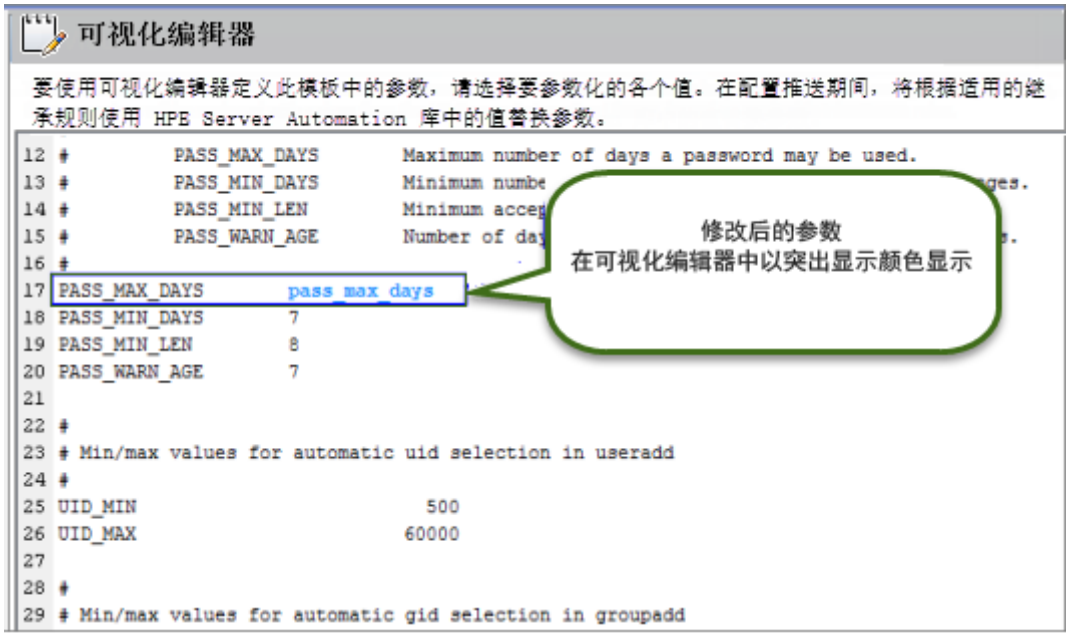


、

3. 为“参数名称”和“显示名称”输入所需的值，然后单击“保存”以保存此参数。



4. 新建的参数将在可视化编辑器中突出显示为蓝色

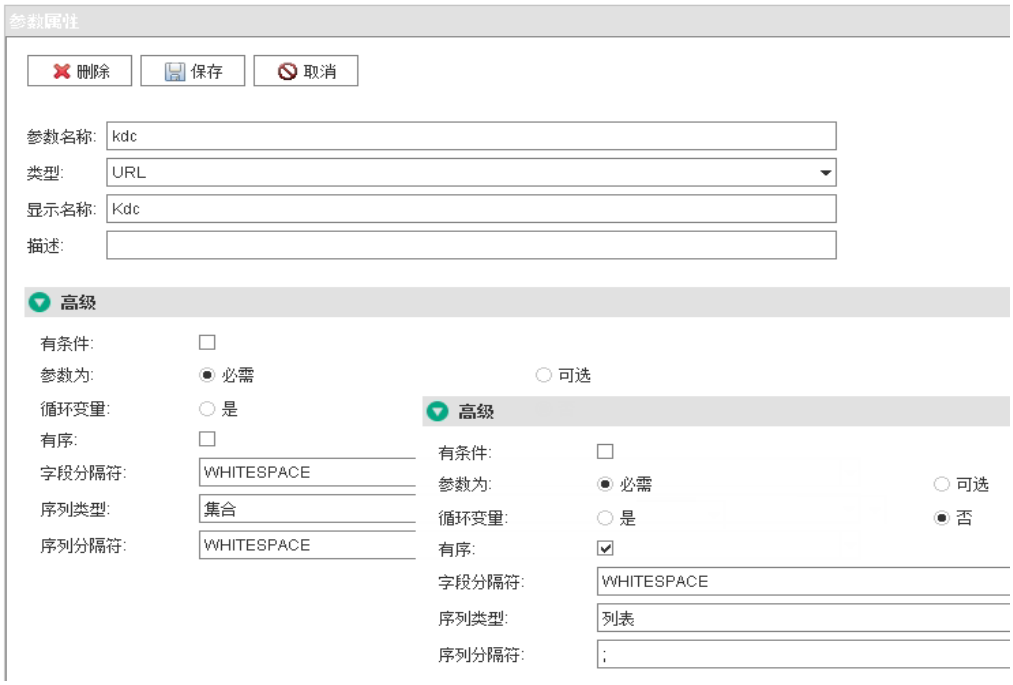


创建序列参数

要创建序列参数，请执行以下操作：

1. 执行的步骤与创建简单参数的步骤相同 (请参见创建简单参数 (第 256 页))。
2. 在“高级”部分的“序列类型”字段中指定一个值：“集合”或“列表”。

“列表”必须按特定序列保存元素，“集合”不能包含任何重复元素。



3. 指定“序列分隔符”。这是用于分隔列表中的值的字符类型。

在“可视化编辑器”面板中，将创建序列参数，方法是使用相同的参数名称突出显示配置文件中与此参数匹配的所有元素。

管理配置文件

通过管理配置文件功能，您可以使用现有模板对工作配置进行快速建模。

> **备注：**为启动此操作，您必须具有 OGFS 权限来访问文件系统根目录。此外，还必须具有正确的权限来访问主文件夹，才能完成此操作。

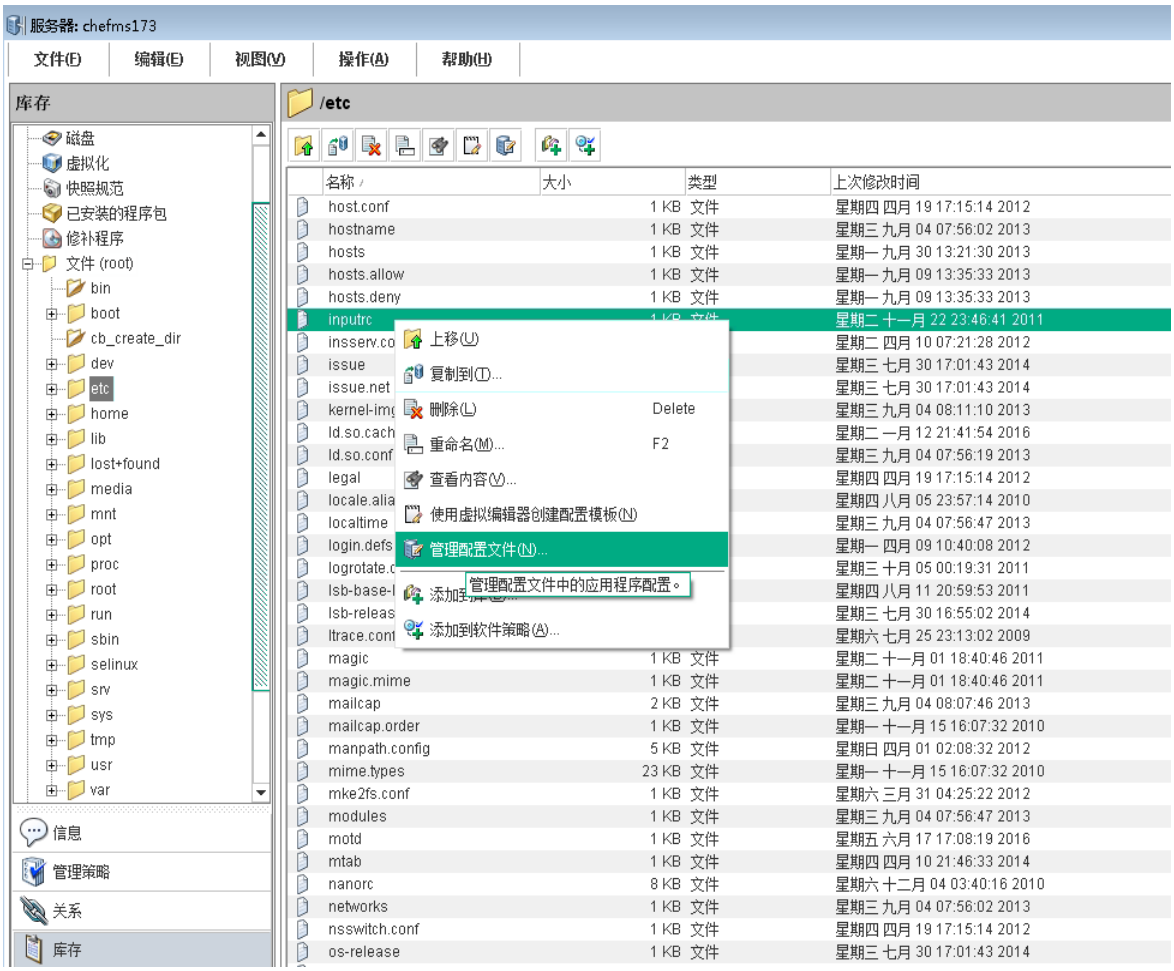
要使用现有模板对工作配置进行建模，请执行以下操作：

1. 打开服务器浏览器：
   a. 在 SA 客户端导航窗格中，访问托管服务器或设备组的列表：
      i. 选择“设备”>“服务器”>“所有托管服务器”，以查看服务器列表。
      ii. 选择“设备”>“设备组”，以查看设备组列表。
   b. 在内容窗格中，选择要打开的服务器或设备组。
   c. 从“操作”菜单中，选择“打开”。

2. 导航到“服务器浏览器”>“库存”>“文件”。(此步骤要求您具有 OGFS 权限。)

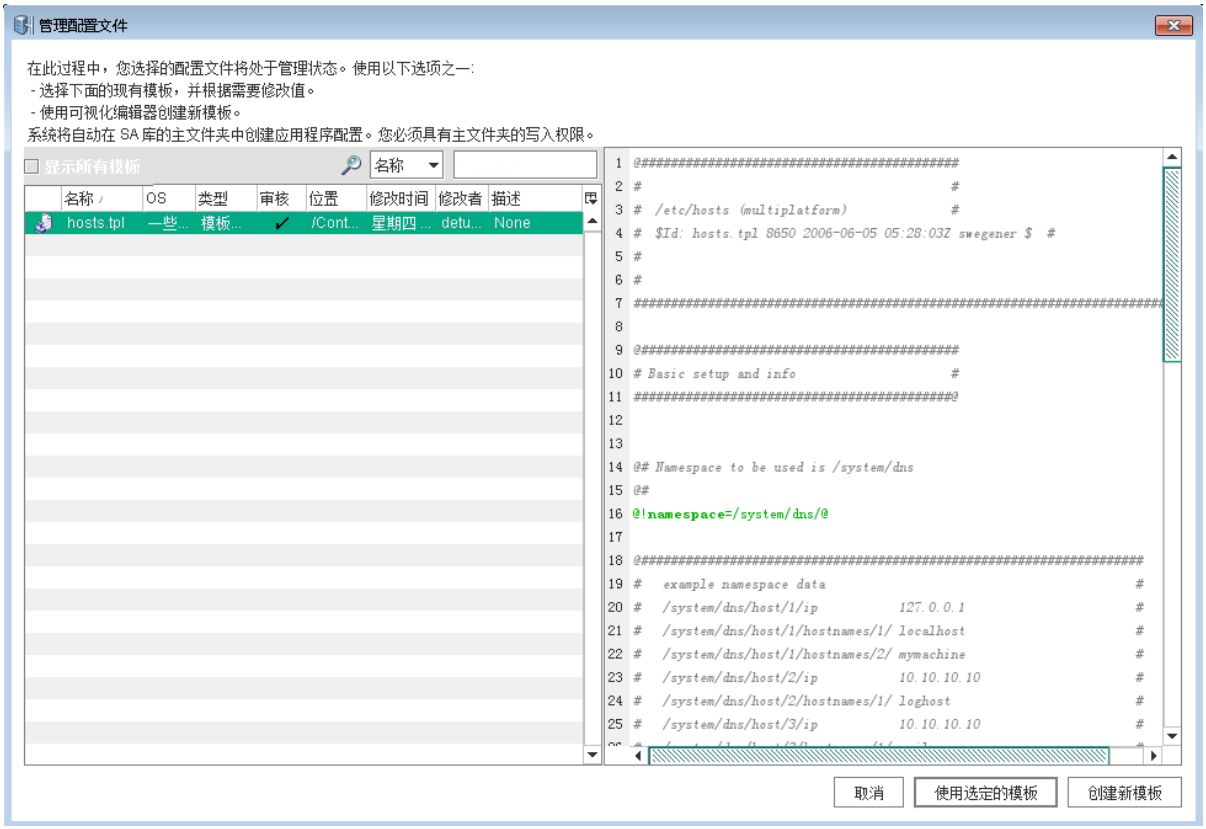
   系统显示提示信息时，请选择服务器文件系统相应的根路径 (通常，针对 UNIX 选择 root，针对 Windows 选择 Administrator。)

3. 右键单击配置文件并选择“管理配置文件”。



   此时将列出 SA 中与配置文件名和路径匹配的现有模板作为参考。将仅提供与服务器平台匹配的模板。

4. 选择模板，以在预览窗格中查看其内容



其他选项:

a. 选中“显示所有模板”复选框展开列表，以包括与服务器平台匹配的所有模板。

b. 单击“取消”关闭此窗口并返回服务器文件系统目录。

c. 单击“创建新模板”，将使用可视化编辑器基于选定的配置文件来创建新的模板。(请参见从工作配置文件创建模板 (第 250 页)或使用可视化编辑器编辑配置模板 (第 255 页)。)

   i. 如果选择“创建新模板”，则在保存和关闭此模板时，系统会提示您是否要生成应用程序配置以及将实例附加到服务器。

   ii. 可选：如果仅要保存此模板以便稍后重新访问，请单击“否”。此模板将保存在“属性”选项卡指定的目录中。

   iii. 单击“是”将立即生成应用程序配置实例并将其附加到服务器。此配置实例将显示在值集编辑器中。

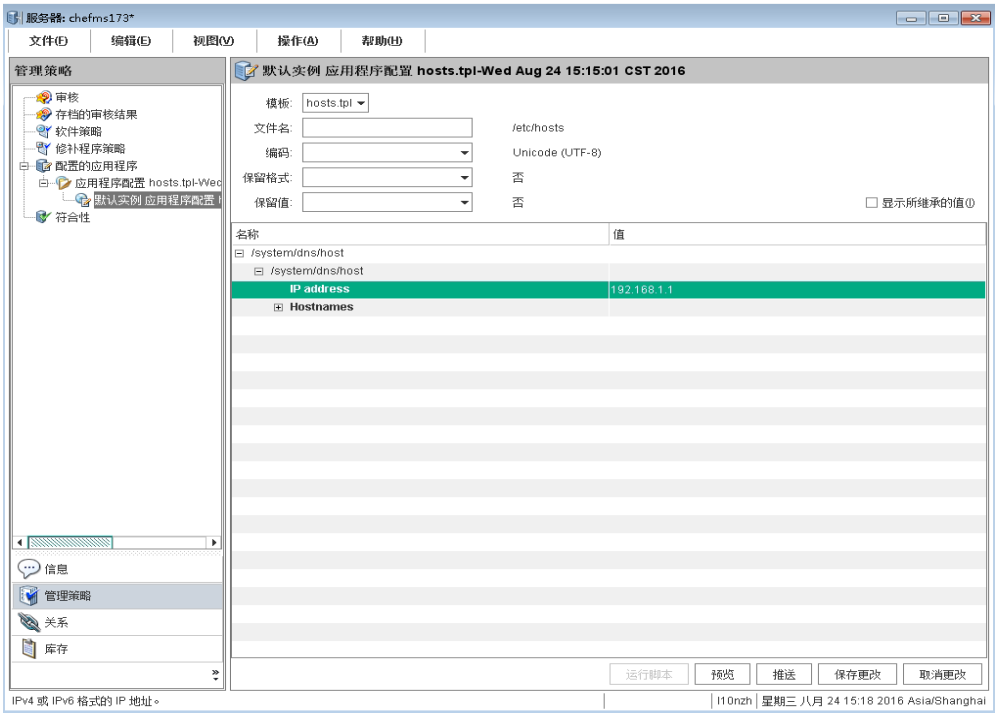
5. 单击“使用选定的模板”，可使用选定的模板创建新的应用程序配置。(此步骤要求您有权访问主文件夹。)

新建的配置文件将位于 SA 库的主文件夹中，并且会将其中一个实例附加到托管服务

器。

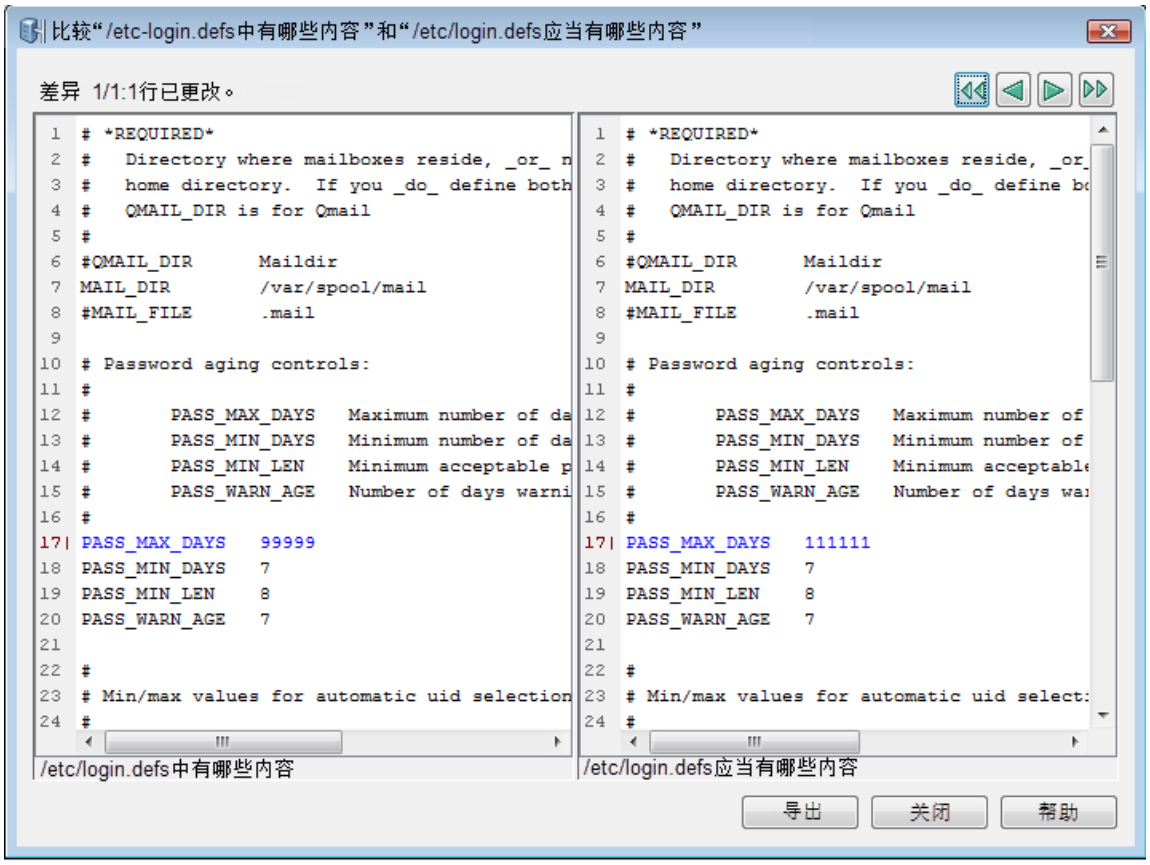
此时会显示一个确认消息，告知已创建配置文件并将其附加到服务器。

6. 单击“确定”。此配置实例将显示在值集编辑器中。



7. 输入此实例的值集数据 (请参见在值集编辑器中设置值 (第 224 页))。

8. (可选)单击“预览”，可并列预览与当前值比较的更改。单击“关闭”可返回处于“编辑”模式中的应用程序配置实例视图。



9. 单击“推送”可将更改推送到服务器。

编辑模板中的 CML 或 XML

可以在“内容”视图中编辑模板的 CML 或 XML。模板编辑器可提供编辑操作和语法突出显示，如下所述。

要编辑模板的 CML 或 XML，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并选择模板文件所在的操作系统。请注意，模板可应用于多个操作系统。
3. 选择一个模板。

4. 选择“操作”>“打开”菜单，或右键单击然后选择“打开”菜单或按 Enter 键。此时将显示包含选定模板的“模板”屏幕。
5. 选择“内容”视图。此时将显示模板的内容。
6. 在模板编辑器中直接输入 CML 或 XML 文本。

   CML 语法使用不同颜色突出显示以便于阅读：
   - 绿色:CML 指令以绿色显示。CML 关键字以绿色粗体显示。
   - 蓝色：将替换为值集中的实际值的变量以蓝色显示。
   - 黑色：固定文本以黑色显示。
   - 灰色：注释以灰色显示。

   CML 编辑器还提供以下编辑操作：
   - 右键单击 CML 文本，然后使用剪切、复制和粘贴操作。
   - “操作”菜单提供查找、替换、撤消和重做操作。
   - “验证”按钮和“操作”>“验证”菜单检查模板的语法并报告所有错误。

有关 CML 和 XML 的详细信息，请参见 CML 引用 (第 352 页)和管理 XML 配置文件 (第 282 页)。

导入和验证模板文件

可以使用文本编辑器编写 CML 模板或 XML 模板并将其导入 SA 库中，以便在应用程序配置中使用。此外，还可以在导入模板之前从 SA 验证此模板 (请参见配置模板和脚本模板 (第 218 页))。

对于 Windows 服务器上使用 UTF-8 编码的配置文件，此配置文件的前三个字符可能包含字节顺序标记 (BOM)。如果将此文件导入应用程序配置模板，则在导入此文件之后，BOM 将显示在此模板中。如果希望应用程序配置模板中不包含此 BOM，则可在将此配置文件上载到模板之后删除它。

SA 客户端中不支持使用 UTF-16 编码。

要验证和导入模板文件，请执行以下操作：

1. 在 SA 客户端中选择导航窗格，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并选择将使用模板文件的操作系统。请注意，模板可应用于多个操作系统。可在后面的步骤中进行指定。
3. 选择“操作”>“验证模板...”菜单。
4. 查找并选择模板文件，选择相应编码，然后选择“打开”。SA 将检查模板的语法并报告结果。如果文件中存在错误，请更正它们并重新进行验证。
5. 选择“操作”>“导入模板...”菜单。
6. 查找并选择模板文件，选择相应编码，然后选择“打开”。请注意，SA 客户端中不支持使用 UTF-16 编码。SA 导入模板并显示“模板”屏幕。
7. 请按照创建配置模板 (第 247 页)下从步骤 4 开始的步骤执行操作。

查看配置模板源

您可以查看配置模板的内容并查看其 CML 或 XML 源。这可以帮助您在将应用程序配置推送到服务器之前了解模板中已设置哪些列表合并模式。有关应用程序配置序列合并模式的信息，请参见序列聚合 (第 392 页)。

要查看配置模板的源，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并导航到模板文件所在的操作系统。请注意，模板可应用于多个操作系统。
3. 选择配置模板，然后选择“操作”>“打开”。
4. 选择“内容”视图，可显示配置模板的 CML 或 XML 内容。

将模板添加到应用程序配置或从应用程序配置删除模板

应用程序配置可包含一个或多个模板。

要将模板添加到应用程序配置或从其中删除模板，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“配置”节点。打开操作系统组并导航到应用程序配置所在的操作系统。请注意，应用程序配置可应用于多个操作系统。
3. 选择应用程序配置，然后选择“操作”>“打开”。
4. 选择“配置值”视图。
5. 要将模板添加到应用程序配置，请选择“操作”>“添加”或选择 "+" 按钮。选择所需模板并选择“确定”。

   这些模板必须适用于拥有此配置的所有平台(即，适用于此配置的 OS 必须是适用于这些模板的 OS 的子集)。

   包含此模板的文件夹客户设置必须包括此应用程序配置对象的客户设置。否则，可用模板的列表中将不会包括此模板。有关文件夹设置的详细信息，请参见SA 管理中的“文件夹权限”。
6. 要删除模板，请选择模板，然后选择“操作”>“删除”或选择 "-" 按钮。
7. 选择“文件”>“保存”以保存更改。

指定应用程序配置中的模板顺序

应用程序配置可以包含一个或多个配置模板及相关脚本。您可以指定应用程序配置中的模板顺序。将以应用程序配置中模板的显示顺序将模板推送到托管服务器。例如，你可能需要将更改在其他更改之前应用到某个配置文件。

应用程序配置中脚本的执行顺序由脚本类型确定：数据操作、安装前、安装后或出错后类型。应用程序配置中脚本的顺序是不相关的。有关详细信息，请参见 应用程序配置脚本的类型 (第 235 页)。

要指定应用程序配置中的模板顺序，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找“应用程序配置”节点并打开此节点。打开“配置”节点。打开操作系统组并导航到应用程序配置所在的操作系统。请注意，应用程序配置可应用于多个操作系统。
3. 选择应用程序配置，然后选择“操作”>“打开”。
4. 选择“配置值”视图。此时将显示应用程序配置中的所有配置模板和脚本(如果有)。请注意，已按顺序对模板和脚本进行编号。

5. 要对模板或脚本重新排序，请选中该模板或脚本，然后选择“操作”>“上移”或“操作”>“下移”，或选择向上箭头和向下箭头图标。
6. 选择“文件”>“保存”以保存更改。

从脚本创建模板

要使应用程序配置对象中包含脚本，必须将脚本复制到 CML 模板，然后将模板导入应用程序配置对象中。有关 CML 的详细信息，请参见CML 引用 (第 352 页)。请参见使用应用程序配置运行脚本 (第 234 页)。

以下示例显示了一个可执行 touch 命令和 echo 命令的 Unix Shell 脚本：

```
#/bin/sh
touch abc.txt
echo abc &gt;&gt;abc.txt
```

以下示例显示了转换成 CML 的此 Unix Shell 脚本：

```
@#################################################################
# /tmp/simple-script/TouchABC.sh                                #
# Version 0.1                                                   #
# Author:&lt;name&gt;                                                 #
#################################################################@
@!namespace=/simple-script-namespace/@
@!filename-key="/TouchABC"@
@!filename-default=/tmp/simple-script/TouchABC.sh@
#/bin/sh
touch abc.txt
echo abc &gt;&gt;abc.txt
```

要从此脚本创建模板，请执行以下操作：

1. 如创建配置模板 (第 247 页)中所述创建模板。将脚本的 CML 版本用作此模板的内容。
2. 将“类型”字段设置为相应的脚本类型。对于以上示例，您应将“类型”设置为“Unix .SH 脚本”。应用程序配置脚本的类型 (第 235 页)中列出了其他支持的脚本类型。
3. 由于必须使用 CML 语法编写所有脚本，因此请将“分析器语法”字段设置为“CML 语法”。
4. 如创建配置模板 (第 247 页)中所述设置剩余的字段。
5. 选择“文件”>“保存”。
6. 选择“文件”>“关闭”。

7. 如 将模板添加到应用程序配置或从应用程序配置删除模板 (第 264 页)中所述将模板添加到应用程序配置对象。
8. 在将模板添加到应用程序配置对象之后，打开此应用程序配置对象。
9. 选择“配置值”视图。
10. 选择此脚本模板并右键单击以显示菜单。
11. 选择脚本类型，此类型将指定运行脚本的时间：数据操作、安装前、安装后或出错后。
12. 选择“文件”>“保存”。
13. 选择“文件”>“关闭”。

通过运行数据操作脚本管理非文本配置

通过 SA，您可以管理非文本配置，方法是创建数据操作脚本，该脚本可提取非文本配置数据并将其放置于文本文件中。然后，SA 会对此文本文件进行操作(操作方式与任何其他文本配置文件相同)并将其放回原始的格式中。有关脚本类型的描述，请参见应用程序配置脚本的类型 (第 235 页)。

以下为非文本配置数据的某些示例：

- SQL 数据库，数据操作脚本可对其运行某些 SQL 查询并将数据置于文本文件中。
- IIS 服务器的配置，数据操作脚本可将其元数据库信息读取到文本文件中。
- 二进制文件，数据操作脚本可从此二进制文件提取值并将值置于文本文件中。

手动运行数据操作脚本

通过“运行脚本”按钮，您可以执行与某个应用程序配置关联的数据操作脚本并在托管服务器上准备目标配置文件，以便将其值导入值集中。

要手动运行数据操作脚本，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 选择“所有托管服务器”或“设备组”。导航到所需服务器或设备组。
3. 选择服务器或设备组，然后选择“操作”>“打开”菜单。

4. 如果选择服务器，则执行以下步骤。如果选择设备组，则跳到下一个步骤。
   a. 选择“管理策略”选项卡。
   b. 在导航窗格中打开“配置的应用程序”节点。此时将显示附加到服务器的所有应用程序配置。
   c. 打开要推送的“应用程序配置”节点。此时将显示此应用程序配置的值集。
   d. 选择要推送的应用程序配置实例。
5. 如果选择设备组，则执行以下步骤。
   a. 在导航窗格中打开“配置的应用程序”节点。此时将显示附加到服务器的所有应用程序配置以及所有服务器的列表。
   b. 打开“服务器”节点。此时将显示设备组中的所有服务器。
   c. 打开要运行数据操作脚本的特定服务器节点。此时将显示附加到该服务器的应用程序配置。
   d. 打开要推送的“应用程序配置”节点。此时将显示此应用程序配置的值集。
   e. 选择要推送的值集。必须在服务器实例级别选择值集。
6. 选择服务器实例级别的值集之后，选择“运行脚本”按钮。此时将显示确认对话框。
7. 选择“是”以在服务器上运行数据操作脚本。
8. 在数据操作脚本运行、提取配置数据并将数据置于一个文本文件之后，您可以使用与任何其他配置文件相同的管理方式管理此文本文件，然后将此文本文件中的数据放回其原始的格式中。

将应用程序配置附加到服务器或设备组

在创建应用程序配置、添加所有必要的配置模板和脚本以及编辑其默认值之后，必须将此配置附加到服务器或公用设备组。在将应用程序配置附加到服务器或服务器组后，必须如推送应用程序配置 (第 272 页)中所述将此应用程序配置推送到服务器。

- 包含应用程序配置的文件夹的客户设置必须包括您要向其推送此应用程序配置的托管服务器的客户设置。有关文件夹设置的详细信息，请参见SA 管理中的“文件夹权限”。有关服务器和客户的详细信息，请参见使用中的“客户帐户”。
- 您只能将应用程序配置附加到单台服务器或公用设备组。不能将其附加到专用设备组。

将应用程序配置附加到单台服务器

快捷方式：“设备”选项卡 > 选择“服务器”>“所有托管服务器”> 右键单击 >“附加”>“应用程序配置”> 选择一个配置文件 > 单击“确定”。

要将应用程序配置**附加到单台服务器，请执行以下操作：**

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 选择“服务器”>“所有托管服务器服务器”。
3. 在内容窗格中选择一台服务器。
4. 选择“操作”>“打开”菜单。
5. 选择“管理策略”选项卡，然后选择“配置的应用程序”。
6. 选择“安装的配置”选项卡。
7. 选择“操作”>“添加配置...”菜单。
8. 在“选择应用程序配置”屏幕中，选择要附加到托管服务器的应用程序配置。
   可使用搜索工具按特定条件搜索，例如名称、上次修改日期等。

   > **备注：**
   > 包含应用程序配置的文件夹的客户设置必须包括您要向其推送此应用程序配置的托管服务器的客户设置。有关文件夹设置的详细信息，请参见《SA 10.50 管理指南》中的“文件夹权限”。有关服务器和客户的详细信息，请参见《SA 10.50 用户指南》中的“客户帐户”。

9. 选择“确定”以将应用程序配置推送到服务器。
10. 选择“保存更改”按钮。
11. 现在可以为该服务器设置应用程序配置值。有关设置应用程序配置值的详细信息，请参见值集 (第 221 页)。

将应用程序配置附加到设备组

要将应用程序配置**附加到设备组，请执行以下操作：**

您只能将应用程序配置附加到公用设备组。不能将应用程序配置附加到专用设备组。

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 打开“设备组”节点并导航到公用设备组。
3. 在内容窗格中选择所需的设备组。
4. 选择“操作”>“打开”菜单。
5. 在设备组屏幕中，选择“配置的应用程序”视图。
6. 选择“操作”>“添加配置”菜单。
7. 在“选择应用程序配置”对话框中，选择应用程序配置。
   使用搜索工具按特定条件搜索，例如名称、上次修改日期等。
8. 输入“实例名称”。这是组实例级别的值集名称。有关详细信息，请参见 在组实例级别设置值 (第 232 页)。
9. 选择“确定”以将应用程序配置附加到设备组。指定的配置文件可被推送到此组中的所有服务器。
10. 设置要推送到服务器的值。有关设置值的详细信息，请参见值集 (第 221 页)。

从服务器或设备组分离应用程序配置

要从服务器中分离应用程序配置，必须打开此服务器并删除此应用程序配置的服务器实例级别的所有实例。要从设备组中分离应用程序配置，必须打开此设备组并删除组实例级别的所有实例。以下部分将提供详细信息。

从服务器分离应用程序配置

要从服务器中分离应用程序配置，必须删除服务器实例级别的所有实例，如下所述。有关详细信息，请参见 在服务器级别设置值 (第 233 页)。

要从服务器中分离应用程序配置，请执行以下操作：

1. 在 SA 客户端中打开服务器。必须将应用程序配置附加到服务器。请参见 将应用程序配置附加到服务器或设备组 (第 268 页)。
2. 选择左侧的“管理策略”选项卡。
3. 在左侧的“管理策略”窗格中打开“配置的应用程序”节点。

4. 在“配置的应用程序”节点下打开所需的应用程序配置节点。此时将显示此应用程序配置在服务器实例级别的所有实例。
5. 在此“应用程序配置”节点下，选择其中一个实例。
6. 选择“操作”>“删除配置”菜单或右键单击然后选择“删除配置”菜单。此操作将删除所选的实例。
7. 请针对每个实例重复执行在此“应用程序配置”节点下，选择其中一个实例。 (第 271 页)和选择“操作”>“删除配置”菜单或右键单击然后选择“删除配置”菜单。此操作将删除所选的实例。 (第 271 页)。删除最后一个实例后，将从此服务器中分离此应用程序配置。
8. 单击“保存更改”。

从设备组中分离应用程序配置

要从设备组中分离应用程序配置，必须删除此应用程序配置在组实例级别的所有实例，如下所述。有关详细信息，请参见 在组级别设置值 (第 231 页)。

要从设备组中分离应用程序配置，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 打开“设备组”节点并导航到公用设备组。
3. 在内容窗格中选择所需的设备组。
4. 选择“操作”>“打开”菜单。
5. 在设备组屏幕中，选择“配置的应用程序”视图并打开“配置的应用程序”节点。此时将显示附加到此设备组的应用程序配置。
6. 在“配置的应用程序”节点下打开所需的应用程序配置节点。此时将显示组实例级别的值集。
7. 在所需“应用程序配置”节点下，选择组实例级别的其中一个应用程序配置实例。
8. 选择“操作”>“删除配置”菜单或右键单击然后选择“删除配置”菜单。此操作将删除所选的实例。
9. 针对服务器实例级别的每个实例重复执行以上步骤 7 和步骤 8。删除最后一个实例后，将从此服务器中分离此应用程序配置。
10. 单击“保存更改”。

推送应用程序配置

无论何时更改值集中的值并要将这些更改与目标服务器上的配置文件合并，都必须将应用程序配置推送到服务器。有关详细信息，请参见将应用程序配置推送到服务器 (第 236 页)。

要将应用程序配置更改推送到服务器或服务器组，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 选择“所有托管服务器”或“设备组”。导航到所需服务器或设备组。
3. 选择服务器或设备组，然后选择“操作”>“打开”菜单。
   - 如果已选择服务器，则请选择“管理策略”选项卡。
   - 如果选择设备组，则跳到下一个步骤。
4. 在导航窗格中打开“配置的应用程序”节点并选择要推送的应用程序配置实例。

   通过选择“预览”按钮，可以选择预览将在单台服务器上进行的更改。比较屏幕将显示任何差异。在完成之后，请选择“关闭”。
5. 在准备好将更改应用到服务器之后，选择“推送”。
6. 在“推送配置”屏幕中，验证应用程序配置和要推送的值集。
   要接受“计划”、“通知”和“作业状态”的其他默认值，请单击“启动作业”。否则，请单击“下一步”继续查看向导选项。
7. 在“计划”窗格中，指定要推送应用程序配置的时间。可以使用“计划”窗格计划在未来运行作业，或定期运行作业，例如每周或每月。
   要接受“通知”和“作业状态”的其他默认值，请单击“启动作业”。否则，请单击“下一步”继续查看向导选项。
8. 在“通知”窗格中，可以选择指定一个或多个电子邮件地址和一个工单标识符。对于每个收件人，请选择发送电子邮件通知的时间。
   - 成功时：在作业成功时向收件人发送电子邮件。
   - 失败时：在作业失败时向收件人发送电子邮件。
   - 终止时：在作业终止时向收件人发送电子邮件。
     - 当通过“结束作业”操作停止正在运行的作业时，作业将终止。
     - 此通知将不会 应用于在运行之前取消的作业。

要接受“作业状态”的其他默认值，请单击“启动作业”。否则，请单击“下一步”继续查看向导选项。

9. 单击“启动作业”。

   在启动作业之后，可通过在主 SA 客户端屏幕上依次选择“作业和会话”选项卡和“作业日志”，查看作业的状态。

   您还可以执行以下任意可选操作：
   - 单击“导出”将作业状态结果导出到文本文件。
   - 单击“结束作业”以停止作业。请参见 停止推送配置作业 (第 273 页)。
   - 单击“关闭”以关闭窗口。要在之后查看作业状态，请在 SA 客户端导航窗格中单击“作业状态”，然后双击要查看其详细信息的作业。

停止推送配置作业

您可以终止正在运行的推送配置作业。例如，您可能需要停止正在生成错误结果的作业或将要超出已分配维护窗口运行的作业。

要保持作业的完整性，不能取消推送配置作业流的某些步骤。在停止作业时，“作业状态”窗口将指示哪些步骤已完成，哪些步骤已跳过。

要停止活动的应用程序配置推送作业，请执行以下操作：

1. 在“作业状态”窗口中，单击“结束作业”。(仅当作业正在进行时，才会显示此按钮。)
2. 此时将显示“结束作业”警告对话框，告知您作业终止的作用方式：
   - 此作业将不会启动任何其他服务器上的操作
   - 如果服务器上已启动工作，则作业将仅跳过可安全取消的步骤
   - “作业状态”将指示已完成或跳过的步骤
   - 如果成功终止作业，则最终的作业状态将为“已终止”
3. 单击“确定”，确认要终止此作业。此时“作业状态”窗口将显示终止过程的进度。

   作业状态将为“已终止”。服务器状态将为“已取消”。任务状态将为“成功”或“已跳过”。
4. 当终止完成后，您还可以在 SA 客户端的“作业日志”中查看作业。

   在 SA 客户端导航窗格中，单击“作业和会话”。此时将显示“作业日志”视图，其中列出状态为“已终止”的作业。

修改推送超时值

默认情况下，推送应用程序配置时，默认超时值为十分钟加上应用程序配置中每个模板一分钟。该应用程序配置中的每个模板将其超时附加到此应用程序配置的基本超时。

例如，如果应用程序配置包含三个模板，则整个应用程序配置的默认超时值为 13 分钟。如果已推送模板并且整个推送时间超过 13 分钟，则推送将超时并且将取消此操作，包含所做的任何更改。

要延长模板的超时值，可以针对应用程序配置中的单个模板使用 CML 超时标记。CML 超时标记语法如下所示：

```
@!timeout=1@
```

有效值介于 0 到 999 之间，以分钟为单位。

如果在推送过程中应用程序配置发生超时，则将退出对推送目标文件的所有更改并取消此操作。

有关 CML 超时标记的详细信息，请参见CML 引用 (第 352 页)。

计划应用程序配置推送

您可以计划使应用程序配置推送立即运行、将来运行或是定期运行，例如每天、每周或每月。要计划应用程序配置推送，请按照推送应用程序配置 (第 272 页)列出的步骤执行操作，但是在执行到“计划”步骤时，请输入希望发生推送的频率和时间。

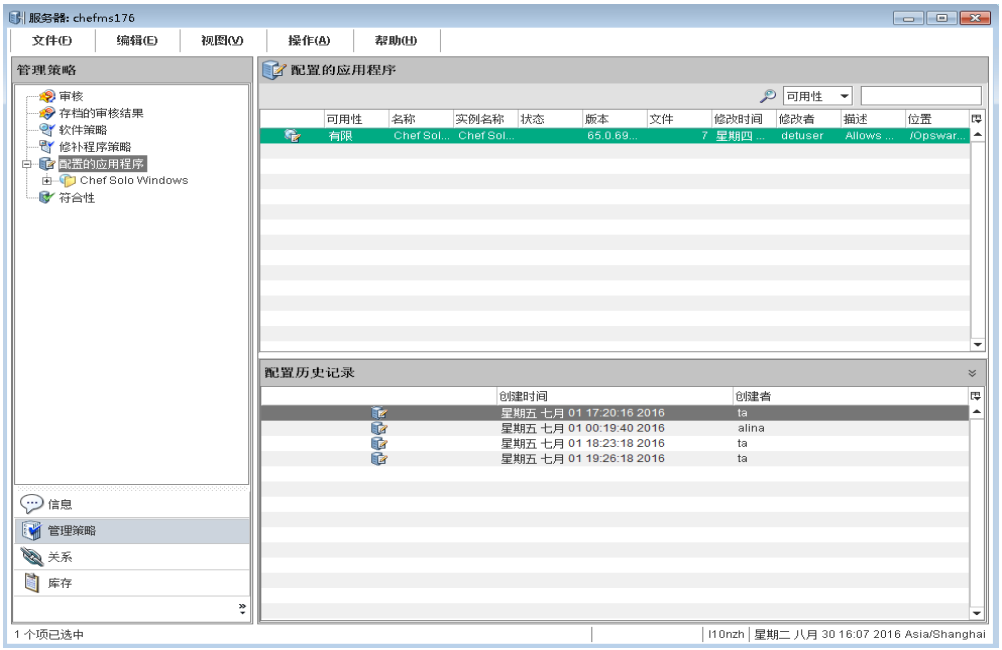
在计划作业之后，可通过在主 SA 客户端屏幕上依次选择“作业和会话”选项卡和“定期计划”，查看作业的状态。

将配置文件恢复到之前的状态

每次将应用程序配置推送到服务器时，都将在配置推送历史记录列表中保存配置文件。在任何时候，您都可以将应用程序配置恢复到历史记录列表中的之前状态。

要将应用程序配置**恢复到之前状态，请执行以下操作：**

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 选择“所有托管服务器”并查找所需的服务器。
3. 选择服务器，然后选择“操作”>“打开”菜单。
4. 选择“管理策略”选项卡。
5. 在导航窗格中选择“配置的应用程序”节点。此时将显示在此服务器上安装的所有应用程序配置。
6. 选择一个配置以在详细信息窗格中查看其配置历史记录。这将显示服务器上运行的所有推送作业。

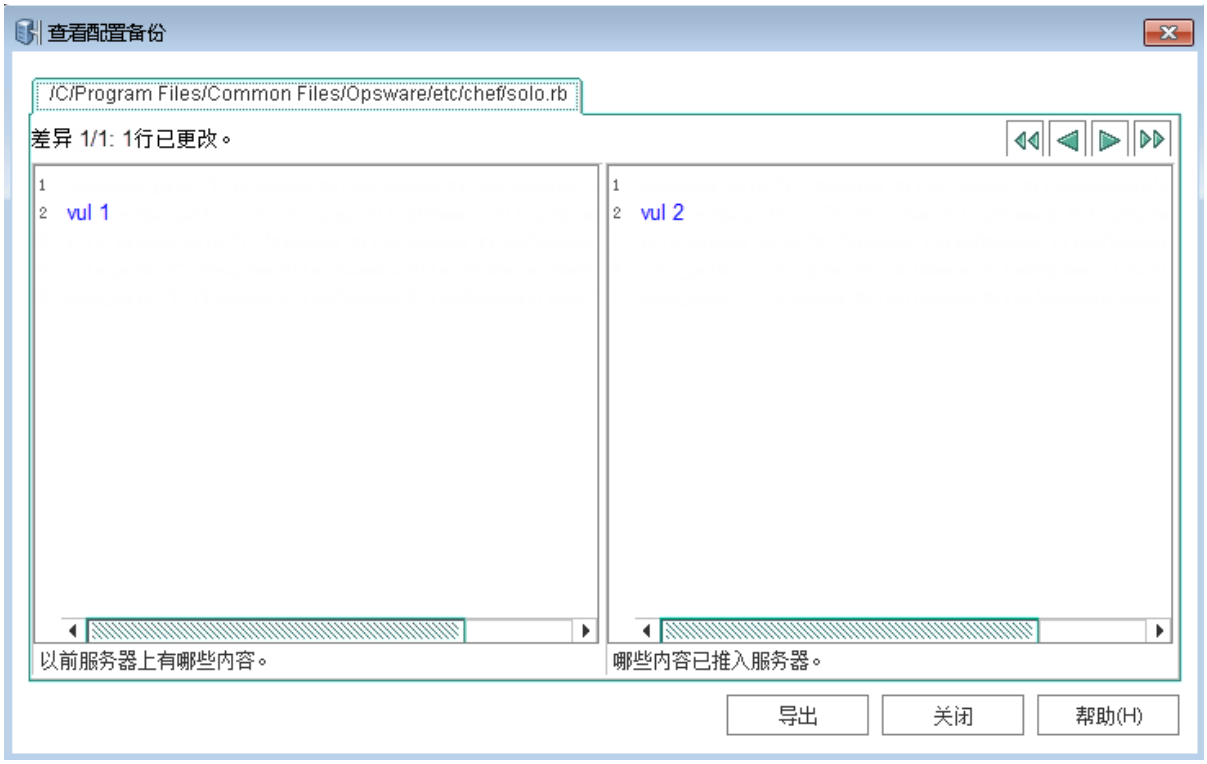


7. 右键单击其中一个历史实例并选择“查看”查看快照。

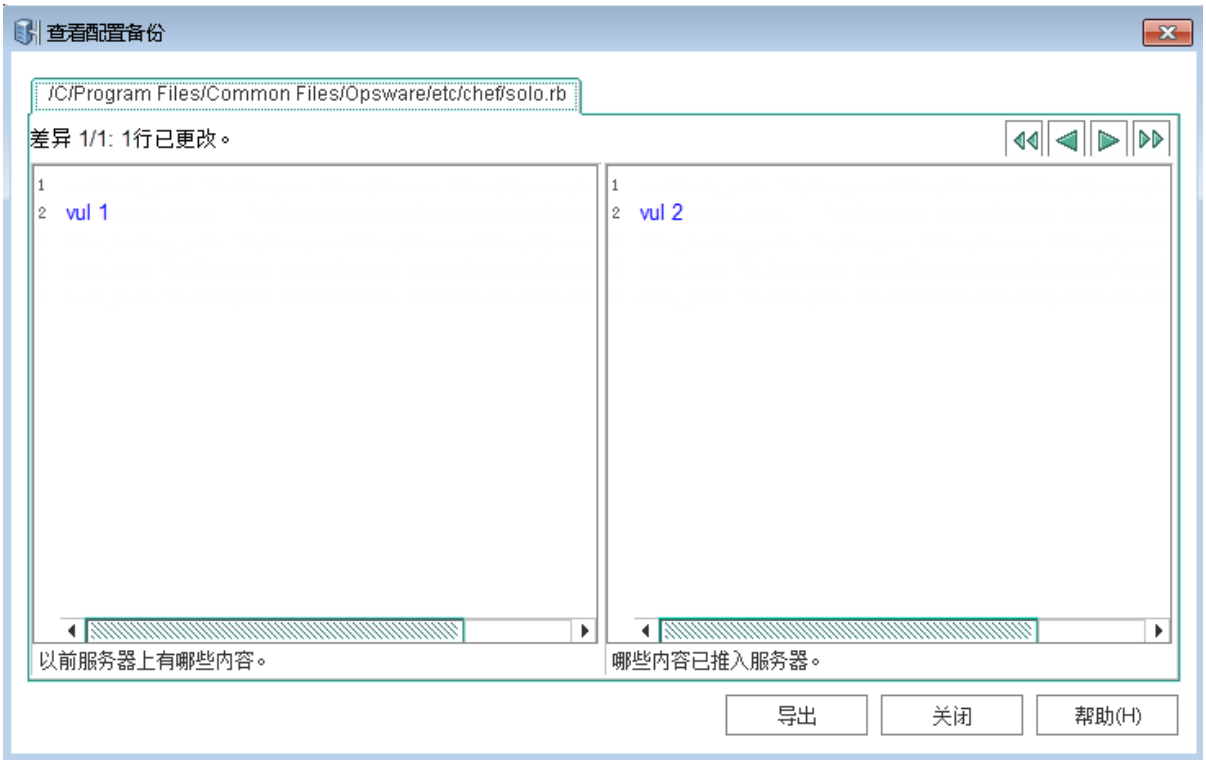
“查看配置备份”窗口将显示为拆分窗格以供比较。

- 左侧窗格显示在快照之前所推送文件的内容，它本质上是现有文件(如果有的话)的备份。

- 右侧窗格显示选定快照中已推送文件的内容。



8. 选定历史记录中要恢复的行，然后单击“恢复”。
   此时将显示“恢复配置”向导。
9. 验证服务器和要恢复的历史实例，然后单击“下一步”。
10. 选择恢复类型：
    - 上一个选定的推送(撤消)可将配置文件恢复到紧邻选定历史记录实例之前的值。即，将撤消选定的实例。
    - 下一个选定的推送(重做)可将配置文件恢复为选定历史实例的值。即，重新推送所选的实例。
11. 单击“下一步”。
12. 在“预览”步骤中，单击“预览”可比较对服务器上当前实例的更改。
    “查看配置备份”窗口将显示为拆分窗格以供比较。
    - 左侧显示在给定位置服务器上的当前内容。
    - 基于作业中选定的“恢复类型”，右侧包含将要恢复到该位置的内容。



该预览提供了在执行恢复之前查看结果的选项：

- 要取消，请关闭窗口并从作业窗口中选择“取消”。
- 要继续执行，请关闭窗口然后继续执行下一步。

13. 单击 **Start Job**。此操作会将服务器上的配置恢复到选定的历史记录实例。
14. 在启动作业之后，可通过在主 SA 客户端屏幕上依次选择“作业和会话”选项卡和“作业日志”，查看作业的状态。

搜索和筛选作业结果

您可以搜索和筛选推送或恢复作业的结果。在大量服务器上运行作业时，此操作将十分有用。请注意，您可以搜索和筛选任何类型的 SA 作业，而不仅仅是推送和恢复作业。

要搜索和筛选结果，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“作业和会话”选项卡。
2. 选择“作业日志”。此时将显示此服务器上已运行的作业列表。
3. 从此作业列表中选择一个作业。
4. 选择“操作”>“打开”菜单。此时将显示选定作业的详细信息。

5. 在作业屏幕中，选择“作业状态”。此时将显示此作业的结果。
6. 在“操作”列中选择任何步骤。
7. 在键盘上键入 Ctrl-F。此时将显示“查找”工具，用于查找此作业中的步骤。在文本框中输入要搜索的文本，然后使用按钮进行搜索并突出显示。
8. 在下方框中选择详细的文本。
9. 在键盘上键入 Ctrl-F。此时将显示“查找”工具，用于查找此作业结果的详细信息。在文本框中输入要搜索的文本，然后使用按钮进行搜索并突出显示。
10. 要删除“查找”工具，请选择此文本框并键入键盘上的 Esc。

将配置模板与目标配置文件进行比较 - 预览

在将应用程序配置推送到服务器之前，可以使用“预览”按钮将建议的应用程序配置与服务器上的目标配置文件进行比较。您可以从托管服务器中选择服务器，也可以在设备组中选择服务器。

从托管服务器中选择服务器

要将配置模板中的值与服务器上的实际配置文件进行比较，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 选择“服务器”>“所有托管服务器服务器”。
3. 在内容窗格中选择一台服务器。
4. 选择“操作”>“打开”菜单。
5. 选择“管理策略”选项卡。
6. 打开“配置的应用程序”节点，将显示附加到服务器的所有应用程序配置。
7. 打开所需的“应用程序配置”节点，将显示该应用程序配置的所有实例。
8. 选择此应用程序配置的一个实例。
9. 在内容窗格中，从下拉列表选择一个模板。

10. 单击“预览”。此时会将从模板生成的配置文件和值集与服务器上的实际配置文件进行比较，然后并列显示两个文件，其中使用颜色编码以便于解释。
    - **绿色**：表示新信息已添加。
    - **蓝色**：表示信息已修改。
    - **红色**：表示信息已删除。
    - **黑色**：表示无更改。

从设备组中选择服务器

要将配置模板中的值与设备组中服务器上的实际配置文件进行比较，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“设备”选项卡。
2. 选择“设备组”>“公用”，将显示公用设备组。
3. 导航到所需设备组，然后在内容窗格中选择一个公用设备组。
4. 选择“操作”>“打开”菜单。
5. 打开“配置的应用程序”节点，将显示附加到此设备组的所有应用程序配置和此设备组中的服务器。
6. 打开“服务器”节点将显示设备组中的所有服务器。
7. 打开所需的服务器节点，将显示附加到该服务器的应用程序配置。
8. 打开所需的“应用程序配置”节点，将显示此应用程序配置的所有实例。
9. 选择此应用程序配置的一个实例。
10. 在内容窗格中，从下拉列表选择一个模板。
11. 单击“预览”。此时会将从模板生成的配置文件和值集与服务器上的实际配置文件进行比较，然后并列显示两个文件，其中使用颜色编码以便于解释。
    - **绿色**：表示新信息已添加。
    - **蓝色**：表示信息已修改。
    - **红色**：表示信息已删除。
    - **黑色**：表示无更改。

比较配置模板

要比较两个配置模板，**请执行以下操作：**

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并导航到模板文件所在的操作系统。请注意，模板可应用于多个操作系统。
3. 选择一个配置模板。
4. 按住键盘上的 Ctrl 键并选择第二个模板。
5. 右键单击并选择“比较”菜单。“比较”屏幕将显示两个文件之间的差异。使用屏幕右上方的箭头可在两个文件之间导航。以下颜色表示差异：
   - **蓝色**表示两个模板之间的不同信息。
   - **红色**表示已删除的信息。
   - **绿色**表示已添加的信息。
   - **黑色**表示相同的文本。
6. 当结束查看差异时，请选择“关闭”。

对配置文件元素名称进行本地化

您可以在 SA 客户端的值集编辑器中以您的本地语言显示配置文件元素的名称。如果负责指定值集的系统管理员使用的语言与您不同，此功能将很有帮助。

本地化文件表示特定于区域设置的资源文件，您可以在其中为配置模板元素定义本地化字符串。值集编辑器中将显示这些本地化字符串。

本地化模板仅在 SA 客户端的值集编辑器中使用，在推送期间将忽略这些模板。

创建本地化文件

要对 SA 客户端中的配置文件元素名称进行本地化，必须首先在 SA 库中创建本地化模板。

要创建本地化文件，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并选择将使用模板文件的操作系统。请注意，模板可应用于多个操作系统。可在后面的步骤中进行指定。
3. 选择“操作”>“新建”菜单。此时将显示“模板”屏幕，您可以在其中定义模板的属性。

4. 选择“属性”视图并输入本地化模板文件的描述及以下信息：
   - **名称**：指定本地化文件的名称。本地化模板文件的命名约定要求本地化文件以 .<区域设置>结尾。<区域设置>的值由 ISO-639 定义，此值是两个小写字母代码，用于表示语言名称。有关 ISO 639 的详细信息，请参见 http://www.loc.gov/standards/iso639-2/php/code_list.php
   - 例如，.es 代表西班牙语、.en 代表英语、.fr 代表法语、.zh 代表中文以及 .hi 代表印地语。
   - **位置**：指定 **SA** 库中用于存储本地化模板的位置。
   - **版本**：版本可以是用于跟踪模板更改的任何字符串。版本不会自动增加。
   - **类型**：指定本地化文件的类型。
   - **可用性**：使用此设置可跟踪已经过测试并准备好使用的本地化文件以及尚未测试或已弃用的本地化文件。此设置不会更改本地化文件可以实现的功能。您可以将此字段值用作搜索条件。
5. 选择“内容”视图。
6. 在模板编辑器中直接输入本地化说明。
   - 本地化说明的显示文本格式如下所示：

     `printable/<命名空间>/<变量> <本地化字符串>`

     其中 `printable` 是关键字，表示此行是本地化说明的显示文本。

     <命名空间>是数据库中用于存储所需变量的命名空间。

     <变量>是配置模板中定义的变量。

     <本地化字符串>是将在值集编辑器中显示的文本，而不是完整命名空间和变量名称。
   - 本地化说明的工具提示格式如下所示：

     `description/<命名空间>/<变量> <本地化字符串>`

     其中 `description` 是关键字，表示此行是本地化说明的工具提示文本。

     <命名空间>和 <变量>与上述相同。

     <本地化字符串>是在值集编辑器中，当某人将鼠标悬停在项上时显示的文本。

   有关编辑操作和语法突出显示的信息，请参见编辑模板中的 CML 或 XML (第 262 页)。
7. 选择“验证”可分析语法并检查错误。
8. 选择“文件”>“保存”以保存模板。
9. 如应用本地化模板 (第 282 页)所述将本地化模板添加到应用程序配置对象。

应用本地化模板

在创建本地化模板之后，您可以将其应用到应用程序配置，以便此配置文件元素显示在您的本地语言中。

要将本地化模板应用到应用程序配置，请执行以下操作：

1. 打开应用程序配置对象，如下所示。
   a. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
   b. 查找并打开“应用程序配置”节点。打开“配置”节点。打开操作系统组并导航到应用程序配置所在的操作系统。请注意，应用程序配置可应用于多个操作系统。
   c. 选择应用程序配置，然后选择“操作”>“打开”。
2. 选择“属性”视图。
3. 在内容窗格的“本地化文件”下，选择 "+" 按钮。
4. 在“选择本地化模板”屏幕中，选择一个本地化模板。
5. 选择“确定”。
6. 选择“文件”>>“保存”。此操作将应用本地化模板并使用本地化模板指定的语言显示配置文件元素。在任何人显示此应用程序配置的值集编辑器时，都将显示本地化字符串，而不会显示原始的命名空间和变量名称。

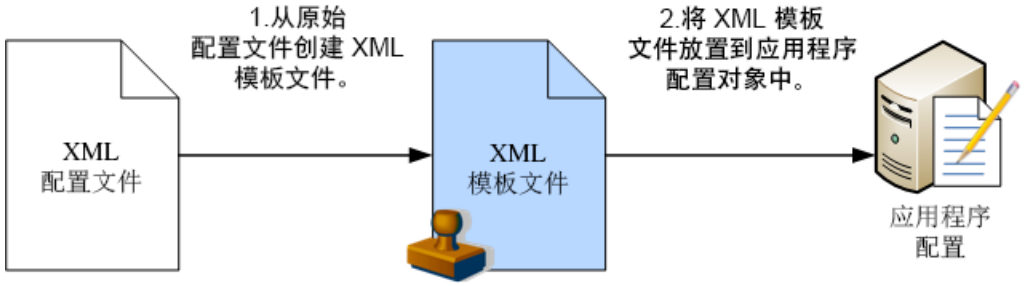
管理 XML 配置文件

使用 SA，可以从一个中心位置管理 XML 配置文件，并在数据中心的多台服务器中传播更改。可创建、编辑和存储配置文件值，确保托管服务器上的 XML 配置文件正确。可以对使用 DTD 的 XML 文件进行管理，也可以对不使用 DTD 的 XML 文件进行管理。

本节将讨论如何构造 XML 配置模板，以便管理常规 (非 DTD) XML 文件以及引用 DTD 的 XML 文件。由于 XML 结构良好，SA 只需要最少的信息就可对基于 XML 的配置文件进行建模和管理。

要管理 XML 配置文件，首先需要为 XML 配置文件创建**模板**文件。在创建模板之后，必须将模板添加到**应用程序配置**对象，以便对托管服务器上的本地配置文件进行管理、编辑和更改。

配置文件



以下部分将描述简单 XML 文件并显示如何分别为基于非 DTD 的 XML 文件以及基于 DTD 的 XML 文件创建应用程序配置。

另请参见以下示例:

- XML 教程 1 - 创建非 DTD 的 XML 配置模板 (第 292 页)。
- XML 教程 2 - 创建 XML-DTD 配置模板 (第 298 页)。

示例：Travel Manager 应用程序和 XML 配置文件

本部分将描述一个 Web 应用程序示例，该应用程序使用简单 XML 文件控制其配置，以及显示如何创建应用程序配置来管理该文件。

Travel Manager 是 Web 应用程序，人们可通过此应用程序执行各种任务来管理旅行，这些任务包括预订酒店、租车、跟踪费用等。Travel Manager 使用 MySQL 关系数据库管理系统 (RDMS)，将其作为存储用户数据和某些应用程序配置数据的库。

由于 Travel Manager 可在多个不同网络上部署，并且每个网络具有不同的数据库服务器，因此灵活提供用于连接 MySQL 服务器的信息十分重要。此应用程序用于从 XML 配置文件 `mysql.xml` 检索连接信息。

借助应用程序配置，可以设置访问本地 MySQL 数据库所需的配置文件值。例如，用于打开数据库连接的用户名和密码针对每个 Travel Manager 应用程序安装可能不同。对配置文件中这些值的修改不需要重新编译 Travel Manager 应用程序代码。

`mysql.xml` 文件中只有四个值是 Travel Manager 用于连接本地 MySQL 数据库所必需的值，每个值以此应用程序的 XML 文件中的元素表示：

- **主机**：已安装 MySQL RDMS 的服务器的主机名。
- **名称**：主机服务器上的数据库名称。

- **用户**：用于打开数据库连接的用户名凭据。
- **密码**：打开数据库连接所需的密码。

Travel Manager mysql.xml 文件内容

以下是 Travel Manager `mysql.xml` 配置文件的一个示例：

```
<?xml version="1.0" ?>
<db-config>
  <db-host>localhost</db-host>
  <db-name>wrightevents</db-name>
  <db-user>root</db-user>
  <db-password>hp-pass</db-password>
</db-config>
```

Travel Manager mysql.xml 基于 DTD 的 XML 文件内容

以下是引用 DTD 的 Travel Manager `mysql.xml` 配置文件的一个示例：

```
<?xml version="1.0"?>
<!DOCTYPE db-config PUBLIC "-//Williams Events//Travel Manager//EN" "mysql2.dtd">
<db-config>
       <db-host>localhost</db-host>
       <db-name>wrightevents</db-name>
       <db-user>root</db-user>
       <db-password>hp-pass</db-password>
</db-config>
```

非 DTD 的 XML 配置模板

可创建基于非 DTD 的 XML 配置模板，将其编写为包含三种所需信息的单个 XML 注释，通过此注释，模板可以从目标 XML 文件提取值并存储它们：

- ACM-NAMESPACE:定义数据库中用于存储从托管服务器的目标 XML 文件中读取的值的位置。此命名空间必须唯一且路径必须以正斜杠 (/) 开头。
- ACM-FILENAME-DEFAULT:定义托管服务器上目标 XML 配置文件的默认绝对路径。
- ACM-FILENAME-KEY:定义命名空间中用于存储目标 XML 配置文件名的位置。

在将配置模板的属性设置为使用 XML 语法时，显示在值集编辑器中的标签与 XML 文件中每个相应元素的标记名称相同。

有关 XML 模板的模板设置的完整列表，请参见XML 配置模板设置 (第 291 页)。

有关将配置模板的分析器语法设置为 XML 的信息，请参见创建配置模板 (第 247 页)。

mysql.xml 的 非 DTD XML 配置模板

以下示例显示了基于 mysql.xml 文件的 XML 配置模板。将此模板文件命名为 mysql.tpl 以表示它是模板文件。

```
<!--
ACM-NAMESPACE = /TravelManager/
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
ACM-TIMEOUT = 1
-->
```

此示例显示 XML 配置模板引用目标 XML 文件 (/var/www/html/we/mysql.xml)，以便应用程序配置分析器可分析此模板，并且可在 SA 库中读取和存储此模板的值。

mysql.tpl 配置模板包含以下所需信息：

- ACM-NAMESPACE:定义数据库中用于存储从托管服务器的 mysql.xml 文件中读取的值的位置。此命名空间必须唯一且路径必须以正斜杠 (/) 开头。
- ACM-FILENAME-DEFAULT:定义托管服务器上 mysql.xml 文件的默认绝对路径。
- ACM-FILENAME-KEY:定义命名空间中将用于存储 mysql.xml 文件名的位置。
- ACM-TIMEOUT:(可选)代表在推送期间要添加到配置模板默认超时值(十分钟)的分钟数。

整个应用程序配置的默认超时值为十分钟，加上此应用程序配置中每个配置模板的超时值。因此，如果此模板是某个应用程序配置(超时值为十分钟)中的唯一模板，并且将此值设置为 1，则在推送时整个应用程序配置的整体超时值将为十一分钟。

基于 DTD 的 XML 配置模板

XML-DTD 配置模板实际上只是一个 XML DTD，其中包含注释中定义的某些应用程序配置选项。由于 DTD 标准定义 XML 文件的语法和布局，因此无需使用其他语言重新定义该语法。

对于基于 DTD 的 XML 文件，XML-DTD 配置模板要求具有与常规 XML 文件所需的相同的三个基本特性 — ACM-NAMESPACE、ACM-FILENAME-DEFAULT 和 ACM-FILENAME-KEY — 以及三个其他特性：

- ACM-DOCTYPE:定义 XML 文件中根元素的名称。此根元素遵循在目标 XML 配置文件中找到的开放 <!DOCTYPE 声明。
- ACM-DOCTYPE-SYSTEM-ID:定义托管服务器上关联 DTD 文件的名称。通常，此值可在 XML 配置文件中作为 DOCTYPE 元素中的 SYSTEM 特性找到。
- ACM-DOCTYPE-PUBLIC-ID:定义可表示 XML 文档公用标识符的字符串。通常，此值可在 XML 配置文件中作为 DOCTYPE 元素的 PUBLICID 特性找到。

有关所有 XML 配置文件特性的完整列表，请参见XML 配置模板设置 (第 291 页)。

mysql.xml 的 XML-DTD 配置模板

以下是为 Travel Manager 基于 DTD 的 XML 文件创建的配置模板示例。

```
<!--
    ACM-FILENAME-KEY = /files/TravelManager
    ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
    ACM-NAMESPACE = /TravelManager/
    ACM-TIMEOUT = 1
    ACM-DOCTYPE = db-config
    ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
    ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
    -->
    <!ELEMENT db-config (db-host,db-name,db-user,db-password)>
    <!ELEMENT db-host (#PCDATA)>
    <!ELEMENT db-name (#PCDATA)>
    <!ELEMENT db-user (#PCDATA)>
    <!ELEMENT db-password (#PCDATA)>
```

在本例中，DOCTYPE 特性引用特定的 XML 和 DTD 信息，这些信息可帮助分析器从 DTD 文件和引用的 XML 文件提取信息。

具体而言，基于 DTD 的 XML 配置模板必须包含以下信息：

- ACM-DOCTYPE:目标 XML 文件的根节点。对于 mysql.xml，根节点为 dbconfig。
- ACM-DOCTYPE-SYSTEM-ID:配置模板的目标 DTD 文件的名称。在 mysql.xml 示例中，使用的 DTD 名为 mysql.dtd。
- ACM-DOCTYPE-SYSTEM-ID:XML 文件的公用 ID。

自定义 XML DTD 元素显示方式

可以向 XML-DTD 配置模板添加两个可选设置，用于支持您自定义目标 XML-DTD 配置文件中的元素在 SA 客户端值集编辑器中的显示方式。通过 ACM-PRINTABLE 和 ACM-DESCRIPTION 可选设置，可以控制元素显示在 SA 客户端客户端中的名称：

- ACM-PRINTABLE:定义在 SA 客户端中显示 XML-DTD 模板时，将在值集编辑器中显示的 XML 文件中每个元素的标签。
- ACM-DESCRIPTION:定义当用户在 SA 客户端的值集编辑器中将鼠标指针悬停在 ACM-PRINTABLE 中定义的字段时显示的鼠标悬停文本。

显式和位置显示设置

可采用两种方式之一为 XML-DTD 配置模板中的特性和元素设置可打印值和描述值：位置方式或显式方式。

- 通过位置定义，直接将 ACM-PRINTABLE 和 ACM-DESCRIPTION 插入到它们在 XML-DTD 配置模板中描述的元素或特性之后。
- 通过显式定义，可在模板中的任意位置定义 ACM-PRINTABLE 和 ACM-DESCRIPTION。

```
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!--
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that contains
the information needed to connect to a database.
-->

<!ELEMENT db-host (#PCDATA)>
<!--
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer (the
server) on which the database engine is running.
-->
```

```
<!ELEMENT db-name (#PCDATA)>
<!--
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->

<!ELEMENT db-user (#PCDATA)>
<!--
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used to
connect to the database.
-->

<!ELEMENT db-password (#PCDATA)>
<!--
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to connect to
the database.
-->
```

添加位置自定义显示设置

将元素表和鼠标悬停文本添加到 XML 模板的位置方法是在要定义的元素或特性定义之后立即添加注释，并在该注释中设置 ACM-PRINTABLE 和 ACM-DESCRIPTION 值。换句话说，对于 XML 元素或特性，可为其指定标签并为此标签直接指定鼠标悬停描述。

在以下示例中，mysql.xml 中的每个 XML 元素可在 XML-DTD 模板中的每个元素之后立即定义 ACM-PRINTABLE 和 ACM-DESCRIPTION 设置。

添加显式自定义显示设置

通过将设置添加到 XML-DTD 模板的这种显式方法，可以在配置模板中的任何位置定义 ACM-PRINTABLE 和 ACM-DESCRIPTION 值，方法是使用 ACM-ELEMENT 标记指定元素名称以及使用 ACM-ATTRIBUTE 标记指定特性名称(可选)。

对于这种方法，ACM-ELEMENT 标记是必需的(即使是在定义特性的可打印值和描述值时)，因为特性始终与特定元素相关联。

在设置 ACM-ELEMENT 和 ACM-ATTRIBUTE 标记之后，还可以在同一个注释块中设置 ACM-DESCRIPTION 和 ACM-PRINTABLE 标记。针对每个注释块应仅使用一个定义。换句话说，为单个元素定义 ACM-PRINTABLE 和 ACM-DESCRIPTION，然后为下一个元素启动新的注释块。

应在 ACM-PRINTABLE 和 ACM-DESCRIPTION 标记之前定义 ACM-ELEMENT 标记和 ACM-ATTRIBUTE 标记(如果适用)。

例如，要自定义 mysql.tpl 模板，需按以下方式构造模板：

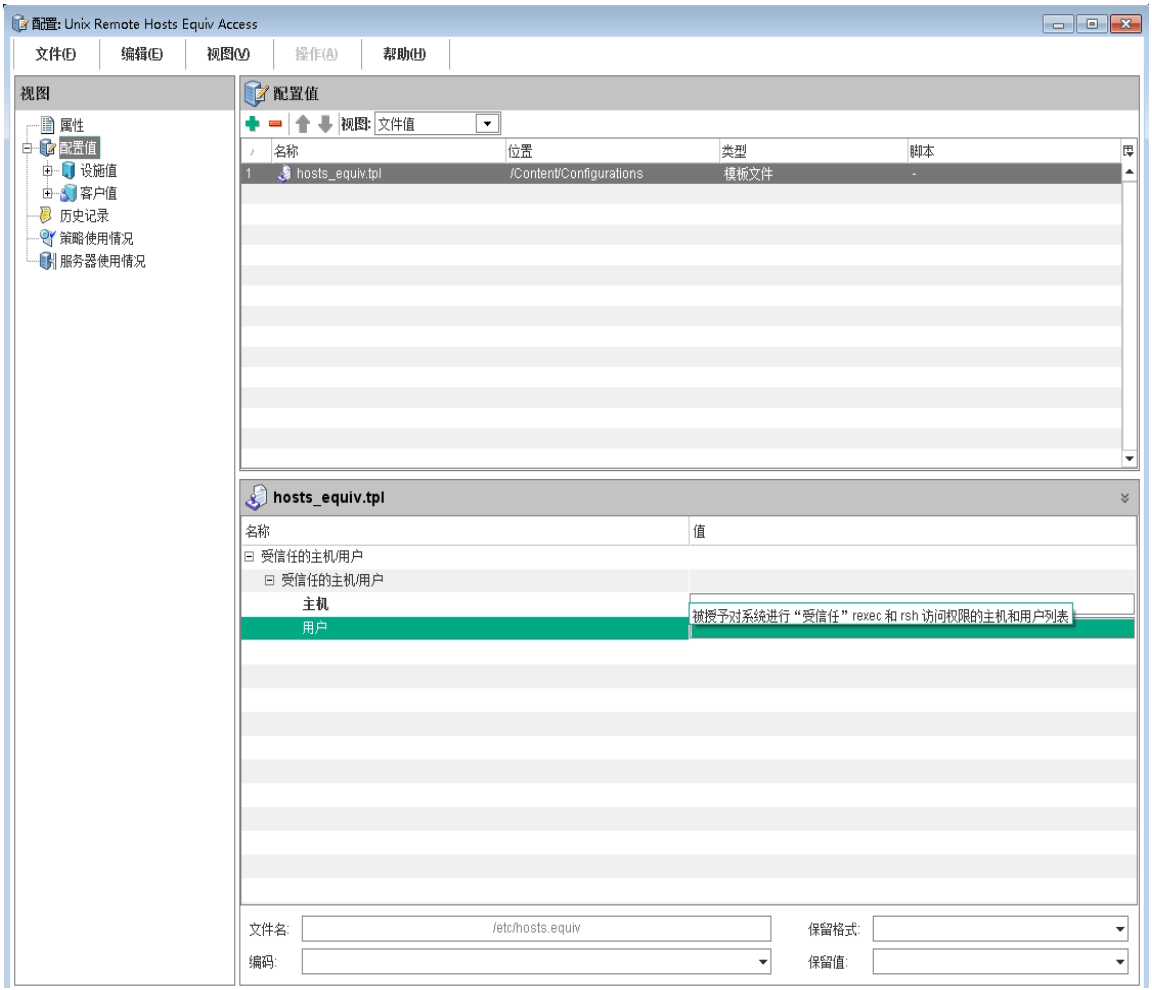
```
<!--
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql2.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->

<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host (#PCDATA)>
<!ELEMENT db-name (#PCDATA)>
<!ELEMENT db-user (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>

<!--
ACM-ELEMENT = db-config
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that contains
the information needed to connect to a database.
-->

<!--
ACM-ELEMENT = db-host
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer (the
server) on which the database engine is running.
-->

<!--
ACM-ELEMENT = db-name
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->

<!--
ACM-ELEMENT = db-user
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used to
connect to the database.
-->

<!--
ACM-ELEMENT = db-password
```

```
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to connect to
the database.
-->
```

自定义元素在 SA 客户端中的显示方式

在这两种情况中，无论是采用位置方式还是显式方式添加这些特性，最终的结果都是相同的：值集编辑器在 SA 客户端中显示元素名称 (在 ACM-PRINTABLE 中定义) 和鼠标悬浮文本 (在 ACM-DESCRIPTION 中定义)，如下图所示。

自定义元素名称和鼠标悬浮文本



XML 配置模板设置

下表描述了在创建常规或基于 DTD 的 XML 配置模板时可用的所有 XML 设置。此列表指示设置是必需还是可选设置，以及是否仅适用于 XML-DTD 模板。

XML 和 XML-DTD 模板设置

特性	描述
ACM-FILENAME-KEY=<key> 必需；无默认值。	filename-key 确定包含已生成文件的文件名的值集中键的路径。
ACM-FILENAME-DEFAULT=<filename> 必需；无默认值。	filename-default 确定值集中无文件名时返回的默认文件名。
ACM-NAMESPACE=<string> 必需；无默认值。	namespace 确定数据库中存储 XML 元素的位置。
ACM-TIMEOUT=<integer> 可选；(默认值为 0)	timeout 表示添加到应用程序配置总超时的分钟数。 有效超时值是介于 0(含)到 999(含)之间的任意整数。 将应用程序配置中所有配置模板的超时值相加并将所得数字加到配置的默认超时值十分钟，就会得到整个配置的最终超时值。 请注意，应用程序配置中运行时间超过十分钟的任何安装前脚本或安装后脚本都将超时，并且取消整个推送作业。
ACM-DOCTYPE = <string> 必需；无默认值。 仅适用于 XML-DTD 模板。	doctype 表示 XML 文件中根元素的名称。它位于 XML 文件开头的 DOCTYPE 标记中。
ACM-DOCTYPE-SYSTEM-ID = <string> 必需；无默认值。 仅适用于 XML-DTD 模板。	system-id 表示配置模板所基于的 DTD 文件的系统 ID。此值位于 XML 文件开头的 DOCTYPE 标记中。
ACM-DOCTYPE-PUBLIC-ID = <string> 必需；无默认值。 仅适用于 XML-DTD 模板。	public-id 表示使用配置模板分析的 XML 文件的公用 ID。此值位于 XML 文件 DTD 选项开头的 DOCTYPE 标记中。

XML 和 XML-DTD 模板设置(续)

特性	描述
ACM-ELEMENT=<element name> 可选 仅适用于 XML-DTD 模板。	element 设置当前选项描述的元素。此选项默认设置为 DTD 文件中此部分之前出现的任何元素或特性。
ACM-ATTRIBUTE=<attribute name> 可选 仅适用于 XML-DTD 模板。	attribute 设置当前选项描述的特性。如果未设置任何特性，则忽略此选项。此特性默认设置为文件中此部分之前出现的任何元素或特性。
ACM-PRINTABLE=<printable> 可选 仅适用于 XML-DTD 模板。	printable 为 SA 客户端中的元素或特性设置的可打印值。此值显示在此字段左侧的值集编辑器中。通常将此值设置为某些简短的描述性内容。
ACM-DESCRIPTION=<description> 可选 仅适用于 XML-DTD 模板。	description 设置要显示在 SA 客户端中的当前元素或特性的描述。将鼠标悬停在值集编辑器中的名称或值字段时，会显示此值。使用此值可描述值集编辑器中字段的用途以及字段的有效值。

XML 教程 1 - 创建非 DTD 的 XML 配置模板

本教程将显示如何为非 DTD 的 XML 配置文件创建配置模板。它将向您展示如何使用 XML 语法创建配置模板、将此模板添加到应用程序配置以及将此应用程序配置附加到托管服务器。然后从托管服务器的 mysql.xml 配置文件中导入值、对某些值进行更改以及将新的配置文件推送回托管服务器。

本教程基于示例：Travel Manager 应用程序和 XML 配置文件 (第 283 页)中描述的 Travel Manager 示例应用程序。

非 DTD XML 文件 mysql.xml 示例

以下是 Travel Manager 应用程序的 XML 配置文件的内容：

```
<?xml version="1.0" ?>
<db-config>
  <db-host>localhost</db-host>
  <db-name>wrightevents</db-name>
  <db-user>root</db-user>
  <db-password>hp-pass</db-password>
</db-config>
```

1.创建 XML 配置模板

使用 SA 客户端基于 mysql.xml 配置文件创建配置模板。

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 打开“应用程序配置”节点，然后打开“模板”节点。此操作将显示所有操作系统组。
3. 打开一个操作系统节点并在其中一个操作系统节点下选择特定操作系统。对于本示例，请选择可以安装此应用程序配置的其中一台服务器的操作系统。
4. 在“操作”菜单中，选择“新建”。
5. 在“属性”视图中，输入以下信息：
   - **名称**：TM-MySql
   - **描述**：这是 Travel Manager 应用程序的 mysql.xml 配置文件的模板。
   - **位置**：可以将 SA 库中的默认位置保留为 /，或选择其他位置用于存储模板文件。请注意，包含此模板的文件夹客户设置必须包括此应用程序配置对象的客户设置。否则，可用模板的列表中将不会包括此模板。有关文件夹设置的详细信息，请参见《SA 10.50 管理指南》中的“文件夹权限”一节。
   - **版本**：0.1。
   - **类型**：模板文件
   - **分析器语法**：XML 语法
   - **OS**:选择可安装此配置模板的所有操作系统。
   - 选择“文件”>“保存”。
6. 针对下一个任务保留打开“模板”窗口。

2.添加 XML 设置

由于 XML 配置文件 mysql.xml 提供了分析此文件内容所需的大量结构设置，因此 SA 中的 XML 配置模板在 XML 注释中仅需要三种信息：ACM-NAMESPACE、ACM-FILENAME-KEY 和 ACM-FILENAME-DEFAULT。

1. 在导航窗格中选择“内容”视图。

   ```
   <!--
   ACM-NAMESPACE = /TravelManager
   ACM-FILENAME-KEY = /files/TravelManager
   ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
   -->
   ```

2. 复制以下 XML 并将其粘贴到内容窗格中：
3. 选择“验证”按钮以确保 XML 有效。
4. 选择“文件”>“保存”菜单保存模板。
5. 选择“文件”>“关闭”菜单。

这些 XML 行定义以下内容：

- ACM-NAMESPACE:指定每个配置模板所需的唯一命名空间。在本例中，由于已建立 Travel Manager 应用程序的命名空间，因此可重用此根命名空间并附加服务名称。例如：
- ACM-NAMESPACE = /TravelManager/web/mysql
- ACM-FILENAME-KEY:指定命名空间中用于存储已生成文件的文件名的键的路径。
- ACM-FILENAME-DEFAULT:指定目标服务器上用于存储 Travel Manger 应用程序 mysql.xml 文件的路径。对于特定服务器或服务器组，可覆盖此路径。

3.创建要包含此模板的应用程序配置

在此步骤中创建要包含此配置模板的应用程序配置对象。

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 打开“应用程序配置”节点，然后打开“配置”节点。此操作将显示所有操作系统组。

3. 打开操作系统节点并选择在先前步骤中用于创建此模板的同一个操作系统。为此应用程序配置指定的 OS 必须是为此模板指定的 OS 的子集。
4. 在“操作”菜单中，选择“新建”。
5. 在“文件配置”屏幕的“属性”视图中，指定以下属性：
   - **名称**：Tm-MySql-Config
   - **描述**：这是 Travel Manager 应用程序的 mySQL 配置文件的应用程序配置。
   - **位置**：可以将 SA 库中的默认文件夹位置保留为 /，或选择其他文件夹用于存储此应用程序配置。请注意，包含应用程序配置的文件夹的客户设置必须包括您要向其推送此应用程序配置的托管服务器的客户设置。有关文件夹设置的详细信息，请参见《SA 10.50 管理指南》中的“文件夹权限”一节。
   - **版本**：0.1。
   - **OS**:选择可安装此应用程序配置的托管服务器中的一个或多个操作系统。
6. 选择“配置值”。
7. 选择添加按钮 "+" 或“操作”>“添加”菜单添加此模板。
8. 在“选择配置模板”屏幕中，选择 TM-MySql 模板，然后选择“确定”。此操作会将此模板添加到此应用程序配置对象。
9. 选择“文件”>“保存”，然后选择“文件”>“关闭”。现在已准备好将应用程序配置和其中的配置模板附加到存储此配置文件的服务器。

4.将应用程序配置附加到托管服务器

既然已创建应用程序模板和应用程序配置对象，您需要将此应用程序配置附加到已安装 Travel Manager 应用程序的服务器并指定托管服务器上用于存储 `mysql.xml` 文件的路径。

要将应用程序配置附加到服务器，请执行以下操作：

1. 在 SA 客户端导航窗格中选择“设备”，然后选择“服务器”>“所有托管服务器”。
2. 查找可模拟安装 Travel Manager 应用程序配置的服务器。此服务器的操作系统必须与此应用程序配置中指定的其中一个操作系统匹配。
3. 选择此服务器，然后在“操作”菜单中选择“打开”。
4. 在此服务器屏幕中，选择“管理策略”选项卡。
5. 在导航窗格中，选择“配置的应用程序”。此时将显示附加到此服务器的应用程序配置。

6. 选择“安装的配置”选项卡。
7. 在“操作”菜单中，选择“添加配置”。
8. 在“选择应用程序配置”屏幕中，选择 Tm-MySql-Config 应用程序配置。
9. 在“实例名称”字段中，输入“默认 mysql 配置值”。此操作将创建服务器实例级别的值集。有关详细信息，请参见服务器级别的值集编辑器 (第 232 页)。
10. 选择“确定”。此时应用程序配置已附加到服务器。
11. 选择“保存更改”按钮。针对下一个步骤保留打开服务器屏幕。

    下图显示了附加到服务器的 Tm-MySql-Config 应用程序配置以及“默认 mysql 配置值”值集。此值集为服务器实例级别的值集。

    附加到服务器的应用程序配置并突出显示服务器实例级别的值集

> **备注:**
> 请注意，此时如果向其添加应用程序配置的服务器具有多个 `mysql.xml` 配置文件的实例(由于此服务器正托管此应用程序的多个实例)，则可右键单击此配置的“默认 mysql 配置值”节点并选择“复制”。此操作将创建其他值集，在其中您可以设置指向此应用程序其他实例的文件名路径。有关不同级别值集的详细信息，请参见值集级别和值集继承 (第 222 页)。

5.为服务器配置应用程序配置设置

既然已将应用程序配置附加到托管服务器，那么需要为此服务器配置它并为配置文件设置值。

要从配置文件导入值 (如下所述)，请复制以上非 DTD XML 文件 mysql.xml 示例中列出的 XML 并将其粘贴到托管服务器上的目标文件 /var/www/html/we/mysql.xml 中。此操作将促使执行以下导入值步骤。

1. 展开 Tm-MySql-Config 节点以显示服务器实例级别的值集。此值集名为“默认 mysql 配置值”。
2. 在内容窗格的应用程序配置“值集编辑器”中配置以下设置：
   - **文件名**：托管服务器上的目标 XML 文件的原始路径和文件名显示在“文件名”字段的右侧。此值与模板中定义的“FILENAME-DEFAULT”的值相同。如果接受服务器的默认路径名，则可将此字段保留为空。如果要将此配置文件放置于此服务器的其他位置，请在“文件名”字段中为目标服务器上的目标 XML 文件设置正确的路径。
   - **编码**：选择托管配置文件的字符编码。默认为托管服务器上使用的编码。(请注意，不支持使用 UTF-16 编码。)
   - **保留格式**：如果要保留注释以及尽可能多的保留目标服务器中原始 XML 配置文件的顺序和位置，请选择此选项。SA 将尽可能多的保留目标文件。有关详细信息，请参见 在值集编辑器中设置字段 (第 224 页)。
   - **保留值**：当值集中没有提供任何值时，要保留服务器上实际配置文件中包含的值，请针对此选项选择“是”。将此选项设置为“是”之后，将使用目标文件的值，除非继承层次结构中任何级别的值覆盖这些值。如果将此选项设置为“否”并且值集中不存在任何值，则配置文件中将不会放置任何条目。有关详细信息，请参见 在值集编辑器中设置字段 (第 224 页)。
   - **显示所继承的值**：选择此选项可显示值集中的值和继承级别。取消选中时，仅显示当前继承级别的值集。选中时，显示值集中的所有值，包括在当前级别设置的值和所继承的值。此视图为只读。
3. 在值集编辑器中右键单击并选择“导入值”。导入值时将读取托管服务器上的 XML 文件并将此 XML 文件的内容复制到服务器实例级别的值集。
4. 要保存更改，请选择“保存更改”按钮。针对下一个步骤保留打开服务器屏幕。

6.编辑值并推送配置

最后步骤是在值集编辑器中编辑值，然后将配置推送到服务器。推送应用程序配置后，值集中的所有值都将替换目标托管服务器上配置文件中的值。应用程序配置中包含的任何脚本都将根据其脚本类型执行。如果目标服务器上不存在配置文件，则在执行推送时会创建此文件。

要编辑值并推送应用程序配置，请执行以下操作：

1. 通过在“值”列中进行编辑，可修改值集的一个或多个值。有关列的描述，请参见值集编辑器中的列 (第 225 页)。
2. 在为应用程序配置设置值之后，选择“预览”可查看服务器上的现有文件以及将被推送到此服务器的文件。
3. 选择“推送”以将新的应用程序配置复制到服务器。
4. 在“推送配置”屏幕中选择“启动作业”。检查推送作业的状态和结果。

XML 教程 2 - 创建 XML-DTD 配置模板

本节使用 Travel Manager 应用程序作为示例，显示如何创建 XML-DTD 配置模板以管理引用 DTD 的 XML 配置文件。有关 Travel Manager 示例的背景信息，请参见示例： Travel Manager 应用程序和 XML 配置文件 (第 283 页)。

首先将 XML-DTD 模板创建为文本文件，然后将此文本文件导入 SA 库中并将其添加到应用程序配置对象。然后将此应用程序配置附加到托管服务器。最后，编辑此应用程序配置中的某些值并将所做更改推送到托管服务器上的目标 XML 文件。

Travel Manager 基于 DTD 的 XML 文件示例: mysql.xml

对于 Travel Manager 应用程序，以下是 `mysql.xml` XML 配置文件。此文件存储在 `/var/www/html/we/mysql.xml` 中。

```
<?xml version="1.0"?>
<!DOCTYPE db-config PUBLIC "-//Williams Events//Travel Manager//EN" "mysql.dtd">
```

```
<db-config>
       <db-host>localhost</db-host>
       <db-name>wrightevents</db-name>
       <db-user>root</db-user>
       <db-password>hp-pass</db-password>
</db-config>
```

Travel Manager XML DTD 文件示例: mysql.dtd

对于 Travel Manager 应用程序，以下是随附的 mysql.dtd DTD。此文件存储在 /var/www/html/we/mysql.dtd 中。

```
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
```

1.在文本编辑器中创建 XML-DTD 模板

在本任务中，将使用文本编辑器创建 XML-DTD 配置模板的源。

要在文本编辑器中创建 XML-DTD 配置模板，请执行以下操作：

1. 在文本编辑器中，输入以下信息：

   ```
   <!--
   ACM-TIMEOUT = 1
   ACM-FILENAME-KEY = /files/TravelManager
   ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
   ACM-NAMESPACE = /TravelManager/
   ACM-DOCTYPE = db-config
   ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
   ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
   -->
   ```

   此信息为必要信息(除 ACM-TIMEOUT 以外)，应用程序配置分析器将使用此信息来读取要管理的 XML-DTD 和 XML 文件：

- ACM-TIMEOUT:(可选)代表在推送期间要添加到配置模板默认超时值(十分钟)的分钟数。
- ACM-FILENAME-KEY:定义命名空间中将用于存储 mysql.xml 文件名的位置。
- ACM-FILENAME-DEFAULT:定义托管服务器上 mysql.xml 文件的默认位置(绝对路径)。
- ACM-NAMESPACE:此值定义数据库中用于存储从托管服务器的 mysql.xml 文件中读取的值的位置。此命名空间必须唯一且路径必须以正斜杠 (/) 开头。
- ACM-DOCTYPE:定义 XML 文件中根元素的名称。此根元素遵循在目标 XML 配置文件中找到的开放 <!DOCTYPE 声明。
- ACM-DOCTYPE-SYSTEM-ID:定义托管服务器上关联 DTD 文件的名称。通常，此值可在 XML 配置文件中作为 DOCTYPE 元素中的 SYSTEM 特性找到。
- ACM-DOCTYPE-PUBLIC-ID:定义可表示 XML 文档公用标识符的字符串。通常，此值可在 XML 配置文件中作为 DOCTYPE 元素的 PUBLIC-ID 特性找到。

2. 保存此文件，并赋予名称 mysql-dtd.tpl。针对下一个任务保留打开此文件。

2.在值集编辑器中为元素描述添加客户设置

在本任务中，将向 XML-DTD 模板文件添加某些额外的信息，以便自定义目标 XML 文件中每个元素在 SA 客户端的值集编辑器中的显示方式。

可以向 XML-DTD 配置模板添加两个可选设置，用于支持您自定义目标 XML-DTD 配置文件中的元素在 SA 客户端值集编辑器中的显示方式：

- ACM-PRINTABLE:定义在 SA 客户端中显示 XML-DTD 模板时，将在值集编辑器中显示的 XML 文件中每个元素的标签。
- ACM-DESCRIPTION:定义当用户在 SA 客户端的值集编辑器中将鼠标指针悬停在 ACM-PRINTABLE 中定义的字段时显示的鼠标悬停文本。

有关这些元素如何显示在 SA 客户端的值集编辑器中的示例，请参见 XML-DTD 模板图。

此示例使用显式方法将这些客户设置放置于 XML-DTD 模板中。有关放置客户设置的方法的详细信息，请参见显式和位置显示设置 (第 287 页)。

要在 XML-DTD 模板中添加客户设置，请执行以下操作：

1. 在打开 mysql-dtd.tpl 文件的文本编辑器中，为 DTD 引用的每个 XML 元素添加以下信息。例如，在模板的主要信息后，使用以下三个 ACM 设置标记添加源 XML 文件中包含的每个元素的列表，然后为每个元素创建 XML 注释：
   - ACM-ELEMENT:声明以下 ACM-PRINTABLE 和 ACM-DESCRIPTION 设置将描述的 XML 文件中的元素。此选项默认设置为 DTD 文件中此部分之前出现的任何元素或特性。
   - ACM-PRINTABLE:为元素设置在值集编辑器中显示时的简短描述性标签。
   - ACM-DESCRIPTION:设置元素的鼠标悬停文本。

   XML-DTD 模板文件示例应类似如下：

```
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
<!--
ACM-ELEMENT = db-config
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that
contains the information needed to connect to a database.
-->
<!--
ACM-ELEMENT = db-host
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer
(the server) on which the database engine is running.
-->
<!--
ACM-ELEMENT = db-name
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->
<!--

ACM-ELEMENT = db-user
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used
to connect to the database.
```

```
-->
<!--
ACM-ELEMENT = db-password
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to
connect to the database.
-->
```

2. 保存并关闭文件。

3.导入 XML-DTD 配置文件

在本任务中，将导入模板文件并创建用于管理目标 XML 和 DTD 文件的新配置模板。

要将 XML-DTD 配置文件导入 SA 库中，请执行以下操作：

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“模板”节点。打开操作系统组并导航到应用此模板文件的操作系统。请注意，模板可应用于多个操作系统。例如，可以选择 Red Hat 操作系统的一个版本。
3. 在“操作”菜单中，选择“导入模板”。
4. 导航到上一个步骤中创建的文件并选择此文件。如果不是默认编码，请设置此编码。
5. 选择“打开”。此操作将导入模板文件并将其显示在“模板”屏幕中。
6. 选择“属性”视图并输入以下信息：
   - **名称**：mysql-dtd.tpl
   - **描述**：这是 Travel Manager 应用程序的 mysql.dtd (mysql.xml) 文件的模板。
   - **位置：指定 SA 库中用于存储模板的位置。**
   - **版本**：0.1。
   - **类型**：模板文件
   - **分析器语法**：XML DTD 语法。
   - **OS**:选择相应的操作系统。
7. 选择“内容”视图以显示刚导入的模板文件的内容。
8. 选择“验证”按钮，以在继续执行操作之前确定语法有效。
9. 选择“文件”>“保存”。
10. 选择“文件”>“关闭”。

4.创建应用程序配置对象。

应用程序配置是容纳配置模板文件的容器。在本步骤中，将创建应用程序配置并导入模板。

1. 在 SA 客户端导航窗格中，选择“库”，然后选择“按类型”选项卡。
2. 查找并打开“应用程序配置”节点。打开“配置”节点。打开操作系统组并选择应用此应用程序配置的操作系统。请注意，应用程序配置可应用于多个操作系统。可在后面的步骤中进行更改。
3. 选择“操作”>“新建”菜单。此时将显示“配置”屏幕，可在其中指定应用程序配置的属性和内容。
4. 在“属性”视图中，指定以下信息：
   - **名称：TM-mysql-dtd**
   - **描述**：这是 Travel Manager 应用程序的 mysql.xml 和 mysql.dtd 文件的应用程序配置。
   - **位置：指定 SA 库中存储应用程序配置的位置。**
   - **版本**：0.1
   - **OS**:选择相应的操作系统。
5. 在“内容”视图中，选择“操作”>“添加”菜单或 "+" 按钮将模板添加到应用程序配置。
6. 在“选择配置模板”屏幕中，选择 mysql-dtd.tpl 模板文件。
7. 选择“确定”。
8. 选择“文件”>“保存”保存应用程序配置。
9. 选择“文件”>“关闭”。

5.将应用程序配置附加到托管服务器

在本任务中，将应用程序配置附加到已安装 Travel Manager 应用程序的服务器，然后输入 `mysql.dtd` 配置文件的路径名。

要将应用程序配置附加到服务器，请执行以下操作：

1. 从 SA 客户端导航窗格中，选择“设备”>“服务器”>“所有托管服务器”。
2. 选择服务器，然后在“操作”菜单中选择“打开”。请确保所选服务器的操作系统与应用程序配置和模板上指定的操作系统匹配。
3. 选择“管理策略”选项卡。
4. 选择“配置的应用程序”节点。
5. 选择“安装的配置”选项卡。
6. 在“操作”菜单中，选择“添加配置”。
7. 在“选择应用程序配置”屏幕中，选择应用程序配置 TM-mysql-dtd。
8. 在“实例名称”字段中，输入“mysql.xml 的值集 1”。
9. 选择“确定”。此时会将此应用程序配置附加到此服务器。
10. 选择“保存更改”按钮。
11. 针对下一个步骤保留打开应用程序配置屏幕。

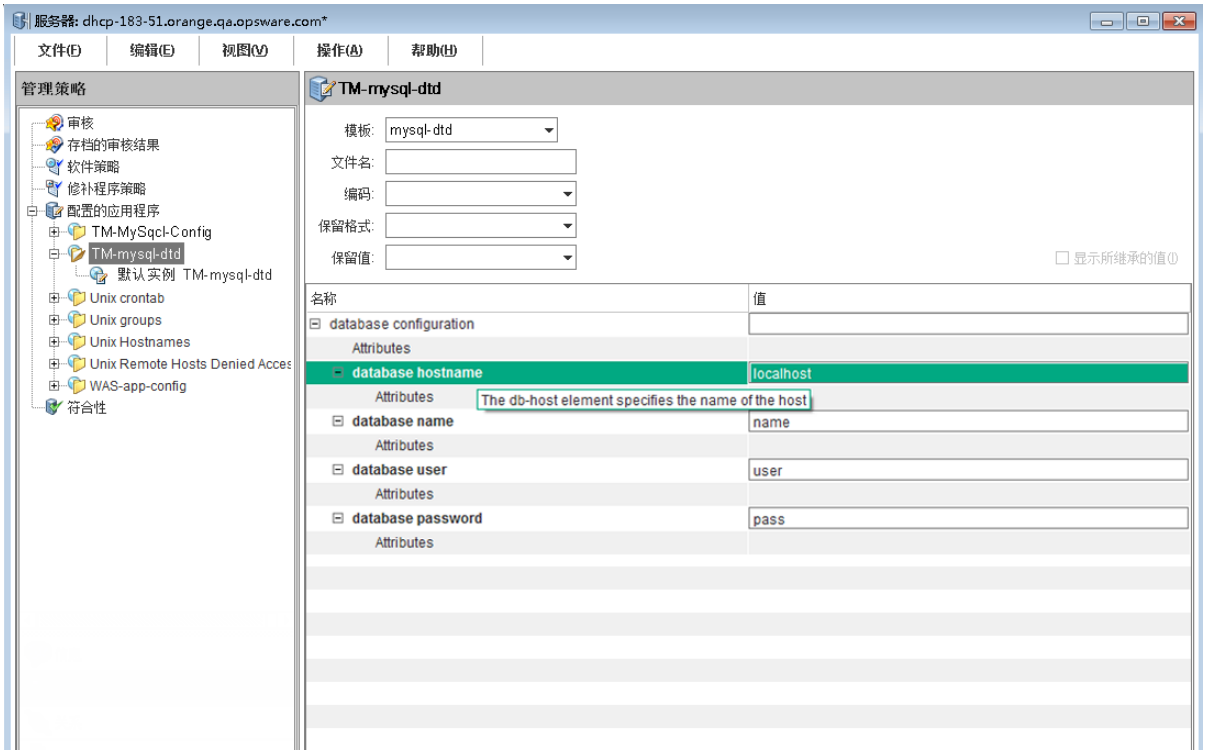
6.从配置文件导入值

下一个步骤是设置值集中的值。虽然可以手动设置值集中的值，但是最简单的方式是从现有配置文件导入值。在本步骤中，将从服务器上的配置文件导入值。

要从配置文件导入值 (如下所述)，请复制以上 Travel Manager 基于 DTD 的 XML 文件示例: mysql.xml 中列出的 XML 并将其粘贴到托管服务器上的目标文件 /var/www/html/we/mysql.xml 中。复制以上 Travel Manager XML DTD 文件示例: mysql.dtd 中列出的 DTD 并将其粘贴到目标文件 `/var/www/html/we/mysql.dtd` 中。此操作将促使执行以下导入步骤。

1. 在服务器“管理策略”中，打开“配置的应用程序”节点。此时将显示附加到此服务器的应用程序配置。
2. 打开“TM-mysql-dtd”节点。此时将显示此服务器实例值集，这些值集是“TM-mysql-dtd”节点下的节点。
3. 选择“mysql.xml 的值集 1”节点。它是 mysql-dtd.tpl 配置模板的服务器实例值集。
4. 右键单击“值”列下的任何值，然后选择“导入值”菜单。
5. 在确认对话框中，选择“是”。此时会将文件 /var/www/html/we/mysql.xml 中的值导入服务器实例级别的值集中。
6. 选择“保存更改”按钮。下图显示了 XML-DTD 模板，其中包含服务器实例值集和从 ACM-DESCRIPTION 元素显示的鼠标悬停文本。

XML-DTD 配置模板的值集和鼠标悬停文本



7. 针对下一个步骤保留显示应用程序配置。

7.编辑值并推送配置

最后一个步骤是在值集编辑器中编辑值，然后将配置推送到服务器。推送应用程序配置后，值集中的所有值都将替换目标托管服务器上配置文件中的值。此外，还会执行应用程序配置中的所有脚本。如果目标服务器上不存在配置文件，则在执行推送时会创建此文件。

要编辑值并推送应用程序配置，请执行以下操作：

1. 确保通过在导航窗格中选择“mysql.xml 的值集 1”节点，显示此服务器实例级别的值集。
2. 在“值”列下修改密码值。
3. 选择“保存更改”。
4. 选择“预览”按钮，以显示服务器上现有配置文件和将被推送到服务器的配置文件之间的差异。

5. 在检查比较屏幕之后，选择“关闭”按钮。
6. 选择“推送”。此时将显示“推送配置”向导。
7. 选择“启动作业”。此时将启动推送操作。
8. 检查推送作业的“作业状态”。选择任意步骤可显示该步骤的详细信息。
9. 完成作业后，选择“关闭”按钮。
10. 您可以登录此服务器并检查 `mysql.xml` 配置文件，验证服务器上是否已更新此文件。

CML 教程 1 - 为简单 Web 应用程序服务器创建应用程序配置

本节演示了如何为运行在两台服务器上的 Web 应用程序服务器设置和管理简单配置文件。每台服务器运行此 Web 应用程序服务器，并需要对它们进行单独配置。此教程显示了如何创建应用程序配置、配置模板、值集以及应用程序配置的两个实例(每台服务器一个实例)。最后，它将显示如何将应用程序配置推送到每台服务器。

1.确定要管理的配置文件

Web 应用程序服务器使用一个名为 WASconfig.txt 的配置文件。此文件位于 /opt/WAS/WASconfig.txt 目录中。此文件的内容如下所示：

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

2.为此配置文件创建模板

可以采用以下两种方式之一创建模板：

- 在文本文件中创建模板并将此文本文件导入 SA 库中。
- 直接在 SA 库中创建模板。

以下描述了这两种方法。请选择一种方法并按照步骤执行操作。

创建模板文件并将其导入 SA 中

1. 在文本编辑器中，将此配置文件导入空的文件：

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

2. 创建一个模板，该模板使用 CML 对此文件进行建模。首先添加注释块和所需的 CML 元数据，用于定义命名空间和目标配置文件名。
   - 命名空间可定义数据库中将用于存储此模板信息的键。
   - 文件名关键字可定义数据库中将存储默认文件名的位置。
   - 默认文件名可指定将用于生成的配置文件的名称。

```
@#########################################################
# /opt/WAS/WASconfig.txt                                #
# Version 1.0                                           #
# Author <name>                                         #
#########################################################@
@!namespace=/WAS-server-namespace/@
@!filename-key=”/WAS-server”@
@!filename-default=”/opt/WAS/WASconfig.txt”@
size=1000
dir=/tmp/WAS_01
primary=yes
```

3. 然后使用 CML 标记将此配置文件的可变部分更改为变量：

```
@#########################################################
# /opt/WAS/WASconfig.txt                                #
# Version 1.0                                           #
# Author <name>                                         #
#########################################################@
@!namespace=/WAS-server-namespace/@
@!filename-key=”/WAS-server”@
@!filename-default=”/opt/WAS/WASconfig.txt”@
size=@value_of_size;int@
dir=@value_of_dir;string@
primary=@value_of_primary;boolean@
```

4. 使用 ".tpl" 扩展名保存此文件，例如 `WASconfig_txt.tpl`。
5. 将此模板文件导入 SA 库中。按照导入和验证模板文件 (第 263 页)中的步骤执行操作。

直接在 SA 中创建模板文件

1. 在 SA 客户端中，选择“库”选项卡。
2. 选择“按类型”选项卡。
3. 打开“应用程序配置”节点和“模板”节点。导航到运行此应用程序的 OS 系列和 OS 版本。对于本示例，请选择 Red Hat Enterprise Linux AS 4。
4. 选择“操作”>“新建”。此操作将显示“模板”屏幕。
5. 输入模板名称“WASconfig_txt.tpl”和简要描述。选择 SA 库中用于存储此模板文件的位置。设置版本字符串。将“类型”设置为“模板文件”。将“分析器语法”设置为“CML 语法”。
6. 选择“内容”视图显示文本编辑器。
7. 键入或粘贴 CML 文本。此文本与以上显示的 CML 文本相同。
8. 选择“操作”>“验证”检查 CML 的语法。进行任何所需的更正。
9. 选择“文件”>“保存”以保存模板。
10. 关闭此模板屏幕。

3.创建应用程序配置对象。

创建要包含此配置模板的应用程序配置对象。

1. 在 SA 客户端中，选择“库”选项卡。
2. 选择“按类型”选项卡。
3. 打开“应用程序配置”节点和“配置”节点。导航到运行此应用程序的 OS 系列和 OS 版本。对于本示例，请选择 Red Hat Enterprise Linux AS 4。
4. 选择“操作”>“新建”。此操作将显示“配置”屏幕。
5. 输入应用程序配置的名称“WAS-app-config”、简要描述和版本字符串。选择 SA 库中用于存储此应用程序配置的位置。
6. 选择“文件”>“保存”以保存应用程序配置。

4.将模板文件添加到应用程序配置对象

1. 打开在上一个步骤中创建的“WAS-app-config”应用程序配置对象。
2. 选择“配置值”视图。

3. 选择 "+" 按钮或选择“操作”>“添加”。将显示“选择配置模板”屏幕。
4. 选择 "**WASconfig_txt.tpl**" 模板文件并选择“确定”。
5. 选择“文件”>“保存”以保存对此应用程序配置对象所做的更改。
6. 选择“文件”>“关闭”以关闭此应用程序配置对象。

5.将应用程序配置对象附加到服务器

两台服务器将运行 Web 应用程序服务器，分别是 RHEL001 和 RHEL008。RHEL001 是主服务器，RHEL008 是辅助服务器。通过将此应用程序配置对象附加到这两台服务器，可创建此应用程序配置的两个实例，如下所示：

1. 在 SA 客户端中查找主服务器 RHEL001。
2. 选择 RHEL001 服务器，然后选择“操作”>“打开”。
3. 选择“管理策略”选项卡。
4. 选择“配置的应用程序”节点。
5. 选择“操作”>“添加配置”菜单。
6. 选择 "**WAS-app-config**" 应用程序配置对象。
7. 在“实例名称”字段中输入“WAS-app-config 主实例”并选择“确定”。
8. 选择“保存更改”。此操作将为服务器 RHEL001 创建此应用程序配置的一个实例。
9. 对辅助服务器 RHEL008 重复执行以上步骤，但不在“实例名称”字段中输入“WAS-app-config 辅助实例”，然后选择“确定”。此操作将为服务器 RHEL008 创建此应用程序配置的第二个实例。

6.设置默认值

两台服务器的配置文件所需值如下所示：

运行 Web 应用程序服务器的两台服务器的配置值

服务器：RHEL001	服务器：RHEL008
size=1000 dir=/tmp/WAS_001	size=1000 dir=/tmp/WAS_008

运行 Web 应用程序服务器的两台服务器的配置值 (续)

服务器：RHEL001	服务器：RHEL008
primary=yes	primary=no

您可以为可由单台服务器继承或可被每台服务器覆盖的配置文件设置默认值。如果单台服务器不会覆盖此默认值，则将使用继承的默认值。

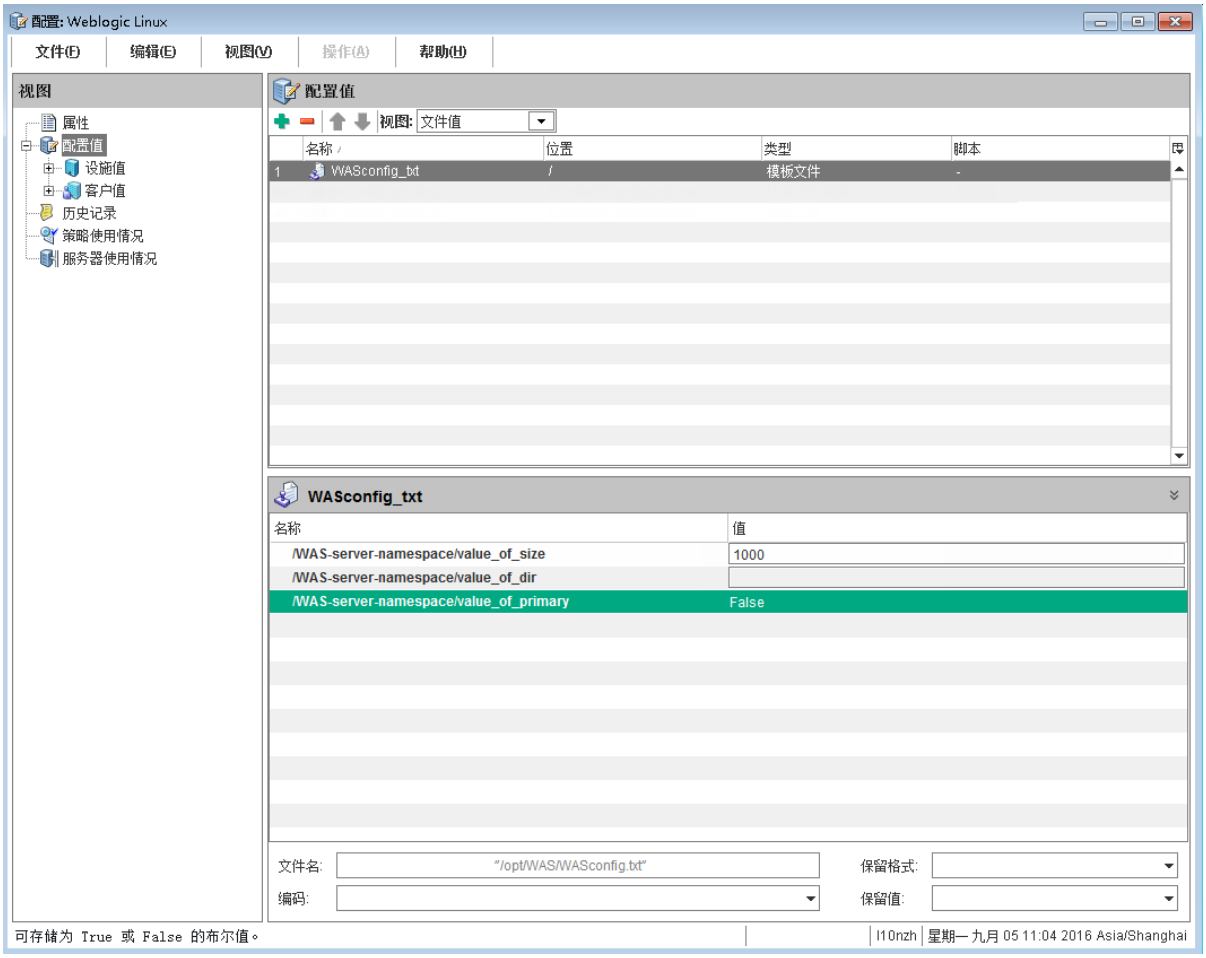
下表显示哪些值将被设置为默认值以及哪些值将由单台服务器设置：

表 7：针对 Web 应用程序服务器的配置文件的应用程序级别默认值

默认值	描述
size=1000	将此值设置为应用程序级别默认值 1000。附加到此应用程序配置的所有服务器都将使用此值，除非这些服务器覆盖此值。
dir	不会将此值设置为默认值。每台服务器将在服务器级别或服务器实例级别设置此值。
primary=no	将此值设置为应用程序级别默认值 “no”。附加到此应用程序配置的所有服务器都将使用此值，除非这些服务器覆盖此值。

设置应用程序级别默认值

1. 打开以上创建的应用程序配置对象。
2. 选择“配置值”视图。
3. 选择 **WAS-config-template.tpl** 模板文件。
4. 在视图下拉列表中选择“文件值”。此操作将显示此模板的应用程序级别默认值。
5. 将“value_of_size”设为 1000 并将“value_of_primary”设为“False”(注意大小写)，如下所示。请勿设置“value_of_dir”的默认值，因为每台服务器将需要设置此值。



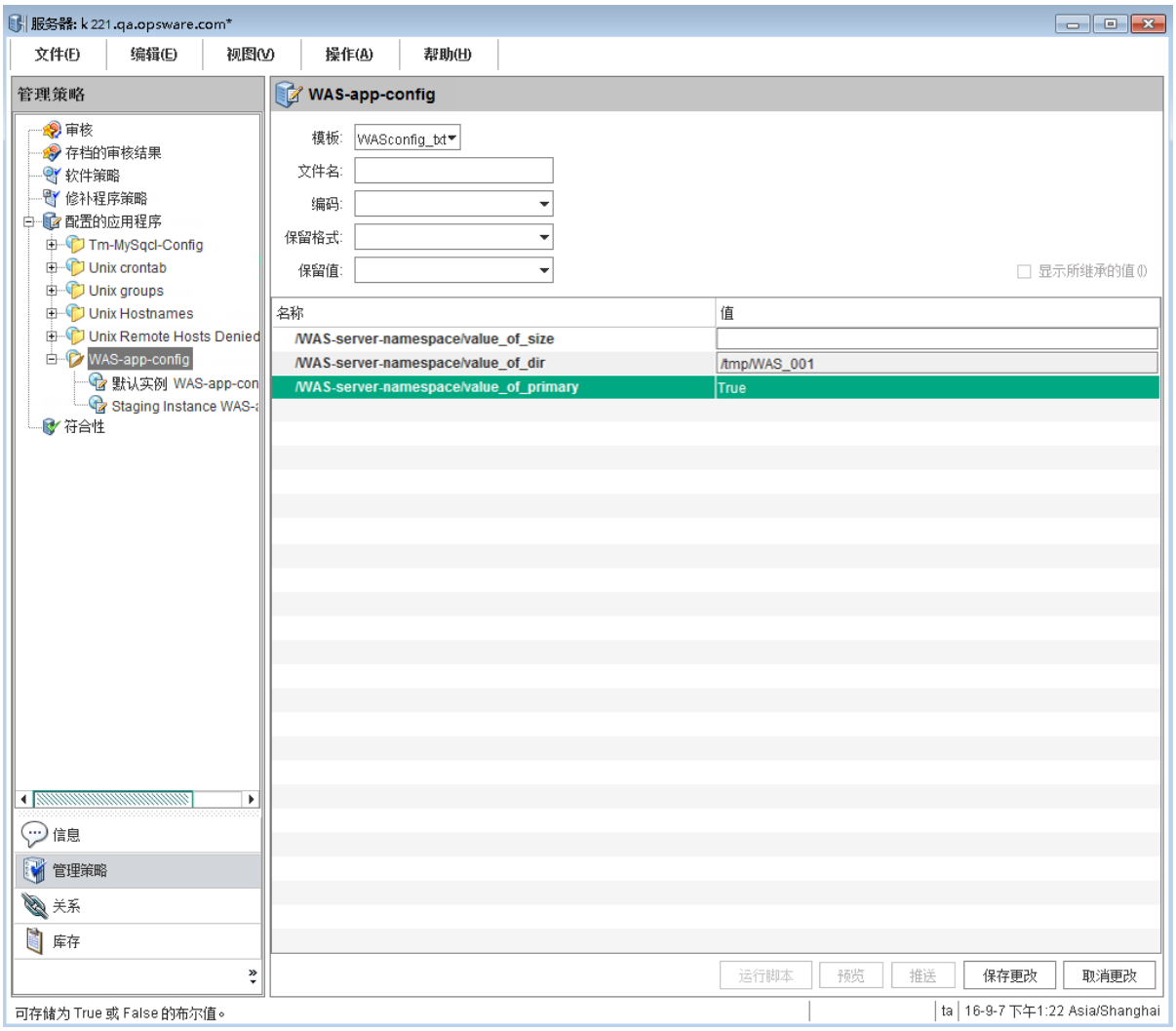
6. 选择“文件”>“保存”以保存应用程序级别默认值。
7. 选择“文件”>“关闭”。

设置 RHEL001 的服务器级别默认值

服务器 RHEL001 需要在服务器级别设置 `dir=/tmp/WAS_001` 和 `primary=yes`。它不需要设置大小，因为它可以使用应用程序级别的值集。

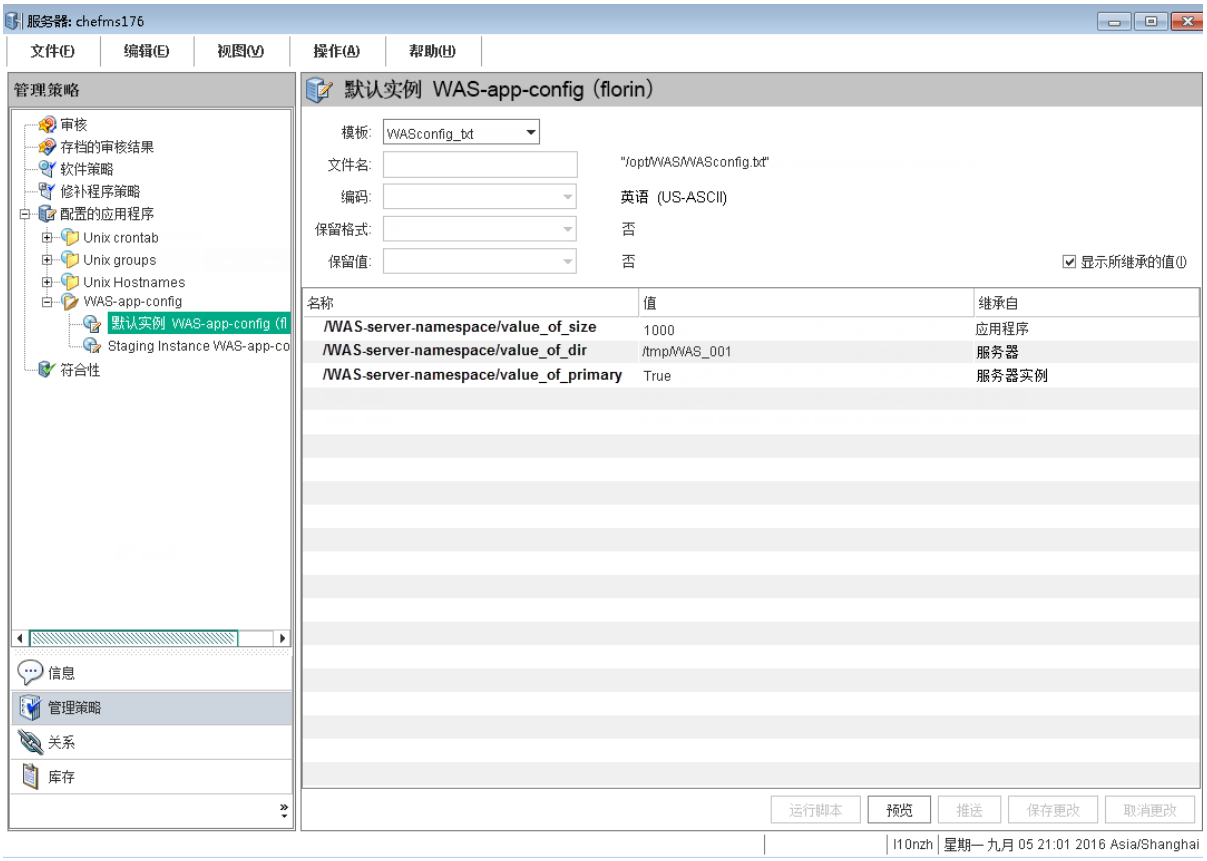
1. 在 SA 客户端中查找 RHEL001 服务器。
2. 选择 RHEL001 服务器，然后选择“操作”>“打开”。
3. 选择“管理策略”选项卡。
4. 打开“配置的应用程序”节点以显示“WAS-app-config”应用程序配置对象。

5. 选择附加到此服务器的“WAS-app-config”应用程序配置对象。此操作将显示服务器级别的默认值集。服务器级别的值集将应用于服务器上应用程序配置的所有实例，除非被服务器实例级别所覆盖。
6. 将“value_of_dir”的服务器级别默认值设置为“/tmp/WAS_001”，如下所示。请勿设置“value_of_size”或“value_of_primary”的默认值，因为这些值将从应用程序级别继承。



7. 选择“保存更改”按钮，或“文件”>“保存”菜单，以保存“服务器级别默认值”。
8. 打开 WAS-app-config 节点以显示应用程序配置实例“WAS-app-config 主实例”。
9. 选择实例“WAS-app-config 主实例”。此操作将显示**实例级别默认值**。这是最低级别的值集，将覆盖所有其他级别。请注意，尚未定义任何实例级别的值。

10. 选择“显示所继承的值”可显示将从应用程序级别默认值和服务器级别默认值继承的值。请注意，“value_of_size”和“value_of_primary”从应用程序级别继承，“value_of_dir”从服务器级别继承。
11. 取消选中“显示所继承的值”，以便可以设置实例级别默认值。
12. 将“value_of_primary”设为“True”(注意大小写)。
13. 选择“保存更改”按钮，或“文件”>“保存”菜单，以保存“实例级别默认值”。
14. 再次选择“显示所继承的值”可显示从应用程序级别、服务器级别和实例级别继承的值。请注意，“value_of_size”从应用程序级别继承，“value_of_dir”从服务器级别继承，“value_of_primary”从实例级别继承，如下所示。



设置 RHEL008 的服务器级别默认值

服务器 RHEL008 需要在服务器级别设置 `ddir=/tmp/WAS_008`它不需要设置大小或主，因为它可以使用应用程序级别的值集。

1. 在 SA 客户端中查找 RHEL008 服务器。
2. 选择 RHEL008 服务器，然后选择“操作”>“打开”。
3. 选择“管理策略”选项卡。
4. 打开“配置的应用程序”节点以显示“WAS-app-config”应用程序配置对象。
5. 选择附加到此服务器的“WAS-app-config”应用程序配置对象。此操作将显示服务器级别的默认值集。服务器级别的值集将应用于服务器上应用程序配置的所有实例，除非被服务器实例级别所覆盖。
6. 将“value_of_dir”的服务器级别默认值设置为“/tmp/WAS_008”。请勿设置“value_of_size”或“value_of_primary”的默认值，因为这些值将从应用程序级别继承。
7. 选择“保存更改”按钮，或“文件”>“保存”菜单，以保存“服务器级别默认值”。
8. 打开 WAS-app-config 节点以显示应用程序配置实例“WAS-app-config 辅助实例”。
9. 选择实例“WAS-app-config 辅助实例”。此操作将显示**实例级别默认值**。这是最低级别的值集，将覆盖所有其他级别。请注意，尚未定义任何实例级别的值。
10. 选择“显示所继承的值”可显示将从应用程序级别默认值和服务器级别默认值继承的值。请注意，“value_of_size”和“value_of_primary”从应用程序级别继承，“value_of_dir”从服务器级别继承。

7.将实际配置文件与配置模板进行比较。

通过在服务器屏幕中选择“预览”按钮，可以选择将应用程序配置中指定的值与服务器上配置文件中的实际值进行比较。以下显示了 RHEL001 服务器上尚无配置文件时服务器上的比较：

以下显示了服务器上现有配置文件不同于应用程序配置中指定值时的比较：

8.将配置更改推送到服务器

1. 在 SA 客户端中查找 RHEL001 服务器。
2. 选择 RHEL001 服务器，然后选择“操作”>“打开”。
3. 选择“管理策略”选项卡。

4. 打开“配置的应用程序”节点以显示“WAS-app-config”应用程序配置对象。
5. 打开 **"WAS-app-config"** 应用程序配置节点以显示实例“WAS-app-config 主实例”。
6. 选择实例“WAS-appconfig 主实例”。
7. 选择“推送”按钮。
8. 可以选择“启动作业”按钮以接受计划和通知的作业默认值，也可以选择“下一步”。
9. 在“计划”屏幕中，可指定运行推送配置作业的时间。选择“下一步”。
10. 在“通知”屏幕中，可指定作业成功或失败时要接收电子邮件消息的一个或多个人员。此外，还可以指定工单标识符。选择“下一步”。
11. 选择“启动作业”。SA 从此模板和值集生成配置文件，然后将生成的配置文件推送到服务器并显示结果。
12. 选择“关闭”。

有关显示更加复杂的配置文件的教程，请参见CML 教程 2 - 创建 Web 服务器配置文件的模板 (第 316 页)。

CML 教程 2 - 创建 Web 服务器配置文件的模板

本教程将说明如何使用配置建模语言 (CML) 基于名为 UrlScan.ini 的 Microsoft Internet 信息服务 (IIS) Web 服务器配置文件创建配置模板。您将使用 CML 语言基于此文件创建模板文件，以便在托管服务器上管理此模板文件。

虽然本教程不会为您提供有关 CML 的所有信息，但是从 UrlScan.ini 创建 CML 模板将帮助您基本了解 CML 以及从配置文件创建配置模板的过程。

UrlScan.ini 文件示例中列出了 UrlScan.ini 文件的示例。完整 url_scan_ini.tpl CML 模板中列出了一个完整的 CML 文件。

要完成本教程，您应具备以下条件：

- `UrlScan.ini` 的文档。此文档随 Microsoft IIS 文档一起提供。
- `UrlScan.ini` 文件。
- 用于创建 CML 文件的文本编辑器。

1.分析本地配置文件和文档

一旦确定要管理的应用程序配置文件之后，首先应分析此本地配置文件及其文档。确保了解此配置文件的用途、文件中的所有元素以及此配置文件所管理的数据种类。

本教程将使用 UrlScan.ini 文件。UrlScan.ini 文件示例中列出了一个示例。通过 UrlScan.ini 文件，系统管理员可以配置 Microsoft Internet 信息服务 (IIS) Web 服务器。UrlScan.ini 文件由多个部分组成，例如 [Options]、[AllowVerbs]、[DenyVerbs]、[DenyHeaders]、[AllowExtensions] 和 [DenyExtensions]。IIS 管理员可以为每个部分设置不同的配置，以允许或禁止 IIS 服务器上某些种类的 HTTP 请求。无需按任何特定顺序排列这些部分。但是必须有序组织每个部分中的信息。例如，[AllowVerbs] 部分必须后跟被允许访问网站的特定 HTTP 请求。

UrlScan.ini 包含字符串列表(例如动词列表和文件扩展名)和选项(采用布尔值“1”表示 True 或采用“2”表示 False)。

2.创建 CML 注释块

CML 模板是一个名称带有 .tpl 文件扩展名的简单文本文件。使用文本编辑器创建名为 Url_Scan_ini.tpl 的新文本文件。.tpl 扩展名是 SA 针对 CML 模板使用的典型(但可选)文件扩展名。

使用有关此模板的信息在文件顶部创建 CML 注释块，如下所示：

```
@################################################
# \system32\inetsrv\urlscan.ini (Windows)    #
# Version 1.0                                  #
# Joe Author (joe_author@your_company.com)   #
################################################@
```

CML 注释标记使用以下语法：

```
@# <one line comment>
```

或

```
@## <comments spanning multiple lines>
        <comments spanning multiple lines> #@
```

3.创建 CML 设置指令

设置部分将指示分析器如何解释 CML 文件。所有 CML 文件都需要 namespace、filename-key 和 filename-default 指令。以下显示的其他指令为可选指令。这些指令定义了空格处理、布尔值、注释格式和排序规则。

要创建基本设置部分，请在 CML 模板中的注释块后输入以下信息：

```
@!namespace=/security/@
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@
```

请注意，每个 CML 指令标记以字符“@!”开头，以字符“@”结尾。

下行将两个 CML 指令合并在一行中：

```
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
```

此行还可写为两个单独的行，如下所示：

```
@!filename-key="/test"@
@!filename-default="/c/UrlScan.ini"@
```

设置指令

下图说明了设置指令。

CML 模板设置指令

CML 标记	描述
@!namespace=/security/@	@!namespace 指令定义此 CML 模板将使用的命名空间。它定义数据库中将存储此 CML 模板使用的值的位置。每个 CML 模板应使用其各自的命名空间，以便名称不与其他模板相冲突。 在本例中，命名空间为 /security。所有值集都将存储在此命名空间中。
@!filename-key="/files/urlscan_ini";filename-default="/c/urlscan.ini"@	@!filename-key 指令定义命名空间中将存储文件名的位置。如果值以“/”开头，则它将定义单独的命名空间。如果值不以“/”开头，则它将附加到由 @!namespace 指令定义的命名空间。 在本例中，默认文件名将存储在数据库的“/file/urlscan-ini”命名空间下。

CML 模板设置指令(续)

CML 标记	描述
	@!filename-default 指令定义服务器上将用于保存本地配置文件的默认路径。可使用 SA 客户端更改此路径。 在本例中，将配置文件推送到托管服务器之后，文件将被放置于 /c/urlscan.ini 中。 请注意，路径名只能使用正斜杠。
@!optional-whitespace@	此指令指示在配置文件的项之间空格是可选的。例如，如果设置此选项，则以下任一条目均将有效： Key = "value" Key="value"
@!boolean-yes-format="1";boolean-no-format="0"@	此指令定义配置文件中允许的布尔值。在此情况中，true 由字符 1 表示，false 由字符 0 表示。禁止使用任何其他布尔值。
@!line-comment-is-semicolon@	指示分析器忽略配置文件中分号后的任何内容。这允许在本地配置文件中的每个注释前使用分号进行注释。
@!unordered-lines@	指示分析器配置文件中的部分可以按任何顺序排列。如果使用 ordered-lines，则配置文件必须与模板的顺序保持一致。

4.定义 [Options] 部分 — 打开块

现在已准备好将 CML 指令添加到模板。您将在 CML 中建模的 UrlScan.ini 文件的第一个部分是 [Options] 部分，它包含配置文件的多个选项。

在 CML 中，如果配置文件中的信息部分具有多种数据(需 CML 分析器采用不同方式读取的数据)，则可打开“块”来单独处理每个信息部分。通常，可以在 CML 中打开一个块，为 CML 文件的某个部分定义特殊的分析器规则。[Options] 部分具有两个信息“块”：此部分的标题，即“[Options]”，以及该部分中的所有选项。由于这些块相互从属，因此您将在不同的级别设置它们，在级别 1 设置第一个块(此部分的标题)，在级别 2 设置第二个块(此部分的内容)。采用这种方式嵌套块可以在分析器读取块时将各个部分一起保持在块中。

1. 要定义 [Options] 部分，请输入以下行：

```
@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
```

2. 在 UrlScan.ini 文件中，[Options] 部分包含一个键值对列表。在两个级别使用块标记 ([) 集，因为此部分中存在两种数据：标题和键值对列表。第一个级别块处理文本字符串 “[Options]”，而第二个级别块处理该部分中所有的键值对。

下表说明了如何打开 [Options] 部分的两个块级别。

标记 [Options] 部分的起始

CML 标记	描述
@1 [;optional;ordered-lines@	数字 1 设置多行块的第一个级别。 [方括号打开一个新块。 optional 表示整个块在配置文件中可选。 ordered-lines 表示此标记 (字符串 [Options]) 之后的任何内容必须在本地 UrlScan.ini 配置文件中首先出现。换句话说，标题 [Options] 必须显示在实际选项之前。
[Options]	用于在本地配置文件中命名此部分的字符串。此字符串将显示在此配置文件中。
@2[;unordered-lines@	数字 2 设置此块的第二个级别。 [方括号打开一个新块，在这种情况中，级别 2 块嵌套于前一个级别 1 块。 unordered lines 表示块中 [Options] 后的行可以在配置文件中以任何顺序排列。即，[Options] 部分中的所有键值对可以按任何顺序排列。

下一步将定义可显示在配置文件的 [Options] 部分中的所有选项。大部分条目都使用 CML 替换标记，因为它们是允许您替换单个值的简单键值对。下表说明了针对每个选项的 CML。

表 10：从 UrlScan.ini [Options] 部分标记键值对

CML 标记	描述
AllowDotInPath = @allow_dot_in_path;boolean@	注意：所有键值对均使用以下语法的某些变体(除非另有说明)： string literal = @source;type@ 此字符串文本定义配置文件中将显示的实际选项名称。源是数据库值集中将用于存储值的位置。类型是将存储

表 10：从 UrlScan.ini [Options] 部分标记键值对(续)

CML 标记	描述
	在值集中的数据类型。 @allow_dot_in_path 此字符串定义用于存储此值的命名空间路径。在本例中，此命名空间是相对的，这意味着它将附加到在模板标题中定义的命名空间 (@!namespace=/security/@)，并且将在该命名空间位置中存储此值。即，此值将存储在数据库的 /security/allow_dot_in_path 键中。 您还可以按以下方式编写此标记： AllowDotInPath = @/security/allow_dot_in_path;boolean@ boolean 由于键值对类型为 Boolean，因此将使用 CML 类型 boolean。请注意，由于此模板的标题将布尔 true 值定义为 1，因此当 IIS 管理员设置此值集时，需要输入 1 以允许在 IIS 服务器的路径中使用点。
AllowHighBitCharacters = @allow_high_bit_characters;boolean@	与上一个示例相似，AllowHighBitCharacters 是显示在配置文件中的选项，allow_high_bit_characters 是相对命名空间路径，boolean 是数据类型。 此 IIS 选项允许用户选择 URL 中是否接受高位字符，在配置文件中使用 1 表示 true，0 表示 false。
AllowLateScanning = @allow_late_scanning;boolean@	允许 IIS 管理员选择是否接受对 URL 进行稍后扫描。定义用于存储此值的命名空间位置。boolean 表示在配置文件中此键可以是 1 用于表示 true，0 表示 false。
AlternateServerName = @alternate_servername@	定义在用户输入或从配置文件读取备用服务器名时可存储此名称的命名空间位置。由于未指定任何类型，因此默认为字符串类型。
EnableLogging = @enable_logging;boolean@	允许用户启用记录，在配置文件中使用 1 表示 true，0 表示 false。
LoggingDirectory = @logging_directory;dir@	在已启用记录之后，允许用户选择用于存储日志文件的目录。类型 dir 表示目录。
LogLongURLs = @log_long_urls;boolean@	允许用户选择是否记录访问服务器的 URL，在配置文件中使用 1 表示 true，0 表示 false。
NormalizeUrlBeforeScan = @normalize_url_before_scan;boolean@	允许用户选择是否在服务器读取 URL 之前标准化此 URL，在配置文件中使用 1 表示 true，0 表示 false。
PerDayLogging = @per_day_logging;boolean@	允许用户选择启用每日记录，在配置文件中使用 1 表示 true，0 表示 false。

表 10：从 UrlScan.ini [Options] 部分标记键值对(续)

CML 标记	描述
PerProcessLogging = @per_process_logging;boolean@	允许用户启用或禁用每个进程的记录，在配置文件中使用 1 表示 true，0 表示 false。

RejectResponseUrl =
 	
@reject_response_url;string;r”(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”;optional@
	语法：
	
	string literal = @source;type;r”regular expression”;option@
	
	`reject response`
	一个字符串文本，用于定义命名空间中将存储字符串的路径。
	
	`string`
	表示拒绝 URL 请求的数据类型为字符串。
	
	`r”`
	这是一个字符串范围说明符，用于引入正则表达式。在这种情况中，它是字符串文本的范围。
	
	`(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”`
	由分析器读取的字符串文本 (已拒绝的 URL 响应): 状态标题、原始动词和原始 URL。
	
	`optional`
	表示值为可选值。即，UrlScan.ini 文件中可忽略 RejectResponseUrl 选项。

RemoveServerHeader = @remove_server_header;boolean@	允许用户启用或禁用 RemoveServerHeading 功能。激活(设置为 1)后，发送到客户端的拒绝响应将删除消息中的服务器标题。此设置在配置文件中使用 1 表示 true，0 表示 false。
UseAllowVerbs = @use_allow_verbs;boolean@	允许用户启用或禁用 UseAllowVerbs 功能。激活(设置为 1)后，对于包含未明确列于 UrlScan.ini 文件 AllowVerbs 部分中的 HTTP 动词的任何请求，服务器将拒绝发送到此服务器的这类请求。在配置文件中使用 1 表示 true，0 表示 false。
UseAllowExtensions = @use_allow_extensions;boolean@	允许用户启用或禁用 UseAllowExtension 功能。激活(设置为 1)后，对于包含未明确列于 UrlScan.ini 文件 AllowExtension 部分中的文件扩展名的任何请求，服务器将拒绝发送到此服务器的这类请求。在配置文件中使用 1 表示 true，0 表示 false。
UseFastPathReject = @use_fast_path_reject;boolean@	允许用户启用或禁用 UseFastPathReject 功能。激活(设置为 1)后，服务器将忽略 RejectResponseUrl 选项，并在 URL 被拒绝后向客户端返回一个简短的 404 响应。在配置文件中使用 1 表示 true，0 表示 false。

表 10：从 UrlScan.ini [Options] 部分标记键值对(续)

CML 标记	描述
VerifyNormalization = @verify_normalization;boolean@	允许用户启用或禁用标准化由 UrlScan.ini 扫描的所有 URL。激活(设置为 1)后，将在扫描 URL 之前标准化此 URL。在配置文件中使用 1 表示 true，0 表示 false。

5.定义 [AllowExtensions] 部分 - 通过打开新块关闭块

既然已定义 UrlScan.ini 文件的 [Options] 部分中的所有选项，那么您已准备好开始定义下一个部分 [AllowExtensions]。请记住，开始 [Options] 部分之前必须打开两个级别的块以说明两个级别的信息 — [Options] 部分的标题及其内容。

在开始定义 [AllowExtensions] 部分之前，需要通过关闭 CML 块来关闭先前的部分。借助 CML，您可以明确关闭带有"]"标记的块，或者也可以通过在较高级别(使用较低的数字指定)或同等级别打开新块来关闭此块。在本任务中，将采用打开 [Options] 部分的块的相同方式打开 [AllowExtensions] 的新块，方法是启动新的级别 1 块。此操作将自动关闭由 [Options] 部分打开的块。

要打开新块并定义 [AllowExtensions] 部分，请执行以下操作：

1. 在 [Options] 部分的最后一行后输入以下内容，打开 [AllowExtensions] 部分的新块：

```
@1[;optional;ordered-lines@
[AllowExtensions]
@2[;unordered-lines@
```

下表说明了如何打开新的两个级别块来关闭先前的块。

启动 [AllowExtensions] 部分的新块

CML 标记	描述
@1 [;optional;ordered-lines@	数字 1 打开新的级别 1 块。由于它是级别 1 块，级别比先前块更高([Options] 部分中键值对的级别 2 块)，与之前的级别 1 块相同，因此它将关闭之前出现的两个块。 请注意，可通过使用关闭块标记明确关闭块。例如： @2]@

启动 [AllowExtensions] 部分的新块(续)

CML 标记	描述
	[用于打开新块的 CML 块标记。 optional 表示整个块为可选块，不必一定出现在配置文件中。 ordered-lines 示此标记 (字符串 [AllowExtensions]) 之后的任何内容必须在本地 UrlScan.ini 配置文件中首先出现。换句话说，您不能首先列出本地文件中的所有选项，然后列出标题。[AllowExtensions] 必须首先出现。在 CML 中，ordered-line 元素确定此顺序。
[AllowExtensions]	用于在本地配置文件中命名此部分的文本字符串。
@2[;unordered-lines@	数字 2 设置此块的第二个级别。 [用于打开新块的 CML 块符号。 unordered lines 表示块中 [AllowExtensions] 后的所有行可以在配置文件中以任何顺序排列。即，[AllowExtensions] 部分中的所有键值对可以按任何顺序排列。

2. 接下来，由于 UrlScan.ini 文件的 [AllowExtensions] 部分可以包含用户输入的任何文件扩展名列表，因此要使用 CML 循环和循环目标标记，指示分析器每次读取一行此部分中的信息。在最后一个步骤中的最后文本 @2[;unordered-lines@ 之后，立即输入以下文本：

```
@*allow_extension;unordered-string-set@
.@.@
```

下表说明了循环和循环目标 CML 标记的工作方式:

循环和循环目标 CML 标记

CML 标记	描述
@*allow_ extension;unordered-string-set@	语法 @<level><tag type><name>;<data type>;<options>@ 循环标记 (*) 将循环并读取 [AllowExtensions] 部分中列出的无序字符串集。 allow_extension 一个字符串，用于定义命名空间中将存储字符串的路径。

循环和循环目标 CML 标记(续)

CML 标记	描述
	unordered-string-set 表示字符串列表不必以任何特定顺序排列。
.@.@	第一个 (.) 在此部分中，分析器读取的无序字符串集是 [AllowExtensions] 部分中列出的以 (.) 字符开头的文件扩展名的列表。 @.@ 循环目标标记 (.) 指示分析器读取此列表中以句点字符开头的任何内容。

3. 保存此文件。

6.定义 [DenyExtensions] 部分

接下来采用定义 [AllowExtensions] 部分的相同方式定义 UrlScan.ini 文件的 [DenyExtensions] 部分。将打开一个新的级别 1 块，此操作将关闭 [AllowExtensions] 部分中的先前块。然后将打开一个级别 2 块，在此块中，可以指示分析器读取要使用 UrlScan.ini 阻止的以 (.) 开头的所有文件扩展名的无序列表。

[DenyExtensions] 部分的 CML 类似如下：

```
@1[;optional;ordered-lines@
[DenyExtensions]
@2[;unordered-lines@
@*deny_extension;unordered-string-set@
.@.@
```

7.定义 [AllowVerbs] 和 [DenyVerbs] 部分

UrlScan.ini 文件中接下来的两个部分遵循先前部分中用于 [DenyExtensions] 的相同 CML。打开第一个级别块将关闭先前块，还会将以下文本分析为有序行。

然后打开第二个级别块，此块可读取以下无序字符串列表 —即动词列表。在这两个部分中，此字符串将指示分析器读取被允许访问网站的动词列表以及被拒绝访问网站的动词列表。

这两个部分的 CML 如下所示：

```
@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
@*allow_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyVerbs]
@2[;unordered-lines@
@*deny_verb;unordered-string-set@
@.@
```

8.定义 [DenyHeaders] 部分

接下来需定义 UrlScan.ini 文件的 [DenyHeaders] 部分，此部分允许您将 IIS 配置为拒绝特定 HTTP 请求标题。

此部分与先前部分相似，同样需要为字符串打开两个块。但是，您要使用 CML 序列分隔符通过冒号来分隔 UrlScan.ini 文件中列出的 HTTP 标题列表。由于 HTTP 请求标题包含冒号 (:)，您需要使用序列分隔符指示分析器读取此部分中的每一行，以便在遇到冒号 (:) 时，分析器将移动到下一个条目。

例如，UrlScan.ini 文件中列出的要拒绝的 HTTP 标题列表可能读为：

Translate:
If:
Lock-Token:

由于配置文件中列出的每个标题请求都以 (:) 结尾，因此需要指示分析器将 (:) 识别为条目的结尾。

1. 要定义 [DenyHeaders] 部分，请在 [DenyVerbs] 部分的最后一行后输入以下文本，打开 [DenyHeaders] 部分的新块：

   @1[;optional;ordered-lines@
   [DenyHeaders]
   @2[;unordered-lines@

   在先前部分中，这些标记打开要读取为有序行的级别 1 块，然后打开要读取为无序行的第二级别块。

2. 接下来输入以下 CML 循环和循环目标标记以指示分析器读取标题请求的列表：

   ```
   @*deny_header;unordered-string-set;;sequence-delimiter=":"@
   @.@:
   ```

   下表描述了这两个标记的语法。

   [DenyHeaders] 部分的循环和循环目标标记

CML 标记	描述
@*deny_header;unordered-string-set;;sequence-delimiter=":"@	* 表示将读取字符串列表的循环 CML 标记。 deny_header 一个字符串文本，用于定义命名空间中将存储字符串的路径。 unordered-string-set 指示字符串列表可以按任何顺序列出。 ; 第一个分号分隔此标记的两个部分。 ; 第二个分号允许您输入以下冒号 (:) 序列分隔符，而无需将其解释为范围。 sequence-delimiter=":" 指示分析器将冒号 (:) 读取为字符串的一部分，并视其为移动到下一个条目的点。
@.@	循环目标标记指示分析器将这些值存储到 deny_header 命名空间位置中。例如：/security/deny_header。
:	最后一个冒号 (:) 向分析器指明此列表中的每个项将后跟一个冒号。即，会将此字符存储为已拒绝标题的条目的一部分。

3. 保存此文件。

9.定义 [DenyURLSequences] 部分

使用定义 [DenyHeader] 部分的相似方式定义 [DenyUrlSequence]。打开将为有序和无序字符串读取的两个块。但是，对于此部分，需使用字段分隔符来分隔模板中 URL 序列的列表。在此处使用的字段分隔符将是一个行结尾元素，可指示分析器在遇到行结尾时停止读取条目。

要定义 [DenyUrlSequence] 部分，请执行以下操作：

1. 在 [DenyHeaders] 部分的最后一行后输入以下文本，打开 [DenyUrlSequence] 部分的新块：

   ```
   @1[;optional;ordered-lines@
   [DenyUrlSequence]
   @2[;unordered-lines@
   ```

   在先前部分中，这些标记打开要读取为有序行的级别 1 块，然后打开要读取为无序行的第二级别块。

2. 接下来键入以下 CML 循环和循环目标标记，以指示分析器读取要拒绝的 URL 序列的列表：

   ```
   @*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@
   @.@
   ```

   下表描述了这些标记的语法。

   [DenyUrlSequence] 部分的循环和循环目标标记

CML 标记	描述
@*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@	* 表示可读取字符串列表的循环 CML 标记。 deny_url_sequence 一个字符串文本，用于定义命名空间中将存储字符串的路径。 unordered-string-set 指示字符串列表可以按任何顺序列出。 ; 第一个分号分隔此标记的两个部分。 ; 第二个分号允许您输入随后的字段分隔符，而无需将其解释为范围。 field-delimiter-is-eol 指示分析器读取下一个条目，直到行结尾。
@.@	循环目标标记指示分析器将这些值存储到 deny_url_sequence 命名空间位置中。例如：/security/deny_url_sequence。

3. 保存此文件。

10.定义 [RequestLimits] 部分

定义 [RequestsLimits] 的方式与定义 [DenyUrlSequence] 部分的方式极为相似。打开将为有序和无序字符串读取的两个块。但是对于此部分，在打开两个块之后，将使用 CML 替换标记定义三个键值对。

要定义 [RequestsLimits] 部分，请执行以下操作：

1. 在 [DenyUrlSequence] 部分的最后一行后输入以下文本，打开 [RequestsLimits] 部分的新块：

   ```
   @1[;optional;ordered-lines@
   [RequestsLimits]
   @2[;unordered-lines@
   ```

   在先前部分中，这些标记打开要读取为有序行的级别 1 块，然后打开要读取为无序行的第二级别块。通过启动新的第一级别块进行撤消，将关闭先前 [DenyUrlSequence] 部分中的第二级别块。

2. 接下来，键入以下 `CML replace` 标记以定义在 [RequestsLimits] 部分找到的三个键值对。

   ```
   MaxAllowedContentLength = @max_allowed_content_length;int@
   MaxUrl = @max_url;int@
   MaxQueryString = @max_query_string;int@
   @1]@
   ```

   下表描述了这些标记的语法。

   [DenyUrlSequence] 部分的循环和循环目标标记

CML 标记	描述

MaxAllowedContentLength = @max_allowed_content_length;int@	`MaxAllowedContentLength`
	配置文件中的请求限制参数字符串。
	`max_allowed_content_length`
	一个字符串文本，用于定义命名空间中将存储值的路径。
	`int`
	表示要存储的值为整数。

MaxUrl = @max_url;int@	`MaxUrl`

[DenyUrlSequence] 部分的循环和循环目标标记(续)

CML 标记	描述

	配置文件中的请求限制参数字符串。
	`max_url`
	一个字符串文本，用于定义命名空间中将存储值的路径。
	`int`
	表示要存储的值为整数。

MaxQueryString = @max_query_string;int@	`MaxQueryString`
	配置文件中的请求限制参数字符串。
	`max_query_string`
	一个字符串文本，用于定义命名空间中将存储值的路径。
	`int`
	表示要存储的值为整数。

@1]@	此级别 1 块标记可关闭块。

3. 保存此文件。

11.将模板置于应用程序配置中

在为 UrlScan.ini 创建 CML 模板并将其另存为 url_scan_ini.tpl 之后，您可以准备好执行以下任务:

- 将模板导入 SA 客户端中并验证 CML 语法。请参见 导入和验证模板文件 (第 263 页)。
- 将模板添加到应用程序配置。请参见 将模板添加到应用程序配置或从应用程序配置删除模板 (第 264 页)。
- 将应用程序配置附加到服务器。请参见 将应用程序配置附加到服务器或设备组 (第 268 页)。
- 通过对模板进行更改并将更改推送到服务器，验证模板。请参见 推送应用程序配置 (第 272 页)。

CML 教程 1 - 为简单 Web 应用程序服务器创建应用程序配置 (第 306 页)中描述了这些步骤。

UrlScan.ini 文件示例

以下是 UrlScan.ini 文件示例。

```
[Options]
UseAllowVerbs=1                ; If 1, use [AllowVerbs] section, else use the
                               ; [DenyVerbs] section.   The default is 1.

UseAllowExtensions=0           ; If 1, use [AllowExtensions] section, else
                               ; use the [DenyExtensions] section.The
                               ; default is 0.

NormalizeUrlBeforeScan=1       ; If 1, canonicalize URL before processing.
                               ; The default is 1.Note that setting this
                               ; to 0 will make checks based on extensions,
                               ; and the URL unreliable and is therefore not
                               ; recommend other than for testing.

VerifyNormalization=1          ; If 1, canonicalize URL twice and reject
                               ; request if a change occurs.The default
                               ; is 1.

AllowHighBitCharacters=0       ; If 1, allow high bit (ie.UTF8 or MBCS)
                               ; characters in URL.The default is 0.

AllowDotInPath=0               ; If 1, allow dots that are not file
                               ; extensions.The default is 0.Note that
                               ; setting this property to 1 will make checks
                               ; based on extensions unreliable and is
                               ; therefore not recommended other than for
                               ; testing.
```

```
RemoveServerHeader=1          ; If 1, remove the 'Server' header from
                              ; response.The default is 0.

EnableLogging=1               ; If 1, log UrlScan activity.The
                              ; default is 1.Changes to this property
                              ; will not take effect until UrlScan is
                              ; restarted.

PerProcessLogging=0           ; This property is deprecated for UrlScan
                              ; 3.0 and later.UrlScan 3.0 and later can
                              ; safely log output from multiple processes
                              ; to the same log file.Changes to this
                              ; property will not take effect until
                              ; UrlScan is restarted.

AllowLateScanning=0           ; If 1, then UrlScan will load as a low
                              ; priority filter.The default is 0.Note
                              ; that this setting should only be used in
                              ; the case where there another installed
                              ; filter is modifying the URL and you wish
                              ; to have UrlScan apply its rules to the
                              ; rewritten URL.Changes to this property
                              ; will not take effect until UrlScan is
                              ; restarted.

PerDayLogging=1               ; If 1, UrlScan will produce a new log each
                              ; day with activity in the form
                              ; 'UrlScan.010101.log'.If 0, UrlScan will
                              ; log activity to urlscan.log.The default
                              ; is 1.Changes to this setting will not
```

```
                                ; take effect until UrlScan is restarted.

UseFastPathReject=0             ; If 1, then UrlScan will not use the
                                ; RejectResponseUrl.On IIS versions less
                                ; than 6.0, this will also prevent IIS
                                ; from writing rejected requests to the
                                ; W3SVC log.UrlScan will log rejected
                                ; requests regardless of this setting.The
                                ; default is 0.

LogLongUrls=0                   ; This property is deprecated for UrlScan 3.0
                                ; and later.UrlScan 3.0 and later will
                                ; always include the complete URL in its log
                                ; file.

UnescapeQueryString=1           ; If 1, UrlScan will perform two passes on
                                ; each query string scan, once with the raw
                                ; query string and once after unescaping it.
                                ; If 0, UrlScan will only look at the raw
                                ; query string as sent by the client.The
                                ; default is 1.Note that if this property is
                                ; set to 0, then checks based on the query
                                ; string will be unreliable.

RejectResponseUrl=

LoggingDirectory=Logs

[AllowVerbs]

;
```

```
; The verbs (aka HTTP methods) listed here are those commonly
; processed by a typical IIS server.
;
; Note that these entries are effective if "UseAllowVerbs=1"
; is set in the [Options] section above.
;

GET
HEAD
POST

[DenyVerbs]

;
; The verbs (aka HTTP methods) listed here are used for publishing
; content to an IIS server via WebDAV.
;
; Note that these entries are effective if "UseAllowVerbs=0"
; is set in the [Options] section above.
;

PROPFIND
PROPPATCH
MKCOL
DELETE
PUT
COPY
MOVE
LOCK
UNLOCK
OPTIONS
```

```
SEARCH

[DenyHeaders]

;
; The following request headers alter processing of a
; request by causing the server to process the request
; as if it were intended to be a WebDAV request, instead
; of a request to retrieve a resource.
;

Translate:
If:
Lock-Token:
Transfer-Encoding:

[AllowExtensions]

;
; Extensions listed here are commonly used on a typical IIS server.
;
; Note that these entries are effective if "UseAllowExtensions=1"
; is set in the [Options] section above.
;

.htm
.html
.txt
.png
.png
.png
```

```
[DenyExtensions]

;
; Extensions listed here either run code directly on the server,
; are processed as scripts, or are static files that are
; generally not intended to be served out.
;
; Note that these entries are effective if "UseAllowExtensions=0"
; is set in the [Options] section above.
;
; Also note that ASP scripts are denied with the below
; settings.If you wish to enable ASP, remove the
; following extensions from this list:
;    .asp
;    .cer
;    .cdx
;    .asa
;

; Deny executables that could run on the server
.exe
.bat
.cmd
.com

; Deny infrequently used scripts
.htw     ; Maps to webhits.dll, part of Index Server
.ida     ; Maps to idq.dll, part of Index Server
.idq     ; Maps to idq.dll, part of Index Server
.htr     ; Maps to ism.dll, a legacy administrative tool
```

```
.idc     ; Maps to httpodbc.dll, a legacy database access tool
.shtm    ; Maps to ssinc.dll, for Server Side Includes
.shtml   ; Maps to ssinc.dll, for Server Side Includes
.stm     ; Maps to ssinc.dll, for Server Side Includes
.printer ; Maps to msw3prt.dll, for Internet Printing Services

; Deny various static files
.ini     ; Configuration files
.log     ; Log files
.pol     ; Policy files
.dat     ; Configuration files
.config  ; Configuration files

[DenyUrlSequences]
;
; If any character sequences listed here appear in the URL for
; any request, that request will be rejected.
;

..; Don't allow directory traversals
./; Don't allow trailing dot on a directory name
\   ; Don't allow backslashes in URL
:   ; Don't allow alternate stream access
%   ; Don't allow escaping after normalization
&   ; Don't allow multiple CGI processes to run on a single request
```

完整 url_scan_ini.tpl CML 模板

以下是完整的 url_Scan_ini.tpl 模板。

```
@#############################################
# \system32\inetsrv\urlscan.ini (Windows)    #
# Version 1.0                                #
# Joe Author (joe_author@your_company.com)   #
#############################################@

@!namespace=/security/@
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@

@#########################################
# Begin data                             #
#########################################@

@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
AllowDotInPath = @allow_dot_in_path;boolean@
AllowHighBitCharacters = @allow_high_bit_characters;boolean@
AllowLateScanning = @allow_late_scanning;boolean@
AlternateServerName = @alternate_servername@
EnableLogging = @enable_logging;boolean@
LoggingDirectory = @logging_directory;dir@
LogLongURLs = @log_long_urls;boolean@
NormalizeUrlBeforeScan = @normalize_url_before_scan;boolean@
PerDayLogging = @per_day_logging;boolean@
PerProcessLogging = @per_process_logging;boolean@
RejectResponseUrl =
@reject_response_url;string;r”(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN
_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”;optional@
RemoveServerHeader = @remove_server_header;boolean@
UnescapeQueryString = @unescape_query_string;boolean@
UseAllowVerbs = @use_allow_verbs;boolean@
UseAllowExtensions = @use_allow_extensions;boolean@
UseFastPathReject = @use_fast_path_reject;boolean@
VerifyNormalization = @verify_normalization;boolean@

@1[;optional;ordered-lines@
[AllowExtensions]
@2[;unordered-lines@
@*allow_extension;unordered-string-set@
.@.@

@1[;optional;ordered-lines@
[DenyExtensions]
@2[;unordered-lines@
```

```
@*deny_extension;unordered-string-set@
.@.@

@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
@*allow_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyVerbs]
@2[;unordered-lines@
@*deny_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyHeaders]
@2[;unordered-lines@
@*deny_header;unordered-string-set;;sequence-delimiter=":"@
@.@:

@1[;optional;ordered-lines@
[DenyURLSequences]
@2[;unordered-lines@
@*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@
@.@

@1[;optional;ordered-lines@
[RequestLimits]
@2[;unordered-lines@
MaxAllowedContentLength = @max_allowed_content_length;int@
MaxUrl = @max_url;int@
MaxQueryString = @max_query_string;int@
@1]@
```

CML 基础知识

本章将介绍“配置建模语言”(CML)。有关 CML 的完整详细信息，请参见CML 引用 (第 352 页)。另请参见CML 教程 1 - 为简单 Web 应用程序服务器创建应用程序配置 (第 306 页)和CML 教程 2 - 创建 Web 服务器配置文件的模板 (第 316 页)。

SA 通过创建**配置模板**来管理配置文件，此模板可用于：

- 对配置文件的语法进行建模。
- 从配置文件提取值并将这些值存储为 SA 数据库中的**值集**。存储这些值之后，可以使用 SA 客户端管理它们。
- 从值集创建新的配置文件。
- 将新的配置文件推送到服务器。
- 审核服务器上的配置文件，确保符合要求。

术语

- **配置文件** - 由 SA 管理的文件。
- **值集** - 配置文件中随服务器变化的数据值。值集中的值以“键 = 值”的格式存储在 SA 数据库中。
- **命名空间** - 在 SA 数据库中存储值集的结构。
- **配置模板** - 使用 CML 编写的配置文件模型。
- **配置建模语言 (CML)** - 用于在配置模板中对配置文件进行建模的一组指令标记。
- **指令**或**标记** - 定义要执行的操作的关键字和字符。所有以“@”字符开头和结尾的指令。术语“指令”和“标记”可以交换使用。
- **应用程序配置对象** - 配置模板的容器，当配置模板与值集组合时可生成配置文件，然后会将配置文件“推送”到托管服务器。此外，此对象还包含作为推送操作的一部分执行的脚本。

CML 基本概念

CML 配置建模语言可对配置文件的语法进行建模。可使用 CML 创建**配置模板**，此模板是目标配置文件的模型。要执行此操作，通常最好的方法是获取目标配置文件的文档，以便了解此配置文件中的有效值和范围以及确定建模的最佳方式。

采用 CML 对整个配置文件进行建模时，配置模板运行得最好。但是，有可能只需采用 CML 编写配置文件中的某些行。这称为部分模板，将在部分模板 (第 351 页)中对其进行讨论。

必需的 CML 指令标记

在任何配置模板中均需要以下三种 CML 指令(也称为 CML 标记)。

- namespace 定义 SA 数据库中存储值集数据的键。请参见 Namespace 标记 (第 341 页)。
- filename-key 定义 SA 数据库中存储目标配置文件名的键。请参见 Filename-key 标记 (第 342 页)。
- filename-default 指定目标配置文件的默认名称。请参见 Filename-default 标记 (第 342 页)。

所有其他标记可供选择，用于对特定配置文件的内容进行建模。有关 CML 的完整详细信息，请参见CML 引用 (第 352 页)。有关这三种必需标记的详细信息，请参见CML 全局选项特性 (第 376 页)。

Namespace 标记

namespace 指令定义 SA 数据库中存储值集的键。

语法：

```
@!namespace=<path>@
```

其中，<path> 是与目录路径相似的字符串，用于定义 SA 数据库中存储值集的键。

示例：

```
@!namespace=/example/namespace/@
```

此示例将此模板中所有其他指令的基本命名空间设置为 “/example/namespace/”。即除非另有说明，否则值集中的所有值都将存储在 “/example/namespace/”键中。

描述:

namespace 指令为值集中的值指定键/值映射中的键。它是任意字符串，可以是除数字以外的任何内容。此指令可确定 SA 数据库中存储值集中值的键。配置文件中的“替换”标记(和其他标记)使用此命名空间键从 SA 数据库获取值。

namespace 值必须是绝对值。后续值可以具有相对或绝对路径名。

- 绝对名称是以“/”字符开头的完整路径名。这些名称不会使用 namespace 指令中指定的值。例如，任何与以下标记匹配的值：

  ```
  @/testval@
  ```

  将会存储在“/testval”键下的值集中。

- 相对名称会附加到 namespace 指令标记中指定的值。例如，任何与以下标记匹配的值：

```
@testval@
```

将会存储在“/example/namespace/testval”键下的值集中。

请注意，任何属于循环的命名标记都需要在名称前面添加一个点“.”，并且该标记命名空间会附加到当前循环的命名空间。例如：`@.testval@`

Filename-key 标记

filename-key 指令指定 SA 数据库中存储目标配置文件名的键。

语法：

```
@!filename-key=<path>@’
```

其中，<path> 是与目录路径相似的任意字符串，用于定义 SA 数据库中存储配置文件名的键。

示例：

```
@!filename-key=/files/example@
```

此示例指定文件名将存储在 SA 数据库中的“/files/example”键下。请注意，由于 <path> 值以“/”字符开头，因此它为一个绝对路径。

```
@!filename-key=files/example@
```

此示例指定相对路径，因为值不以“/”字符开头。如果将其与前一个 namespace 示例组合，则配置文件名将存储在“/example/namespace/files/example”中。

描述:

filename-key 指令指定 SA 数据库中将用于存储目标配置文件名的键。例如，在 SA 客户端中设置模板的“文件名”字段时，该文件名将存储在值集的“/files/example”键下。请注意，filename-key 不是一个文件系统路径，而仅仅是与路径写法相似的一个键。

对于需要在被推送到目标服务器之前或之后了解配置文件名的 Pre 和 Post 脚本，这将十分方便。例如，如果您的 Post 脚本需要在被推送之后向配置文件的末尾添加一行，则可能类似如下：

```
echo “#end of the file” >> @/files/example@
```

Filename-default 标记

语法：

```
@!filename-default=<file>@
```

其中，<file>为托管服务器的文件系统中目标配置文件的目录路径和文件名。这是所生成配置文件将被推送到的托管服务器上的目标位置。

示例：

```
@!filename-default=/etc/hosts@
```

此示例指定目标配置文件为 /etc/hosts。此值存储在 SA 数据库中由 Filename-key 指令指定的键下。

描述:

filename-default 指令指定目标配置文件的标准文件系统路径。此值为默认的文件名和目录，存储在由 filename-key 指令指定的键下。它是 SA 客户端中“文件名”字段的默认值。

将标记组合在一行

可以通过使用分号分隔指令并仅以一个感叹号开头，将多个指令标记组合到一个指令中，如下所示：

```
@!namespace=/example/namespace/;filename-key=/files/example;
filename-default=/etc/example@
```

这将十分方便，因为任何文件中包含此行都将使 CML 模板变得有效，当然，我们假设所有其他 CML 标记具有正确的格式。

用例 1 - 简单键=值配置文件

最简单的配置文件类型具有一个或多个键=值条目，如下例所示：

```
Port = 1280
IPAddress = 192.168.0.1
ServerName = server01
```

在此配置文件中，类型和描述很容易理解。

使用替换指令

要编写此配置文件的模板，请使用 CML 标记来表示此值存在的位置以及将存储在值集命名空间中的位置。用于此目的的标记为替换标记。

此替换标记由多个字段组成(包括 CML 语言的所有标记)，它以“@”符号开头和结尾并使用分号分隔字段。格式类似如下：

```
@ <name> ; [type] ; [range] ; [option] ; [option] … @
```

在所有的字段中，只有 <name> 字段为必填字段。因此，可表示以上配置文件的最基本的 CML 将为：

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port@
IPAddress = @ipaddress@
ServerName = @servername@
```

这指定端口号的值将存储在 SA 数据库的“/example/namespace/port”键中。IP 地址将存储在“/example/namespace/ipaddress”键中，服务器名将存储在“/example/namespace/servername”键中。

这在技术上可行，但是可能会缺失大量可用的字段验证和错误检查，例如，验证和检查可防止输入无效的数据(如为端口输入“someport”)。

替换指令标记中的 <name> 字段

如果 <name> 字段为相对字段 (即不以“/”或“.”开头)，则它会附加到当前的命名空间并且成为键 (用于存储由此标记从 SA 数据库读取的值) 的一部分。

如果此名称为绝对名称(即以“/”开头)，则它是整个键并且值将存储在此键下。

最后，如果此名称以点“.”开头，则它将附加到其所属的任意循环的命名空间。除极少数异常外，循环中的每个标记均应以点“.”开头。

替换指令标记中的 <type> 字段

<type> 字段允许您根据已知类型将某些预定义的范围和错误检查分配到不同的值。有关类型的完整列表，请参见CML 类型特性 (第 367 页)。对于此配置文件，请针对单独的条目使用预定义类型“port”、“ip”和“hostname”，如下所示：

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port;port@
IPAddress = @ipaddress;ip@
ServerName = @servername;hostname@
```

添加这些类型将会限制值并且提供验证和错误检查。

此外，还可使用替换标记通过前置“ordered-”或“unordered-”并且附加“-set”或“-list”，来表示重复值的序列。下一个示例中包含更多有关此操作的信息。

此字段的默认值为“string”，将与任何内容匹配。

替换指令标记中的 <range> 字段

<range> 字段允许您为值设置允许的范围。可以设置整数范围或字符串范围。整数范围对严格由整数组成的任意类型有效，字符串范围对所有其他类型有效。

可以通过使用逗号“,”将范围与逻辑 OR 组合使用。可以通过使用与字符“&”将范围与逻辑 AND 组合使用。“!”字符可对范围求反。

请记住，在读取配置文件以及接受 SA 客户端中的值时将使用指定的范围。如果配置文件中的某个值在模板设置的范围之外，则在解析该文件时将提示错误。根据配置文件的文档指定有效的范围。

整数范围

整数范围只能使用 < 和 = 符号来指定“小于”或“大于”范围。指定用于比较的数字位置，如下所示：

指定整数范围

范围条件	使用的符号
大于	n<
大于或等于	n<=
小于	<n
小于或等于	<=n
等于	=n

例如，如果配置文件中的端口只能介于 1024(含)和 2048(含)之间，则会将此范围添加到标记，类似如下：

```
Port = @port;port;1024<=&<=2048@
```

字符串范围

字符串范围可以是包含在引号中的有效字符串列表以及以 r" 字符开头并以引号结束的正则表达式列表。例如，如果 ServerName 字段只能是以单词“server”开头的任意内容，则需将此范围添加到 servername 标记，类似如下：

```
ServerName = @servername;hostname;r"server.*”@
```

替换指令标记中的 [option] 字段

可根据需要将任意多个选项附加到此标记。在第三个分号后的所有内容都将被视为选项，并且每个选项以分号分隔。

例如，如果配置文件中的 IPAddress 行是可选行而不是有效配置文件的必需行，并且 IP 地址可在正斜杠处停止，则可添加选项，类似如下：

```
IPAddress = @ipaddress;ip;;optional;delimiter="/"@
```

这将与以下条目匹配：

```
IPAddress = 192.168.0.1
```

此外，它还与以下条目匹配：

```
IPAddress = 192.168.0.2/
```

请注意，将范围字段保留为空。如果要填充任何之后的字段，则必须仍然表示要保留为默认值的任何字段。例如，以下两行有效：

```
@ipaddress@
```

```
@ipaddress;;;optional@
```

但是，以下行则无效，因为字段“optional”将被解释为 <type> 字段，而不是一个选项，从而导致错误。

```
@ipaddress;optional@
```

最终 CML 模板

在设置了所有类型、选项和范围之后，CML 模板应类似如下：

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port;port;1024<=&<=2048@
IPAddress = @ipaddress;ip;;optional;delimiter="/"@
ServerName = @servername;hostname;r"server.*"@
```

生成的值集

使用以上配置模板读取以上示例目标配置文件将生成以下值集，该值集存储在 SA 数据库中：

```
/example/namespace/port = 1280
/example/namespace/ipaddress = 192.168.0.1
/example/namespace/servername = server01
```

正如您所看到的，命名空间中的键是模板的命名空间 (/example/namespace) 和每个标记名称的组合，因为它们都使用相对名称。

用例 2 - 配置文件中的重复值

您可能会遇到一个配置文件，其中只包含一个值列表，例如一个文件中只包含对某个目录具有写入权限的用户名列表。此文件的格式可能类似如下：

```
admin;
user1;
user2;
```

此文件中的重复行不仅仅调用先前示例中描述的替换标记。可以与此配置文件匹配的最基本 CML 为以下循环指令：

```
@!namespace=/wuserlist/namespace/@
@!filename-key=/wuserlistfile/example@
@!filename-default=/etc/wusers.txt@
@*users@
@.@
```

使用循环指令标记

当值集在配置文件中多次出现时，可使用循环指令。循环指令的默认行为是直接在 CML 行之后循环该行，虽然可将其修改为多行循环或在单行中循环。

格式类似如下：

```
@ [group level] * <name> ; [ "ordered" | "unordered" ] - [type] - ["set" | "list" ]
; [range] ; [option] ; [option] … @
```

您会注意到此标记和先前部分中的替换标记之间的某些差异和相似之处。以下部分将描述循环标记上的每个选项。

<group level> 字段

对于此配置文件示例，组级别并不重要，将在之后对其进行讨论。此时只需要了解，组级别用于确定要循环的特定部分，是在单行中循环还是进行多行循环。

对于本示例，将组级别保留为空，表示此循环标记将仅直接遍历其下方的 CML 行。

指令类型说明符字段 *

“*”是一个指令类型说明符。它表示指令类型，在本例中，此类型为循环指令。替换指令是默认的指令类型，由于它比较常用，因此不具有指令类型说明符。

<name> 字段

相同的 <name> 字段规则适用于此标记，也适用于替换标记。要查看此内容，请参见替换指令标记中的 <name> 字段 (第 344 页)。任何属于此循环的命名标记都需要一个“.”(点)位于此名称的前面，并且该标记的命名空间会附加到此循环的命名空间。同样，如果此循环是另一个循环的一部分，则名称的前面需要添加一个“.”。

本示例将使用名称“users”，如下所示：

```
@*users@
```

[type] 字段

循环标记的 [type] 字段与替换标记的 [type] 字段的差异主要存在于两个方面。(要查看此内容，请参见替换指令标记中的 <type> 字段 (第 344 页)。)

- Ordered 与 Unordered 以及 Set 与 List 的对比

  基本类型包括所有与替换标记相同的类型，但是由于会存在重复的值序列，因此需要指定序列的信息。通过前置“ordered”或“unordered”(后跟短划线)并将后跟“set”或“list”的短划线附加到修改的 [type] 字段，可以使此字段包含此信息。
  - 前置“ordered”指定保留值的顺序。
  - 前置“unordered”指定可以按任何顺序排列值。
  - 附加“set”指定值必须唯一。
  - 附加“list”指定值可以重复。

    ordered 或 unordered 选项以及 set 或 list 选项对于替换标记来说是可选的，然而对于循环标记来说是必需的。

    对于本示例，数据是无序排列的，因此应附加“set”，因为没有必要对访问列表排序或者包含重复的值。

- 命名空间类型
  此类型对循环标记唯一。如果要遍历的部分包含多个值，则需使用命名空间类型。

此标记的默认类型为“unordered-string-list”。

对于本示例，此默认值可行，但是最好将类型指定为“user”。假设一个无序集，生成的标记可能类似如下：

```
@*users;unordered-user-set@
```

[range] 字段

除命名空间类型以外，范围与替换标记中的每种类型的范围都相同。由于命名空间类型遍历可能拥有各自范围的多个不同标记，因此不应使用任何范围。

由于本示例使用“user”类型，因此还可以使用一个范围。例如，如果配置文件的文档指明“root”不是有效用户，则可以将范围设置为对除 root 以外的任何对象有效，如下所示：

```
@*users;unordered-user-set;!"root;"@
```

[option] 字段

“选项”字段与替换标记的选项字段相同。要查看此内容，请参见替换指令标记中的 [option] 字段 (第 345 页)。

对于本示例，通过将分号设置为字段分隔符并使其包含于遍历的行中来消除读取的值中的分号，可以从用户名中消除“;”，如下所示：

```
@*users;unordered-user-set;!"root";field-delimiter-is-semicolon@
@.@;
```

循环目标标记

循环目标标记类似如下：

```
@.@
```

它的唯一目的是表示循环值的位置，即在生成的配置文件中放置数据的位置。

最终 CML

在设置了所有类型、选项和范围之后，CML 模板应类似如下：

```
@!namespace=/wuserlist/namespace/@
@!filename-key=/wuserlistfile/example@
@!filename-default=/etc/wusers.txt@
@*users;unordered-user-set;!"root";field-delimiter-is-semicolon@
@.@;
```

生成的值集

读取的每个值需要存储在唯一键下的值集中。要处理序列，CML 会将唯一编号附加到命名空间，此编号从 1 开始，随每个附加迭代递增。

使用 CML 的以上示例配置文件生成的值集类似如下：

```
/example/namespace/users/1 = admin
/example/namespace/users/2 = user1
/example/namespace/users/3 = user2
```

用例 3 - 配置文件中的复杂重复值

此示例对 /etc/hosts 文件建模，此文件是后跟主机名列表的 IP 地址列表，类似如下：

```
127.0.0.1 localhost
192.168.0.1 server1 server1.domain.com
192.168.0.2 server2 server2.domain.com
```

本示例与上一个示例相似，除了它遍历更加复杂的行。最佳方式是首先使用 CML 替换指令对行的单个实例进行建模，类似如下：

```
@ip-addr;ip@ @sname;unordered-hostname-set@
```

此行定义两个替换指令，一个针对 IP 地址，另一个针对服务器名。

请注意，“ip-addr”替换标记将数据类型指定为“ip”，因为它必须是一个 IP 地址。

“sname”替换标记的类型被指定为“unordered-hostname-set”。这意味着它可以与主机名列表匹配并且会将它们与对应的 IP 地址一同存储。这与循环标记的工作方式相似，并且以相同的方式存储值。

此 CML 针对一个迭代。下一步骤会将此迭代包含在一个循环中。要实现这一点，请使用命名空间循环，因为此循环将遍历每行中的多个值并且在循环中将“.”前置到标记的名称，如下所示：

```
@*entries;unordered-namespace-set@
@.ip-addr;ip@ @.sname;unordered-hostname-set@
```

循环标记(由 @* 字符表示)可定义循环。“unordered-hostname-set”指示数据为主机名、主机名可以按任何顺序排列并且值必须唯一。

在替换指令中的“ip-addr”和“sname”字符串前添加的“.”字符指示这些字符串是循环指令的目标。

最终 CML

将以上循环和替换 CML 添加到所需的命名空间和文件行将生成以下结果。

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/hosts@
@*entries;unordered-namespace-set@
@.ip-addr;ip@ @sname;unordered-hostname-set@
```

生成的值集

如果使用 SA 客户端读取示例文件中的条目，则以下值将存储在 SA 数据库中。

```
/example/namespace/entries1/ip-addr = 127.0.0.1
/example/namespace/entries1/sname/1 = localhost
/example/namespace/entries2/ip-addr = 192.168.0.1
/example/namespace/entries2/sname/1 = server1
/example/namespace/entries2/sname/2 = server1.domain.com
/example/namespace/entries3/ip-addr = 192.168.0.2
/example/namespace/entries3/sname/1 = server2
/example/namespace/entries3/sname/2 = server2.domain.com
```

部分模板

虽然最好的方式是对整个配置文件进行建模，但是您也可以使用部分模板只对配置文件的一部分进行建模。要创建部分模板，必须在服务器上为要读取的模板创建完整配置文件的备份。并且必须使用“保留格式”选项来保留文件的其他部分。

以下显示了一个简单配置文件。

```
UserName = alice
Password = pass
HomeDir = /home/alice
```

要仅管理主目录行，请使用 @!partial-template 指令并仅对要管理的行进行建模。此模板将类似如下：

```
@!namespace=/example/@
@!filename-key=/files/example@
@!filename-default=/usr/example@
@!partial-template@
HomeDir = @homedir;dir@
```

有关“保留格式”设置的详细信息，请参见在值集编辑器中设置值 (第 224 页)。另请参见@!full-template 和 @!partial-template 特性 (第 377 页)。

CML 引用

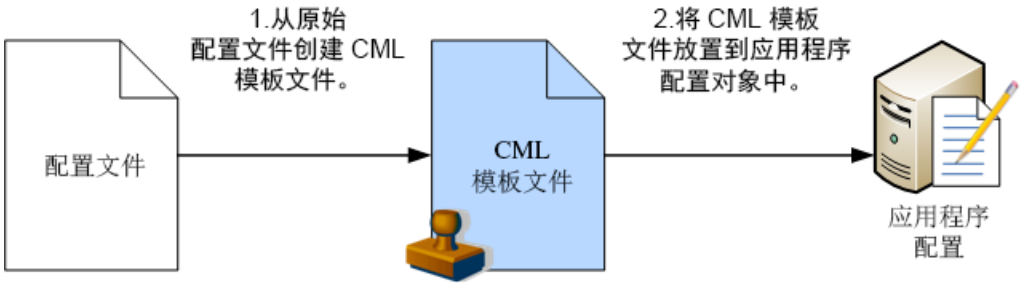
配置建模语言 (CML) 用于创建**配置文件**的**模板**，以便从 SA 管理此模板。CML 模板是所创建的独立文件，可对配置文件的格式进行建模，以便针对不同服务器集将配置文件中的变量值设置为不同值。

模板文件包含数据、指令和定义，以便可从模板和值集生成实际的配置文件。

CML 定义双向转换：它指定如何从配置文件将值移动到 SA 数据库中的值集，以及指定值集中的数据如何与模板合并来创建可向托管服务器推送的格式正确的配置文件。

此外，还可以使用 CML 编写脚本，这些脚本将在向托管服务器推送配置时运行。有关详细信息，请参见使用应用程序配置运行脚本 (第 234 页)。

CML 模板



XML 配置文件

SA 还可以管理 XML 配置文件。有关使用 XML 配置模板的详细信息，请参见管理 XML 配置文件 (第 282 页)。

配置模板

配置模板是实际配置文件的“模板化”版本，模板中已将此配置文件的值变为变量。通过使用 SA 客户端，可以定义模板的值集、将值集保存到 SA 数据库，然后将这些值传播到托管服务器上的实际配置文件。

值集存储在 SA 数据库中。通过将所有值存储在 SA 数据库中，您可以从中心位置管理配置值并确保数据中心中的应用程序配置一致。

在创建配置文件的模板版本、将其添加到应用程序配置对象并为此模板创建值集之后，可以将这些值推送到托管服务器上的配置文件。

CML 概述

配置模板由一系列 CML 标记组成。每个标记代表一个 CML 分析器指令，指示如何解释配置文件中的文本或占位符(可标识配置文件中值的位置以及如何将此值映射到值集中)。

请注意，配置模板不包含任何值。它仅定义在 SA 数据库中值集和托管服务器上配置文件实例之间如何移动值。有关详细信息，请参见 值集 (第 221 页)。

通常，包含 CML 的模板文件的名称以“.tpl”作为文件扩展名，但是此文件扩展名并非必需的。

CML 标记的结构

CML 标记的基本构建块类似如下：

```
@{level}{tagtype}{source};{type};{range};{option};...;{option}@
```

> **备注：**
> CML 标记内既不能有空格也不能有“@”。'@' 符号可以通过在本身前面前置一个 '@' 来实现转义。

以下规则可应用于所有 CML 标记：

- 所有 CML 标记以 @ 符号开头和结尾。
- 针对忽略的特性使用分号 (;) 标记占位符。例如，以下显示 @name 特性和 optional@ 特性之间的两个忽略特性。

  ```
  @name;;;optional@
  ```

- 如果分号右侧的特性为空，则分号为可选符号。例如：
  ```
  @name@
  ```
- **{level}** - 块级别是用于指定块的嵌套级别的整数。级别还确定块是跨多行还是为单行的一部分。如果级别介于 1 和 99 之间，则为多行块。如果级别在 101 之上，则为单行中的块。每个块的打开标记均关闭所有先前级别相同或较大级别的块。

  请勿使用级别 100，因为级别 100 已被保留。
- **{tagtype}** - CML 定义以下列出的标记类型。每种类型是对 CML 分析器的一种指令。有关完整的详细信息，请参见CML 标记类型 (第 356 页)。
  - 注释标记：**@#** 和 **@##** ...**#@** - 定义模板中的注释。
  - 替换标记：**@** - 定义如何使用值集中的值替换变量。
  - 指令标记：**@!**- 为 CML 分析器提供指令。
  - 块标记：**@[@**...**@]@**- 创建新的范围。
  - 循环标记：**@*** - 创建对多个相似值的循环。
  - 循环目标标记：**@.**- 结束一个循环。
  - 条件标记：**@?**
  - DTD 标记：**@~** - 定义
- **{source}** - 定义值集中用于存储值的键。绝对路径名以“/”字符开头。相对路径名不以“/”开头，它们与 `@!namespace` 指令定义的命名空间键值连在一起。
- **{type}** - 定义配置文件所需的值以及值集中相应的值的数据类型。例如整型(代表整数)、字符串、布尔型、IP 地址等。此外还可指定有序和无序列表和集合。
- **{range}** - 定义数据值的其他限制，以便更好地进行错误检查。
- **{option}** - 定义可指定的其他参数，以修改 CML 标记的行为。

必需的 CML 标记

对于模板建模的配置文件，每个 CML 文件必须使用以下 CML 标记定义此配置文件的命名空间和默认文件名：

- @!namespace 定义模板的命名空间。模板所使用的值集中的所有值将存储在 SA 数据库中由 @!namespace 标记定义的键中。有关详细信息，请参见 使用 @!namespace CML 标记定义命名空间 (第 355 页)。
- @!filename-key 定义 SA 数据库中将存储默认文件名的特定键。此键可为独立的命名空间，也可附加到由 @!namespace 标记定义的命名空间。有关详细信息，请参见 使用 @!filename-key 和 @!filename-default CML 标记定义默认配置文件名 (第 355 页)。
- @!filename-default 定义模板建模的配置文件的目录和名称。此值可通过值集进行修改。有关详细信息，请参见 使用 @!filename-key 和 @!filename-default CML 标记定义默认配置文件名 (第 355 页)。

使用 @!namespace CML 标记定义命名空间

CML 模板文件中的命名空间定义数据库中用于存储数据的唯一键值。此命名空间值以路径名表示，类似于文件系统中的目录路径名或 Web 浏览器位置栏中的 URI。使用 namespace 标记定义命名空间。

单个值的路径名可以是绝对或相对的。绝对路径名以“/”开头，是值集中值位置的完整表示。不以“/”开头的路径名是相对路径名；它的值将附加到命名空间的当前值。

namespace 标记是必需标记。

CML 模板中的所有键名称必须为 ASCII。其他字段和文本可以是 ASCII 或非 ASCII 文本。

以下是 CML 模板中的 namespace 标记示例：

```
@!namespace=/security/@
```

使用 @!filename-key 和 @!filename-default CML 标记定义默认配置文件名

每个模板必须定义在向服务器推送生成的配置文件时将使用的默认配置文件名。此文件名可由值集覆盖。使用 filename-default 标记定义默认文件名。

此外还必须指定 SA 数据库中将存储默认文件名的唯一键。此键可与命名空间组合来生成默认文件名的唯一存储位置。此键定义命名空间以路径名表示。使用 filename-key 标记定义键值。

filename-default 和 filename-key 标记为必需标记。

CML 模板中的所有键名称必须为 ASCII。其他字段和文本可以是 ASCII 或非 ASCII 文本。

以下示例 CML 指定 SA 数据库中的键值“/files/hosts”将用于存储默认文件名。此外，它还指定生成的配置文件的默认文件名将为“/etc/hosts”。

```
@!filename-key="/files/hosts"@
@!filename-default="/etc/hosts"@
```

还可以在一行中组合 CML 标记，如下所示：

```
@!filename-key="/files/hosts";filename-default="/etc/hosts"@
```

/etc/hosts 的 CML 模板示例

以下示例是对典型 /etc/hosts 文件进行建模的 CML 模板。

```
@##############################################
#                                            #
# /etc/hosts (multiplatform)                 #
# Version 2.0                                #
# Joe Author (joe_author@your_company.com)   #
#                                            #
##############################################@
@!namespace=/system/dns/@
@!filename-key="/files/hosts";filename-default="/etc/hosts"@
@!unordered-lines;missing-values-are-error@
@!relaxed-whitespace@
@!sequence-delimiter-is-whitespace@
@!line-comment="#"@
@~host/.ip
type = ip
printable = IP address
description = This is an IP address
@
@~host/.hostnames
type = unordered-hostname-set
printable = Hostnames
description = A set of hostnames
@
@1*host;unordered-namespace-set;;sequence-append@
@.ip@   .hostnames@
@1]@
```

CML 标记类型

以下是主要的 CML 标记。下方详细描述了这些标记。

- 注释标记:@# 和 @## (第 357 页)
- 替换标记：@ (第 358 页)
- 指令标记:@! (第 359 页)
- 块 (或组) 标记:@[@...@]@ (第 360 页)

注释标记:@# 和 @##

此标记定义 CML 模板文件中的注释。可以定义单行注释或多行注释。

语法

```
@#   <one line comment>
```

或：

```
@## <comments spanning multiple lines>
    <comments spanning multiple lines>
    <comments spanning multiple lines> #@
```

描述

注释标记可用于在 CML 文件中的任何位置插入注释。

最佳实践是在 CML 模板的开头使用注释标记以创建用于描述模板的标题，例如模板的名称、模板所基于的配置文件、模板用途、作者、日期等等。

特性

无。

示例

以下是单行注释：

```
@# This comment ends at the end of this line.
```

以下是多行注释：

```
@##

 This comment spans

   multiple lines.

#@
```

以下也是多行注释：

```
@#############################################################
# /etc/hosts (multiplatform) #
# $Id: hosts.tpl 8650 2006-06-05 05:28:03Z joe_author $
#############################################################@
```

替换标记：@

此标记使用值集中的值替换模板文件中的文本。

语法

```
@{source}[;[{type}][;[{range}[;{option}[;{option}]...]]]]@
```

描述

此替换标记使用命名空间中指定位置的数据替换 CML 行中的标记。它指示此位置中的文本为数据，并且还指定有关如何存储和验证该数据的详细信息。源名称是在值集中找到数据的索引键。替换标记的其他字段指定有关如何存储和验证数据的详细信息。

替换标记是唯一不由后跟“@”字符的特殊字符表示的标记。替换标记中唯一必需的元素是源。所有其他元素是可选元素。

特性

- **源**：源特性是用于存储和访问值集中值的键。如果源特性为相对特性(即不以“/”或“.”开头)，则它会附加到当前的命名空间并且成为键(用于存储由此标记读取的值)的一部分。如果此名称为绝对名称(即以“/”开头)，则它是一个键并且值将存储在此键下。
- 替换标记中唯一必需的元素是源。所有其他元素是可选元素。如果名称以“.”开头，则它将附加到其所属循环的命名空间。循环中的标记通常以“.”开头。
- **类型**：类型特性指定替换标记的类型，它将某些预定义的限制和错误检查应用于不同的值。替换标记的默认类型为“字符串”。
- CML 类型特性 (第 367 页)中描述了可用的类型。
- **范围**：通过范围特性，可以设置重要值的范围。(请记住，在读取文件以及接受用户值时将使用所有范围。)如果配置文件中的某个值不在指定范围之内，则在分析该文件时将发生错误。
- CML 范围特性 (第 373 页)中描述了范围。
- **选项**：选项特性修改标记的行为。大多数标记的结尾可附加多个选项，由分号进行分隔。第三个分号后的所有内容将被视为一个选项。选项也可用作指令标记。

CML 全局选项特性 (第 376 页)和 CML 常规选项特性 (第 378 页)中描述了选项。

示例 1

```
Title=@main_title@
```

在本例中，main_title 将提取配置文件中“Title=”文本后的字符串，并将其存储在值集的 /main_title 键位置中。

或者，如果正在执行推送，main_title 将提取存储在值集的 /main_title 位置中的值，并将此值推送到配置文件的字符串 "Title=" 文本之后。

示例 2

```
Port = @port;port;1024<=&<=2048@
IPAddress = @ipaddress;ip;;optional;delimiter=”/”@
ServerName = @servername;hostname;”localhost”,r”server.*”@
```

指令标记:@!

此标记指定分析器的操作。有关可用指令的列表，请参见CML 全局选项特性 (第 376 页)和CML 常规选项特性 (第 378 页)。

语法

```
@!{option}[[;{option}]...]@
```

描述

此指令标记设置将在分析时使用的选项。例如，定义命名空间，列表是否已排序、是有序还是无序，分析器应如何解释空格、可接受的分隔符、定义注释字符等。

指令标记使用的唯一特性是选项。指令标记中可出现一个或多个选项。由分号分隔多个选项。要理解任何特定指令标记如何影响分析器，请参见嵌入式选项的描述。

特性

指令标记仅与选项特性一起使用。

选项：指令标记中的选项特性定义标记的行为。大多数标记的结尾可附加多个选项，由分号进行分隔。很多选项与其他选项交替出现。当块中显示交替组的一个组中的某个选项时，该组中的其他选项都不会显示在相同的块中。

CML 全局选项特性 (第 376 页)和CML 常规选项特性 (第 378 页)中描述了选项。

示例 1

以下指令标记指示 CML 分析器，任意制表符和空格的组合将与模板中的空格匹配。

```
@!relaxed-whitespace@
```

示例 2

以下指令标记中的两个选项指示 CML 分析器，配置文件中行的相对顺序对这些行中的值与值集的映射不重要，并且如果配置文件中的文本与值集中的值不匹配，这并不是错误。

```
@!unordered-lines;missing-values-are-null@
```

示例 3

```
@!namespace=/test/@
@!filename-key="/test";filename-default="/tmp/test.txt"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@
```

块 (或组) 标记:@[@...@]@

块标记有时被称为组标记。此标记可创建相关标记的块或组并允许嵌套相关标记的组。

语法

块标记可以具有单行语法，也可以具有多行语法。

块标记的单行语法如下所示 (带引号的文本字符串):

```
"@" [{level}] "[" [ ";" {option}[ ";" {option}]...]"@" {CML statements} "@"
[{level}] "]@"
```

块标记的多行语法如下所示:

```
"@" {level} "[" [ ";" {option}[ ";" {option}]...]"@"
{CML statements}
"@" {level} "]@"
```

@[{level}][[;{option}[;{option}]...]@{set of CML tags} @[{level}]]@

@[{level}][[;{option}[;{option}]...]@

{set of CML tags}

@[{level}]]@

级别是一个整数，用以确定块是跨多行还是为单行的一部分。如果级别介于 1 和 99 之间，则为多行块。如果级别大于 101，则为单行中的块。请勿使用级别 100，因为已将其保留。

可显式或隐式结束每个块。要显式结束块，请使用包含级别数的结束块标记。例如，以下标记显式关闭级别 3 块：@3]@。

要隐式结束块，请使用包含较低级别数的结束块标记来结束封闭块，或定义较低级别的新块。每个块的打开标记均将关闭所有先前级别相同或较大级别的块。

描述

通过块标记，可以对相关标记进行分组以及嵌套标记组。借助块，可以为配置文件的每个部分定义单独的分析规则。

可使用较高级别数将块嵌套在其他块中。包含级别值的任何后续标记都将关闭所有打开的具有相同或较大值的级别。块关闭标记 @]@ 不是必需的标记。

打开块标记可以包括选项特性。这些特性仅影响打开标记声明的级别块中的标记。将其与块中显示的指令标记进行对比：这些指令标记影响当前级别和任何嵌套块的行为。

通过使用块，可以为配置文件的每个单独部分指定唯一选项。例如，在配置文件的某个部分中，分别将 True 和 False 的值定义为“1”和“0”。在相同文件的其他部分中，可将 True 和 False 的值定义为“T”和“F”。使用块标记可以独立采用两种不同的方式定义 True 和 False。

还有其他示例，例如对于配置文件中的某个部分，特定空间数十分重要，然后对于其他部分，任何空间数都是可以接受的。可以使用块标记表示空间数量不同的位置。

特性

块标记不与名称、类型或范围特性一起使用。

- **级别**：块级别是用于指定块的嵌套级别的整数。级别还确定块是跨多行还是为单行的一部分。如果级别介于 1 和 99 之间，则为多行块。如果级别在 101 之上，则为单行中的块。每个块的打开标记均关闭所有先前级别相同或较大级别的块。

  > **备注：**
  > 请勿使用级别 100，因为级别 100 已被保留。

- **选项**：选项特性可修改块中 CML 标记的行为。块中的指令标记可影响当前块和嵌套块中 CML 标记的行为。大多数标记的结尾可附加多个选项，由分号进行分隔。选项也可用作指令标记。

  CML 全局选项特性 (第 376 页)和 CML 常规选项特性 (第 378 页)中描述了选项。

示例 1

以下示例创建了两个块，其中一个块嵌套在另一个块中。第一行定义第一个块，它是外部块。第四行定义第二个块，它是嵌套在第一个块中的内部块。倒数第二行关闭内部块。此

行为可选行。最后一行关闭外部块。如果忽略倒数第二行，则最后一行将同时关闭这两个块。

```
@1[@
@!ordered-lines@
[SectionOne]
@2[@
@!unordered-lines@
optionA = @section_one/option_a@
optionB = @section_one/option_b@
@2]@
@1]@
```

示例 2

此示例对 Windows `UrlScan.ini` 文件中的名为 [Options] 和 [AllowVerbs] 的两个部分进行建模。此文件中的这两个部分均包含键值对列表。

要定义第一个部分(行 1 到 3)，可以使用两个级别的块标记 ([)集，因为此部分中存在两种数据：固定的标题，后跟键值对列表。第一个级别块处理文本字符串“[Options]”，而第二个级别块处理该部分中所有的键值对。

第二个部分(行 4 到 6)定义 [AllowVerbs] 部分。请注意，使用 `@2]@` 和 `@1]@` 标记不会如上一个示例一样显式关闭第一个部分，因为打开下一个级别 1 部分(行 4)会隐式关闭先前的部分。

```
@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
```

循环标记：@*

此标记可定义处理循环。另请参见循环目标标记：@. (第 364 页)。

语法

```
@[{level}]*{source}[;[{type}][;[{range}[;{option}[;{option}]...]]]]@ {target}
```

描述

当值集在配置文件中多次出现时，可使用循环标记。循环标记的默认行为是直接在 CML 行之后遍历该行，虽然可将其修改为多行遍历或在单行中遍历。

循环是一种组标记，有关详细信息，请参见组标记。

循环标记允许枚举序列(列表和集合)。将在输入文件中为与循环元素关联的块的每个突发事件处理该块，并将在输出文件中为值集中该数据的每个匹配项生成此块。

与循环元素关联的组将使新元素存储在配置文件中该组的每个突发事件的值集中，或者值集中该数据的每个匹配项会将某个值推送到此配置文件。源特性是用于映射值集中的值的索引键。

特性

- **级别**：组级别是一个整数，用以确定组是跨多行还是为单行的一部分。如果级别介于 1 和 99 之间，则为多行组。如果级别在 101 之上，则为单行中的组。将保留级别 100 用于内部用途。每个组的打开标记均将关闭所有先前级别相同或较大级别的组。
- **源**：源特性是用于访问值集中值的键。如果源特性为相对特性(即不以“/”或“.”开头)，则它会附加到当前的命名空间并且成为键(用于存储由此标记读取的值)的一部分。如果此名称为绝对名称(即以“/”开头)，则它是一个键并且值将存储在此键下。循环标记中唯一必需的元素是源，任何其他元素都是可选元素。如果源名称以“.”开头，则它会附加到其所属的任意循环的命名空间。循环中的标记通常以“.”开头。
- **类型**：类型特性指定替换标记的类型，它将某些预定义的限制和错误检查应用于不同的值。替换标记的默认类型为“字符串”。

  有关类型的完整列表，请参见CML 类型特性 (第 367 页)。

  可将“ordered-”或“unordered-”前置于类型。可将“-set”或“-list”附加到类型。
  - 前置“ordered-”指定必须按顺序排列值。
  - 前置“unordered-”指定可按任何顺序排列值。
  - 附加“-set”指定值必须唯一。
  - 附加“-list”指定值可以重复。
- **范围**：通过范围特性，可以设置值的范围。请记住，在读取文件以及接受用户值时将使用所有范围。如果配置文件中的某个值不在指定范围之内，则在分析该文件时将发生错误。
- CML 范围特性 (第 373 页)中描述了范围。
- **选项**：选项特性可修改标记的行为。大多数标记的结尾可附加多个选项，由分号进行分隔。第三个分号后的所有内容将被视为一个选项。选项也可用作指令标记。

  CML 全局选项特性 (第 376 页)和CML 常规选项特性 (第 378 页)中描述了选项。

示例 1

星号字符表示循环标记。例如：

```
@1*includegroup;ordered-namespace-set;;optional@
#BEGIN_ALTERNATE
@*.include@
#INCLUDE @.@
#END_ALTERNATE
@1]@
```

示例 2

```
@*users;unordered-user-set;!”root”;field-delimiter-is-semicolon@
@.@;
```

循环目标标记：@.

循环目标标记定义循环标记的迭代。请参见 循环标记：@* (第 362 页)。

语法

```
@.[{source}[;[{type}][;[{range}[;{option}[;{option}]...]]]]]@
```

描述

循环目标标记表示循环中值的占位符。如果认为循环标记表示循环的开始(因而与组标记相似)，则循环目标标记与替换标记十分相似。

在组中遇到每个循环迭代时，此标记仅将配置文件中当前位置的文本与值集中的当前值进行映射。如果使用可选源特性，则会将源附加到循环所创建的命名空间。

特性

无。

示例

循环目标标记由“@”字符后跟一个句点来表示。例如：

```
@*keys;unordered-namespace-set@
@.key@ = @.value@
```

条件标记：@?

此标记可定义条件。

语法

```
@[{level}]?{source}@{text}
```

描述

条件标记使用命名空间中的布尔值映射配置文件中是否存在某个文本。在读取目标配置文件时，如果文本匹配，则命名空间值为 true，否则为 false。在写入配置文件时，如果命名空间值为 true，则此配置文件获取此文本；否则将不会写入任何文本。

此标记外部的某些内容实际为值，这是为数不多具有这种情况的几个标记中的一个。此标记的主要用途是将布尔值存储在命名空间的某个位置中(如果此标记之后存在文本)。

特性

条件标记不与类型、范围或选项特性一起使用。

- **级别**：级别是一个整数，用以确定组是跨多行还是为单行的一部分。如果级别介于 1 和 99 之间，则为多行组，如果在 101 以上，则为单行中的组。将保留级别 100 用于内部用途。每个组的打开标记均将关闭所有先前级别相同或较大级别的组。
- **源**：源特性是用于访问布尔值的键。如果源特性为相对特性(不以“/”或“.”开头)，则它会附加到当前的命名空间并且成为键(用于存储由此标记读取的值)的一部分。如果此名称为绝对名称(以“/”开头)，则它*是*一个键并且值将存储在此键下。

  如果源名称以“.”开头，则它会附加到其所属的任意循环的命名空间。通常，循环中的标记应以“.”开头。

示例 1

条件标记以问号 (?) 表示。例如：

```
@?debug@options debug
```

在本例中，如果将配置文件导入配置模板并且配置文件中存在“options debug”文本，则 /debug 键中的值将被设置为 true。

如果要推送应用程序配置并且 **/debug** 键中存储的值为 true，则会将“options debug”文本推送到配置文件。

示例 2

例如，如果配置文件指定基于配置文件中的“threaded”关键字线性运行应用程序，则 CML 可能类似如下：

```
@?is_threaded@threaded
```

如果配置文件中存在“threaded”值，则会将命名空间键 /is_threaded 中的值设置为 true，如果配置文件中不存在“threaded”值，则将其设置为 false。

DTD 标记：@~

此标记可定义 DTD。

语法

```
@~{source}
[type = {type}]
[description = {description}]
[printable = {printable}]
[range = {range}]
[{option}
...]
@
```

描述

CML 支持使用文档类型定义 (DTD) 标记，此标记可用于预定义其他 CML 标记的特性。通过将 DTD 标记的所有特征存储于其他位置并按名称仅引用此标记本身，可将此标记用于使 CML 模板的实际功能部分更加清晰。

DTD 定义可用于定义包含源特性的任何标记；例如循环标记、循环目标标记、替换标记，但是不可定义类似于指令标记或组标记(不包含源特性)的标记。

在 CML 中使用 DTD 标记的其他优点是可定义“PRINTABLE”和“DESCRIPTION”值。“PRINTABLE”和“DESCRIPTION”值为用户提供了有关字段计划用途的某些反馈。在值集编辑器屏幕中将鼠标光标滚动到此字段上时，将显示 DESCRIPTION 特性的字符串值。Printable 性的字符串值将使用更易于阅读的字段标签替换值集编辑器中的路径名。

CML 中的 DTD 标记本身还是多行标记。除第一行和最后一行之外，所有其他行都可按任何顺序排列，并且此处所有元素都与标记中的字段相关(除可打印和描述元素之外，这两个元素仅对 DTD 定义标记有效)。

有关在配置模板中使用 DTD 标记的详细信息，请参见在 CML 中使用 DTD 标记 (第 391 页)。有关 XML 模板，请参见自定义 XML DTD 元素显示方式 (第 287 页)。

特性

DTD 标记不与任何级别特性一起使用。DTD 标记中唯一必需的特性是源；任何其他特性都是可选特性。但是，仅定义了名称的 DTD 标记没有任何作用。

- **源**：源特性是用于访问值的键。如果源特性为相对特性(不以“/”或“.”开头)，则它会附加到当前的命名空间并且成为键(用于存储由此标记读取的值)的一部分。如果此名称为绝对名称(以“/”开头)，则它*是*一个键并且值将存储在此键下。如果源名称以“.”开头，则它会附加到其所属的任意循环的命名空间。通常，循环中的标记应以“.”开头。
- **类型**：类型特性根据已知类型将某些预定义限制和错误检查分配到不同的值。替换标记的默认类型为“字符串”，此类型将匹配或多或少的任何内容。
- 在此文档的CML 类型特性 (第 367 页)中可找到类型的完整列表。
- **描述**：描述特性的值是一个字符串，它简要描述了此标记所代表的值种类。此特性将显示为 SA 客户端值集编辑器中的鼠标悬停文本。
- **PRINTABLE**:可打印特性的值是一个字符串，它仅是变量的整洁名称。它将在 SA 客户端值集编辑器中显示为特性的名称。
- **范围**：通过范围特性，可以设置值的范围。请记住，在读取文件以及接受用户值时将使用所有范围。如果配置文件中的某个值不在模板设置的范围之内，则在分析该文件时可能会抛出异常。最好基于配置文件的文档使用正确的范围。
- CML 范围特性 (第 373 页)中完整描述了范围。
- **选项**：选项特性用于修改或影响标记的行为。大多数标记的结尾可附加多个选项，由分号进行分隔。可将所需的足够多的选项附加到标记，在第三个分号后的所有内容均被视为一个选项，并且每个选项由分号进行分隔。选项也可用作指令标记。

  CML 全局选项特性 (第 376 页)和CML 常规选项特性 (第 378 页)中完整描述了选项。

示例

```
@~port
type = port
range = 1024<=&<=2048
printable = Port
description = The port used for this application.It
should be a port number between 1024 and 2048
@
```

CML 类型特性

CML 特性定义和控制 CML 标记的语义。此部分定义 CML 模板中可使用的类型。请注意，通过将“-set”或“-list”附加到某些类型，可将这些类型修改为表示重复值的序列。通过将“ordered-”或“unordered-”前置于某些类型，可将这些类型修改为忽略重复值的序列顺序。

int 类型

Int 是一种数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;int][;[{range}][;{option}[;{option}]...]]]@
```

描述

一个整数值 ...、-2、-1、0、1、2、... (Z)。

十进制类型

十进制是一种数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;decimal][;[{range}][;{option}[;{option}]...]]]@
```

描述

十进制数字。

GUID 类型

GUID 是一种数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;guid][;[{range}][;{option}[;{option}]...]]]@
```

描述

全局唯一标识符 (GUID)，128 位 ID。

字符串类型

字符串是一种非数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;string][;[{range}][;{option}[;{option}]...]]]@
```

描述

如果未明确指定任何其他类型，则所有值的默认类型为字符串。

引用字符串类型

引用字符串是一种非数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;quotedstring][;[{range}][;{option}[;
{option}]...]]]@
```

描述

引用字符串。

布尔值类型

布尔值是一种非数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;boolean][;[{range}][;{option}[;{option}]...]]]@
```

描述

布尔值。

持续时间类型

持续时间是一种非数字类型。

语法

```
@[{level}]{tag-type}[[{source}][;duration][;[{range}][;{option}[;{option}]...]]]@
```

描述

持续时间。

ipv6 类型

IPv6 是特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;ipv6][;[{range}][;{option}[;{option}]...]]]@
```

描述

CML 支持以下两种约定格式，用于将 IPv6 地址显示为文本字符串：

x:x:x:x:x:x:x:x，其中，“x”表示 IPv6 地址的 8 个 16 位部分的 1 至 4 位十六进制数。例如：

```
ABCD:EF01:2345:6789:ABCD:EF01:2345:6789
```

IPv6 地址中的“::”表示一组或多组 16 位零。“::”在一个地址中只能出现一次。“::”也可用于压缩一个地址开头或结尾的零。例如：

2001:DB8:0:0:8:800:200C:417A 变为 2001:DB8::8:800:200C:417A

0:0:0:0:0:0:0:1 变为 ::1

ipv4 类型

IPv4 是特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;ipv4][;[{range}][;{option}[;{option}]...]]]@
```

描述

IP v4 地址。

ip 类型

Ip 是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;ip][;[{range}][;{option}[;{option}]...]]]@
```

描述

IP 地址(ipv4 和 ipv6)。

主机名类型

主机名是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;hostname][;[{range}][;{option}[;{option}]...]]]@
```

描述

主机服务器的名称。

主机类型

主机是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;host][;[{range}][;{option}[;{option}]...]]]@
```

描述

主机 IP 地址或主机名。

网络类型

网络是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;network][;[{range}][;{option}[;{option}]...]]]@
```

描述

IP v4 网络。

端口类型

端口是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;port][;[{range}][;{option}[;{option}]...]]]@
```

描述

TCP 或 UDP 端口。

用户类型

用户是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;user][;[{range}][;{option}[;{option}]...]]]@
```

描述

用户名。

组类型

组是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;group][;[{range}][;{option}[;{option}]...]]]@
```

描述

组名称。

文件 – 特定于系统的类型

语法

```
@[{level}]{tag-type}[[{source}][;file][;[{range}][;{option}[;{option}]...]]]@
```

描述

文件名。

目录类型

目录是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;dir][;[{range}][;{option}[;{option}]...]]]@
```

描述

目录路径名。

电子邮件类型

电子邮件是一种特定于系统的类型。

语法

```
@[{level}]{tag-type}[[{source}][;email][;[{range}][;{option}[;{option}]...]]]@
```

描述

电子邮件地址。

CML 范围特性

CML 特性定义和控制 CML 标记的语义。此部分定义 CML 模板中可使用的范围特性。对于给定 CML 类型，范围特性允许您使用范围说明符定义和限制标记的有效值。

!& , – 逻辑运算符

!– not 说明符

& – and 说明符

, – or 说明符

语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;!{range}][;{option}[;{option}]...]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;{range}&{range}][;{option}[;
{option}]...]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;{range},{range}][;{option}[;
{option}]...]]]@
```

描述

逻辑运算符可修改范围说明符，以控制输入的验证方式。三种可用的运算符(按优先顺序排列)包括：not、and、or。

- not 运算符由感叹号表示，是一个前缀一元运算符。它求反范围的意义，这意味着符合范围要求的项将返回 false，不符合范围的项将返回 true。
- and 运算符由与号表示，是一个中缀二进制运算符。当且仅当两个操作数均返回 true 时，它才会返回 true。
- or 运算符由逗号表示，是一个中缀二进制运算符。当且仅当任一操作数返回 true 时，它才会返回 true。

在指定范围时，空格并不重要。

> **备注：**
> 当前 CML 分析器要求 CML 标记内不出现空格和“@”。

n< n<= <n <=n =n – 比较说明符

n< – 大于说明符

n<= – 大于或等于说明符

<n – 小于说明符

<=n – 小于或等于说明符

=n – 等于说明符

语法

@[{level}]{tag-type}[[{source}][;[{type}][;{number}<][;{option}[;{option}]...]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;{number}<=][;{option}[;{option}]...]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;<{number}][;{option}[;{option}]...]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;<={number}][;{option}[;{option}]...]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;={number}][;{option}[;{option}]...]]]@

描述

可用于数字值的说明符包括：大于、大于或等于、小于、小于或等于以及等于。

大于说明符 (n<) 由一个数字和后跟的一个左尖括号字符构成。大于指定数字的数字值将符合此范围。

大于或等于说明符 (n<=) 由一个数字、后跟的一个左尖括号字符以及最后的等号字符构成。大于或等于指定数字的数字值将符合此范围。(请注意，对于数字 n，n<= 与 !<n 相等，也与 n<,=n 相等，提供 n<= 以便于使用)

小于说明符 (<n) 由一个左尖括号字符和后跟的一个数字构成。大于指定数字的数字值将符合此范围。

小于或等于说明符 (<=n) 由一个左尖括号字符、后跟的一个等号字符以及最后的一个数字构成。大于或等于指定数字的数字值将符合此范围。(请注意，对于数字 n，<=n 与 !n< 相等，也与 <n,=n 相等，提供 <=n 以便于使用)

等号说明符 (=n) 由一个等号字符和后跟的一个数字构成。与指定数字相等的数字值将符合此范围。

建议在提供由一个 and 运算符分隔的两个范围说明符时，大于(或等于)说明符应先于小于(或等于)说明符，例如 0<=&<256。

在指定范围时，空格并不重要。

> **备注:**
> 当前 CML 分析器要求 CML 标记内不出现空格和“@”。

"– 字符串文本说明符

语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;"{string}"][;{option}[;{option}]...]]]@
```

描述

字符串文本说明符由一个双引号字符、后跟一个文本字符串以及最后一个双引号字符构成。引号和转义规则遵循 C 语言的规则；即，嵌入式引号转义为反斜杠，换行符由 \n 表示，制表符由 \t 表示，文本反斜杠由 \\ 表示。完全匹配此文本的字符串值将符合此范围。

在指定范围时，空格并不重要。

> **备注:**
> 当前 CML 分析器要求 CML 标记内不出现空格和“@”。

r" – 正则表达式说明符

语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;r"{regular expression}"][;{option}[;{option}]...]]]@
```

描述

正则表达式说明符由一个“r”字符、双引号字符、后跟一个正则表达式以及最后一个双引号字符 (") 构成。引号和转义规则遵循 Python 正则表达式的规则(除引号字符以外，此字符必须转义为反斜杠字符)。与正则表达式匹配的字符串值将符合此范围。

在指定范围时，空格并不重要。

> **备注:**
> 当前 CML 分析器要求 CML 标记内不出现空格和“@”。

CML 全局选项特性

CML 特性定义和控制 CML 标记的语义。此部分定义 CML 模板中可使用的全局特性。全局选项只能用于指令标记，不能用作其他标记类型的选项特性。

@!filename-key 特性

语法

```
@!filename-key={key}@
```

{key} 没有默认值。

描述

filename-key 确定将包含推送期间已生成文件的文件名的值集中键的路径。

filename-key 值是一个路径名。可写为相对路径且无需以斜线 (/) 开始。

filename-key 值不得以 / 结尾。可能会在之后的版本中对此不作要求。

@!filename-default 特性

语法

@!filename-default={filename}@

{filename} 没有默认值。

描述

filename-default 确定在值集中无文件名时将返回的默认文件名。例如，用户可能在值集编辑器中输入文件名，因而覆盖默认文件名值。

@!full-template 和 @!partial-template 特性

语法

@!full-template@

@!partial-template@

full-template 是默认行为。

描述

full-template 是默认行为，表示必须在模板中对文件中的所有预期数据进行建模。

partial-template 表示应忽略文件中不匹配的数据并直接传递到输出中。此选项只能与 preserve-format 一起使用。

@!timeout 特性

语法

@!timeout={minutes}@

{minutes} 默认值为 1。

描述

timeout 表示应添加到配置总超时的分钟数。有效超时值是介于 0(含)到 999(含)之间的任意整数。将配置中所有模板的超时值相加并将所得数字加到配置的默认超时值(10 分钟)，就会得到整个配置的最终超时值。

@!unix-newlines 和 @!windows-newlines 特性

语法

```
@!unix-newlines@
@!windows-newlines@
```

unix-newlines 是默认行为。

描述

unix-newlines 是默认行为，表示由此模板生成的配置文件将具有 unix 风格的换行符(ASCII 换行符)。

windows-newlines 表示由此模板生成的配置文件将具有 Windows 风格的换行符(ASCII 回车符和换行符组合)。

CML 常规选项特性

CML 特性定义和控制 CML 标记的语义。此部分定义 CML 模板中可使用的选项特性。常规选项可用作指令标记，也可用作其他标记类型的选项特性。

@! unordered-lines 和 @!ordered-lines 特性

指令标记语法

```
@!unordered-lines@
@!ordered-lines@
```

unordered-lines 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;unordered-lines[;
{option}]...]]]]@
```

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;ordered-lines[;
{option}]...]]]]@
```

对组有效。

描述

unordered-lines 允许以任何顺序显示模板的子标记；但是，将保留有序序列元素中项的位置。unordered-lines 是默认行为。

ordered-lines 指示分析器，模板对象(行、循环、条件等)的子标记必须按模板中指定它们的顺序显示在文件中。

无序和有序元素特性

指令标记语法

```
@!unordered-elements@
@!ordered-elements@
```

unordered-elements 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;unordered-elements[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;ordered-elements[;
{option}]...]]]]@
```

对组有效。

描述

unordered-elements 允许以任何顺序显示当前组的子标记；但是，将保留有序序列元素中项的位置。unordered-elements 是默认行为。

ordered-elements 指示分析器，组对象(循环、条件、元素等)的子标记必须按模板中指定它们的顺序显示在文件中。

relaxed-whitespace 和 strict-whitespace 特性

指令标记语法

```
@!relaxed-whitespace@
@!strict-whitespace@
```

relaxed-whitespace 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;relaxed-whitespace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;strict-whitespace[;
{option}]...]]]]@
```

对组有效。

描述

relaxed-whitespace 允许模板中的空格与制表符和空格的任何组合匹配。relaxed-whitespace 是默认行为。

strict-whitespace 需要模板中的空格与该文件中的空格完全匹配。

required-whitespace 和 optional-whitespace 特性

指令标记语法

```
@!required-whitespace@
@!optional-whitespace@
```

required-whitespace 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;required-whitespace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;optional-whitespace[;
{option}]...]]]]@
```

对组有效。

描述

required-whitespace 要求模板中的空格必须在文件中。optional-whitespace 可使不重要的空格选择性地显示在文件中。

missing-values-are-null 和 missing-values-are-error 特性

指令标记语法

```
@!missing-values-are-null@
```

```
@!missing-values-are-error@
```

missing-values-are-null 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;missing-values-are-null[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;missing-values-are-error[;
{option}]...]]]]@
```

描述

missing-values-are-null 指示文件中找不到的值为 null，因此值集中不会提供这些值。

missing-values-are-error 在文件或值集中找不到模板中指定的所有值时会抛出错误。

case-insensitive-keywords 和 case-sensitive-keywords 特性

指令标记语法

```
@!case-insensitive-keywords@
@!case-sensitive-keywords@
```

case-insensitive-keywords 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;case-insensitive-keywords[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;case-sensitive-keywords[;
{option}]...]]]]@
```

描述

case-insensitive-keywords 匹配文件中忽略大小写的文字类型文本。case-insensitive-keywords 是默认行为。

case-sensitive-keywords 指示模板中的文字类型文本必须在区分大小写的基础上与文件的文字类型文本匹配。

reluctant 特性

指令标记语法

@!reluctant@

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;reluctant[;{option}]...]]]]@

描述

reluctant 指定特定循环或序列将尝试匹配配置文件中尽可能少的元素。这不是循环和序列的默认行为。

required 和 optional 特性

指令标记语法

@!required@
@!optional@

required 是默认行为。

在指令标记中使用 optional 可能会产生意外结果。

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;required[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;optional[;{option}]...]]]]@

描述

required 元素必须匹配(除非嵌套在可选组中)。required 是默认行为。

optional 元素为可选元素。

将 optional 用作选项特性可对任何标记有效(除指令标记之外)。在指令标记中使用 optional 可能会产生意外结果。

skip-lines-without-values 和 show-lines-without-values 特性

指令标记语法

@!skip-lines-without-values@
@!show-lines-without-values@

skip-lines-without-values 是默认行为。

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;skip-lines-without-values[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;show-lines-without-values[;{option}]...]]]]@

描述

skip-lines-without-values 指示在某一行包含替换元素并且这些元素的所有值为 null 时，应从输出中取消该行。skip-lines-without-values 是默认行为。

show-lines-without-values 指示应显示所有行，无论是否包含 null 值。

skip-groups-without-values 和 show-groups-without-values 特性

指令标记语法

@!skip-groups-without-values@
@!show-groups-without-values@

skip-groups-without-values 是默认行为。

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;skip-groups-without-values[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;show-groups-without-values[;{option}]...]]]]@

描述

skip-groups-without-values 指示在某个组包含替换元素并且这些元素的所有值为 null 时，应从输出中取消该组。skip-groups-without-values 是默认行为。

show-groups-without-values 指示应显示所有组，无论是否包含 null 值。

sequence-append、sequence-replace 和 sequence-prepend 特性

指令标记语法

```
@!sequence-append@
@!sequence-replace@
@!sequence-prepend@
```

sequence-append 是默认行为。

对循环和序列有效。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-append[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-replace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-prepend[;
{option}]...]]]]@
```

描述

sequence-append 表示序列元素子范围附加到序列元素父范围。sequence-append 是默认行为。

sequence-replace 表示序列元素子范围替换序列元素父范围。

sequence-prepend 表示序列元素子范围前置于序列元素父范围。

not-primary-field 和 primary-field 特性

指令标记语法

```
@!not-primary-field@
@!primary-field@
```

not-primary-field 是默认行为。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;not-primary-field[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;primary-field[;
{option}]...]]]]@
```

描述

not-primary-field 表示在执行列表聚合时不应使用此字段来确定重复项。

not-primary-field 是默认行为。

primary-field 表示在执行列表聚合时应使用此字段来确定重复项。

对序列和序列中的替换标记有效。

namespace 特性

指令标记语法

```
@!namespace={namespace}@
```

{namespace} 的默认值为“/”(根命名空间)。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;namespace={namespace}[;
{option}]...]]]]@
```

{namespace} 的默认值为“/”(根命名空间)。

描述

namespace 是一个字符串，用以标识将存储包含非限定名称(前面不带有斜杠或句点的名称)的元素的命名空间。

{namespace} 的默认值为根命名空间，由字符串“/”(正斜杠)表示。

命名空间值为路径名。它必须以斜杠 (/) 开头。

boolean-no-format 特性

指令标记语法

@!boolean-no-format={string}@

{string} 的默认值为“no”

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;boolean-no-format={string}[;{option}]...]]]]@

{string} 的默认值为“no”

描述

boolean-no-format 可标识将用于匹配 false 布尔元素的字符串。对布尔替换标记有效。

boolean-yes-format 特性

指令标记语法

@!boolean-yes-format={string}@

{string} 的默认值为“yes”

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;boolean-yes-format={string}[;{option}]...]]]]@

{string} 的默认值为“yes”

描述

boolean-yes-format 和 boolean-no-format 可标识将用于匹配布尔元素的字符串。{string} 的默认值为“yes”

对布尔替换标记有效。

delimiter 特性

whitespace-delimited
comma-delimited
semicolon-delimited
tab-delimited
quote-delimited
delimiter

指令标记语法

```
@!whitespace-delimited@
@!comma-delimited@
@!semicolon-delimited@
@!tab-delimited@
@!quote-delimited@
@!delimiter={string}@
```

whitespace-delimited 是默认行为

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;whitespace-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;comma-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;semicolon-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;tab-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;quote-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;delimiter={string}[;
{option}]...]]]]@
```

whitespace-delimited 是默认行为。

描述

delimiter 可设置默认的分隔符字符。如果未明确指定，则 sequence-delimiter 和 field-delimiter 将继承此值。

对 replace 和 sequence 标记有效。

line-comment 特性

```
line-comment-is-comma
line-comment-is-semicolon
line-comment-is-tab
line-comment-is-whitespace
line-comment
```

指令标记语法

@!line-comment-is-comma@
@!line-comment-is-semicolon@
@!line-comment-is-tab@
@!line-comment-is-whitespace@
@!line-comment={string}@

{string} 没有默认值。

选项特性语法

@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-comma[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-semicolon[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-tab[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-whitespace[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment={string}[;{option}]...]]]]@

{string} 没有默认值。

描述

line-comment 可设置字符，该字符可表示行的剩余部分将被分析为注释。

sequence-delimiter 特性

sequence-delimiter-is-comma
sequence-delimiter-is-semicolon
sequence-delimiter-is-tab
sequence-delimiter-is-whitespace
sequence-delimiter-is-quote
sequence-delimiter

指令标记语法

@!sequence-delimiter-is-comma@
@!sequence-delimiter-is-semicolon@
@!sequence-delimiter-is-tab@
@!sequence-delimiter-is-whitespace@
@!sequence-delimiter-is-quote@
@!sequence-delimiter={string}@

默认情况下，sequence-delimiter 使用分隔符的值。后者的默认值为 whitespace-delimited。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-comma[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-issemicolon[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-tab[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-whitespace[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-quote[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter={string}[;{option}]...]]]]@
```

默认情况下，sequence-delimiter 使用分隔符的值。后者的默认值为 whitespace-delimited。

描述

sequence-delimiter 可设置用于分隔序列中项的字符。默认情况下，sequence-delimiter 使用分隔符的值。后者的默认值为 whitespace-delimited。

对序列有效。

field-delimiter 特性

```
field-delimiter-is-comma
field-delimiter-is-semicolon
field-delimiter-is-tab
field-delimiter-is-eol
field-delimiter-is-whitespace
field-delimiter-is-quote
field-delimiter
```

指令标记语法

```
@!field-delimiter-is-comma@
@!field-delimiter-is-semicolon@
@!field-delimiter-is-tab@
@!field-delimiter-is-whitespace@
```

```
@!field-delimiter-is-quote@
@!field-delimiter={string}@
```

默认情况下，field-delimiter 使用分隔符的值。后者的默认值为 whitespace-delimited。

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-comma[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-semicolon
[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-tab[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-whitespace
[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-quote[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter={string}[;
{option}]...]]]]@
```

默认情况下，field-delimiter 使用分隔符的值。后者的默认值为 whitespace-delimited。

描述

field-delimiter 可设置将用于终止替换元素值分析的字符。默认情况下，field-delimiter 使用分隔符的值。后者的默认值为 whitespace-delimited。

对替换标记和序列标记有效。

line-continuation 特性

指令标记语法

```
@!line-continuation={string}@
```

选项特性语法

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-continuation={string}[;
{option}]...]]]]@
```

描述

line-continuation 可设置一个字符，用于表示应将配置文件中的当前行换到后续行。

在 CML 中使用 DTD 标记

CML 支持使用文档类型定义 (DTD) 标记，此标记可用于预定义 CML 标记的特性。通过在 CML 中使用 DTD 标记，可以更改模板在 SA 客户端中的某些显示方式。通常在文件的开始部分进行 DTD 定义，并且将标记缩写为一个名称和一个标记类型。

在 CML 中使用 DTD 标记的主要优点是可以定义反映在 SA 客户端中的“printable”值和“description”值，从而提高可用性。DTD 定义可用于定义包含名称的任何标记；例如循环标记、循环目标标记、替换标记等，但是不可定义类似于指令标记或块标记的标记。CML 中的 DTD 标记本身还是多行标记。

DTD 标记示例

此处将选择一个标记并创建该标记的 DTD 版本。CML 中的 DTD 标记与常规 CML 标记并无不同；它包含除“标记类型”之外的所有标记元素。

例如，在以下 CML 标记中：

```
@*deny_header;unordered-string-set;;sequence-delimiter=":";optional@
```

这是代表以下 CML 格式的实例：

```
@<tag type><name>;<data type>;;<option1>;<option2>@
```

此标记的 DTD 版本选择现有元素并对其重新排序，如下所示：

```
<start code block>
@~<name>
type = <data type>
description = <description>
printable = <printable>
<option1>
<option2>
...
@
@<tag type><name>@
<end code block>
```

正如所见，此用法还允许附加两个新元素：“description”和“printable”。定义“printable”将在 SA 客户端中为标记定义主要文本。定义“description”将在 SA 客户端中为值创建描述，在将鼠标指针悬停在 SA 客户端的值集编辑器中的字段时，将看到此描述。

以下是完整 DTD 格式的同一个标记：

```
<start code block>
@~deny_header
type = unordered-string-set
printable = Headers to Deny
description = This is a list of headers that IIS should deny
sequence-delimiter = ":"
optional
@
@*deny_header@
<end code block>
```

在以上示例中需注意几点。在为“description”定义值时，只要第一行后的行的第一个字符为空格，此值便可以跨多行。

选项自成一行，在存在 <option>=<value> 的位置，需要在“=”符号之前和之后插入空格。

现在，在使用标记 @*deny_header@ 的位置，分析器将针对所有该标记的信息使用预定义的 DTD。

使用类似于 @*deny_header;unordered-string-set@ 的行重新定义 DTD 定义标记 @*deny_header@ 将导致 CML 模板变得无效。

> **备注：**
> 另请注意，DTD 风格的 CML 当前不是必需的，但是在 SA 客户端中查看应用程序配置时，它是最明显的 CML。如果不使用 DTD 标记，则将不会看到“printable”和“description”字段，而只会看到基本变量名。

序列聚合

由于可以跨应用程序配置继承层次结构(也称为继承范围)中的多个不同层级设置应用程序配置值，因此，在将应用程序配置推送到服务器时，您必须能够控制多个序列值合并到一起的方式。

使用 ACM，可以控制跨继承范围合并序列值的方式。这意味着，您可以(例如)分别在客户范围、组范围和服务器范围中将某些值添加到一个序列，所有值将合并到一起，构成最终的序列。

序列值合并的方式受 CML 模板中的特殊标记控制，使用下列三种不同的序列合并模式：

- 序列替换 (第 393 页):较具体范围中的序列值完全替换不太具体范围中的序列值。集合和列表序列都采用这种模式。
- 序列附加 (第 394 页):对于列表，较一般范围中的值附加 到较具体范围中的值(放在其后面)。重复值(如果有)将不删除。对于集合，行为几乎相同，只不过需要合并重复值。对于列表，根据标有 primary-key 标记的子元素来标识重复值，然后合并重复值。对于标量，只需删除重复值，仅保留较具体范围中的值(最后一个值是合并的序列)。这是默认模式，将在未指定任何其他模式时使用。
- **序列前置**:方式与附加相同，只不过较一般范围中的值前置 于较具体范围中的值(放在其前面)。

例如，对于下面两个集合：

- “a, b” – 位于继承范围的较具体(内部)层级，例如服务器实例层级。
- “c, d” – 位于继承范围的较一般(外部)层级，例如服务器组层级。

当将应用程序配置模板推送到服务器时，合并结果将为：

- 序列替换：“a, b”
- 序列附加：“a, b, c, d”
- 序列前置：“c, d, a, b”

序列聚合不仅在范围间发生，还发生在范围本身中。当命名空间序列中有重复值时显然会发生这种聚合。

序列替换

在替换合并模式(CML 标记为“sequence-replace”)中，在特定范围定义的序列内容将替换不太具体范围中的序列内容，不对序列的各个元素执行合并。

例如，如果已经为 配置模板 CML 源中的列表设置了 sequence-replace 标记，则在服务器实例层级中为该列表设置的值将覆盖或替换在组层级设置的值和在应用程序配置默认值层级设置的值。

例如，如果在组层级(外部)定义了 etc/hosts 文件中的列表，如下所示：

```
/system/dns/host/1/ip                    127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine
```

```
/system/dns/host/2/ip           10.10.10.10
/system/dns/host/2/hostnames/1  loghost
```

在设备范围(内部)定义了相同的列表，如下所示：

```
/system/dns/host/1/ip           127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine.mydomain.net
/system/dns/host/2/ip           10.10.10.100
/system/dns/host/2/hostnames/1  mailserver
```

如果模板已使用 sequence-replace 标记来定义 /system/dns/host 元素，则推送操作后服务器上配置文件的最终结果将为：

```
127.0.0.1 localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

序列附加

当对序列使用附加列表合并模式(CML 标记为“sequence-append”)时，较一般范围中的值附加到较具体范围中的值(放在其后面)。序列附加模式是合并列表值的默认模式。如果模板的 CML 中未指定任何内容，则将使用序列附加。

如果在组层级(外部)定义了 etc/hosts 文件中的列表，如下所示：

```
/system/dns/host/1/ip                      127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine
/system/dns/host/2/ip           10.10.10.10
/system/dns/host/2/hostnames/1  loghost
```

在设备范围(内部)定义了相同的列表，如下所示：

```
/system/dns/host/1/ip           127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine.mydomain.net
/system/dns/host/2/ip           10.10.10.100
/system/dns/host/2/hostnames/1  mailserver
```

使用上述示例中设置的值，如果 /system/dns/host 元素是在配置模板中设置了 sequence-append 标记的列表，则推送操作后服务器上配置文件的最终结果将为：

```
127.0.0.1 localhost mymachine.mydomain.net
10.10.10.100 mailserver
127.0.0.1 localhost mymachine
10.10.10.10 loghost
```

但是，由于不允许主机文件包含重复条目，/system/dns/host 元素将必须在配置模板中标记为集合而不是列表，因为集合不允许重复值。为了避免示例中的列表值重复，配置模板作者将使用“主键”选项。

序列合并中的主键选项

当在附加模式下对集合进行操作时，较具体范围中的新值附加到不太具体范围中的值，重复值根据新值在较具体范围中的位置与结果序列中的结果值合并。

这对已合并序列值的影响取决于序列中包含了何种数据：

- 对于序列中作为标量的元素，将使用较具体范围中的值。换句话说，实例层级的值将替换组层级的值。
- 对于作为命名空间序列的元素，将在匹配主要字段的基础上通过应用为该元素指定的合并模式(在该示例中为附加)来获取值。

为了避免 /system/dns/host/.ip 值重复，配置模板作者将使用 CML primary-key 选项。设置此选项后，ACM 会将具有相同 /system/dns/host/.ip 值的条目视为相同，并合并其内容。

在上述示例中，推送操作后服务器上配置文件的最终结果将为：

```
127.0.0.1 localhost mymachine.mydomain.net mymachine
10.10.10.100 mailserver
10.10.10.10 loghost
```

> **备注:**
> 由于集合可以不包含主键，因此，如果序列中有标量，则所有标量值的聚合将用作主键。如果没有标量，则第一个序列中的所有值的聚合将用作主键。尽管这是一种估计操作，但大多数情况下，值可以有效合并。为了确保将正确值用作主键，建议您始终显式设置序列中的主键。

序列前置

当对序列使用前置列表合并模式(CML 标记为“sequence-prepend”)时，较一般范围中的值前置于较具体范围中的值(放在其前面)。

例如，如果在组层级(外部)定义了 etc/hosts 文件中的序列，如下所示：

```
/system/dns/host/1/ip 127.0.0.1
/system/dns/host/1/hostnames/1 localhost
/system/dns/host/1/hostnames/2 mymachine
/system/dns/host/2/ip 10.10.10.10
/system/dns/host/2/hostnames/1 loghost
```

在设备范围(内部)定义了相同的序列，如下所示：

```
/system/dns/host/1/ip 127.0.0.1
/system/dns/host/1/hostnames/1 localhost
/system/dns/host/1/hostnames/2 mymachine.mydomain.net
/system/dns/host/2/ip 10.10.10.100
/system/dns/host/2/hostnames/1 mailserver
```

如果 /system/dns/host 元素是在配置模板中设置了 sequence-prepend 标记的集合，则推送操作后服务器上配置文件的最终结果将为：

```
10.10.10.10 loghost
127.0.0.1 mymachine localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

CML 语法

下表描述了用于阐明多种 CML 标记类型的 CML 语法。

CML 语法

CML 标记/元素	描述
replace-tag	"@" source [";" [type] [";" [range] *option]] "@"
data-definition-tag	"@~" source CRLF *def-line "@"
conditional-tag	"@" [group-level] "?" source [";" [type] [";" [range] *option]] "@"
loop-tag	“@" [group-level] "*" source [";" [type] [";" [range] *option]] "@"
loop-target-tag	"@.@"

CML 语法(续)

CML 标记/元素	描述
block-tag	"@" [group-level] "[" *option "@"
block-termination-tag	"@" [group-level] "]@"
line-continuation-tag	"@\"
instruction-tag	"@!"*option "@"
single-line-comment	"@#" [string CRLF]
multi-line-comment	"@##" *[string / CRLF] "#@"
def-line	type-line / range-line / option-line / printable-line / desc-line
type-line	"type" WSP "=" WSP type-elem CRLF
range-line	"range" WSP "=" WSP range CRLF
option-line	option-elem CRLF
printable-line	"printable" WSP "=" WSP string CRLF
desc-line	"description" WSP "=" *[WSP string CRLF]
group-level	int
source	absolute-path / relative-path / local-path
absolute-path	"/" path-component* name
relative-path	[path-component*] name
path-component	(name / sequence-id) "/"
sequence-id	int
local-path	"." name
name	string
type	sequence / type-elem
sequence	[order "-"] type-elem "-" sequence-elem
sequence-elem	"set" / "list"
type-elem	"int" / "string" / "ip" / "port" / "file" / etc...
order	ordered" / "unordered"
range	and-range *["," and-range]

CML 语法(续)

CML 标记/元素	描述
and-range	range-elem *["&" range-elem]
range-elem	numeric-range / string range
numeric-range	gt-range / ge-range / lt-range / le-range / eq-range
string range	string-literal / regular-exp
gr-range	int ">"
ge-range	int ">="
lt-range	">" int
le-range	">=" int
eq-range	"=" int
string-literal	<"> string <">
regular-exp	"r" <"> string <">
option	";" option-elem
option-elem	option-name / option-nv
option-nv	option-nv
option-name	string
option-value	string

应用程序部署

- **应用程序专家**清楚地了解其多层应用程序的工作方式，以及应该如何部署应用程序的各个部分 (例如，代码和脚本) 才能获得最佳结果。
- **操作专家**了解数据中心的硬件，以及如何使用该硬件最有效地运行多层应用程序。
- **环境所有者**负责特定部署环境 (例如: 开发、QA、用户验收测试或生产) 中的硬件和软件。
- **部署专家**负责在相关生命周期将应用程序部署到环境中。
- **应用程序部署管理员**负责指定可用于部署应用程序的环境、层、生命周期、脚本和代码组件源类型。

以下利益相关人也可从了解应用程序部署的工作方式中获益:

- **SA 管理员**负责所有的 HPE Server Automation 管理任务。他们控制着每个用户角色的特权和权限，并决定由 SA 管理的服务器。他们还负责安装和更新 SA。
- **开发团队**成员和经理负责设计和实现最终部署在生产环境中的应用程序。
- **QA 团队**成员和经理必须了解应用程序如何完成生命周期以及 QA 环境将受到哪些影响。他们可以使用应用程序部署部署和测试应用程序。

先决条件

应用程序部署是 Server Automation (SA) 的一部分。有关 SA 先决条件的信息，请参考《SA 安装指南》。

要执行本指南中所述的各种操作，您必须具有适当的权限。有关详细信息，请参见设置权限 (第 546 页)。

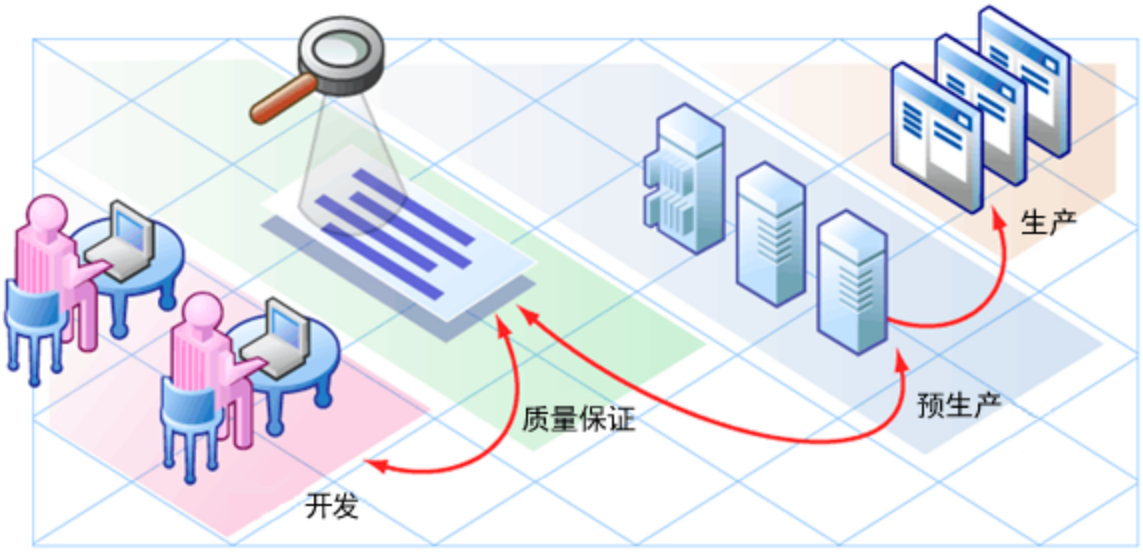
应用程序部署用户界面需要 Adobe Flash Player 版本 9.0 (或更高版本)。

概述

自动化的应用程序部署为何至关重要

创建、部署和测试自定义应用程序形成了一个错综复杂的过程，旨在保证软件的整个开发生命周期。这一生命周期为涉及的各项活动提供了大致结构。以下是一个简单的软件开发生命周期示例：

典型的软件开发生命周期



在此软件开发生命周期中，应用程序将从开发团队转到质量保证 (QA) 团队。QA 团队负责测试应用程序，但 QA 与开发团队之间通常会经历几个测试和修复周期。QA 团队批准应用程序后，即可部署 (无论是首次部署还是进行升级)应用程序，验证应用程序能否在预生产环境中正常工作 (这与生产环境非常类似)。此部署可用于执行性能测试。只有在测试成功之后，应用程序才能进入生产阶段。

如果手动执行，这一过程既非常耗时又容易出错。挑战源自分散的人员和流程。例如，应用程序可能需要分布在各个地域的开发团队提供组件。QA 团队可能负责控制 QA 环境，而系统管理员则负责预生产和生产环境。这就需要大量的沟通和协调。

还有一些其他挑战：

- 复杂的环境使流程变得十分复杂。
- 开发团队通常无法一致地描述部署发行版所需执行的步骤。
- QA 团队必须拥有创建环境所需的机器。然后，他们必须创建并反复清理这些环境。
- 甚至，他们还必须执行多次部署，才能使应用程序达到最新的生产水平。
- 由于 QA 环境和预生产环境中使用的流程不一致，因此在 QA 环境中正常运行的产品在预生产环境中可能会出现故障。
- 预生产环境必须精确地模拟生产环境。
- 每次代码发生更改时，软件工程师均必须重新打包代码。
- 必须有人负责生产机器并了解需要更新的内容。
- 发布经理和团队没有一个专门的场所来交流发行版的当前状态。
- 由于缺乏审核、报告和度量，因此组织不知道哪些因素需要修复，因而无法改进流程。
- 开发团队与运营团队的需求发生冲突，导致了一系列安全问题。

HPE Server Automation 中的应用程序部署可通过提供单点访问 (每个涉及的人员都可通过此访问点查看或输入与他们或他们的角色有关的数据) 来减少部署应用程序所需的复杂通信过程。

此外，应用程序部署还可与其他 HPE 软件技术集成，确保应用程序成功进入生产阶段。

应用程序体系结构

应用程序由各种各样的组件组成，其中包括:

- 代码组件 (例如:HTML 文件、JAR 文件、WAR 文件、.NET 配置集或数据库)
- 脚本
- 配置文件
- SA 应用程序配置
- SA 软件策略
- SA 库中的程序包
- HPE Operations Orchestration (OO) 流
- Windows 注册表设置

这些组件将部署到适当的层，例如，Web 服务器、应用程序服务器和数据库。

简单的双层应用程序

单个应用程序可以具有多个发行版。反过来，一个发行版通常具有多个版本。版本是发行版在特定时间点的不变“快照”。将应用程序部署到环境时，需要部署特定版本。

指定包含在发行版中的组件时，您可以为每个组件同时指定回滚和取消部署指令。

如果启用回滚，应用程序部署将在部署之前执行备份。如果部署期间发生故障，可使用回滚指令将目标恢复到其先前状态。在这种情况下，应用程序部署将追溯其步骤并根据回滚指令撤消每个部署步骤。

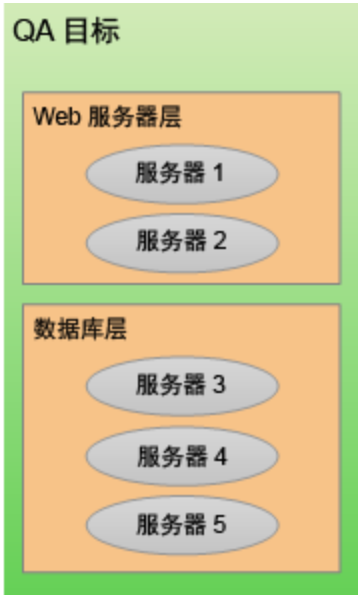
如果已启用取消部署，则可使用取消部署指令卸载应用程序。

您可以创建应用程序组，帮助您以最便捷的方式组织应用程序。这是一个完全出于为用户方便而提供的组织工具；它不影响应用程序的部署。

操作体系结构

从操作角度来说，您已在目标上部署应用程序。每个目标通常适合支持一个或多个应用程序，具体取决于其结构。目标由许多层构成，例如，应用程序服务器层或数据库层。每个目标与一个环境关联，例如 QA 环境或生产环境。创建环境之后，您可以查看环境如何映射到软件开发生命周期。

双层目标



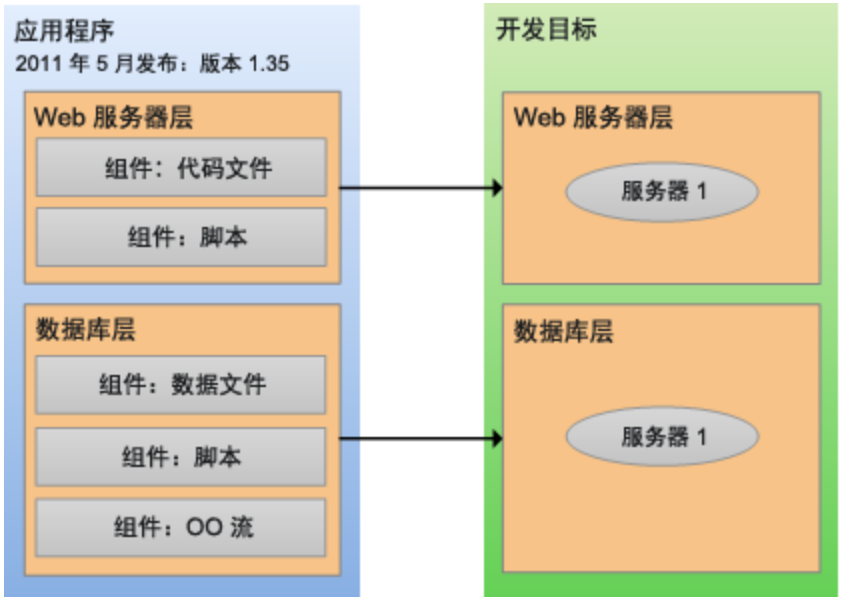
应用程序体系结构与操作体系结构如何协调一致

层是用于将应用程序连接到目标的机制。应用程序通过将属于每层的组件映射到属于同一层的目标服务器进行部署。无论目标的大小或组成如何，应用程序层总是保持不变。

请思考以下示例:一个简单的应用程序 (也许是一个修补程序) 如何从开发环境转移到 QA 环境、最终进入生产环境。

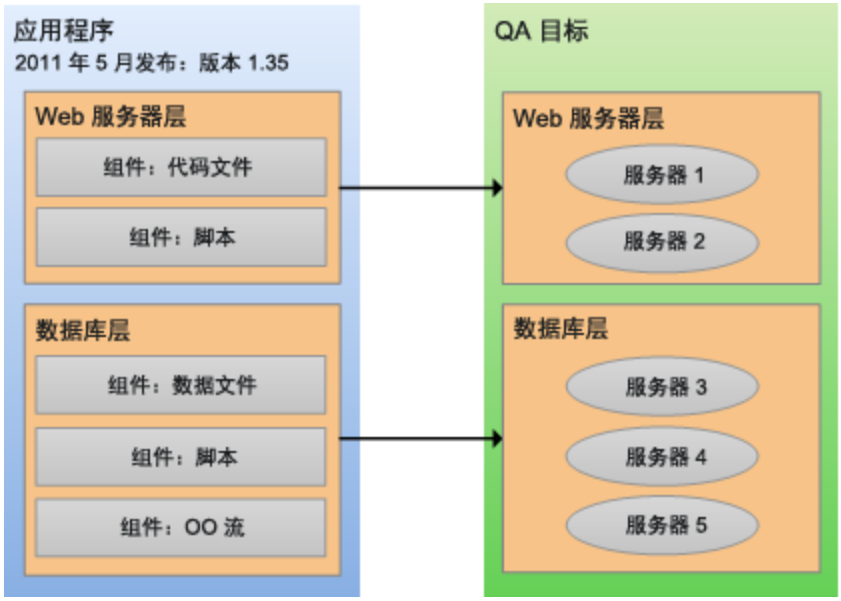
在开发阶段中，应用程序总会不断更改，如此众多的版本将部署到小型测试环境中。在此示例中，同一台服务器同时运行 Web 服务器和数据库。

部署到小型开发目标环境



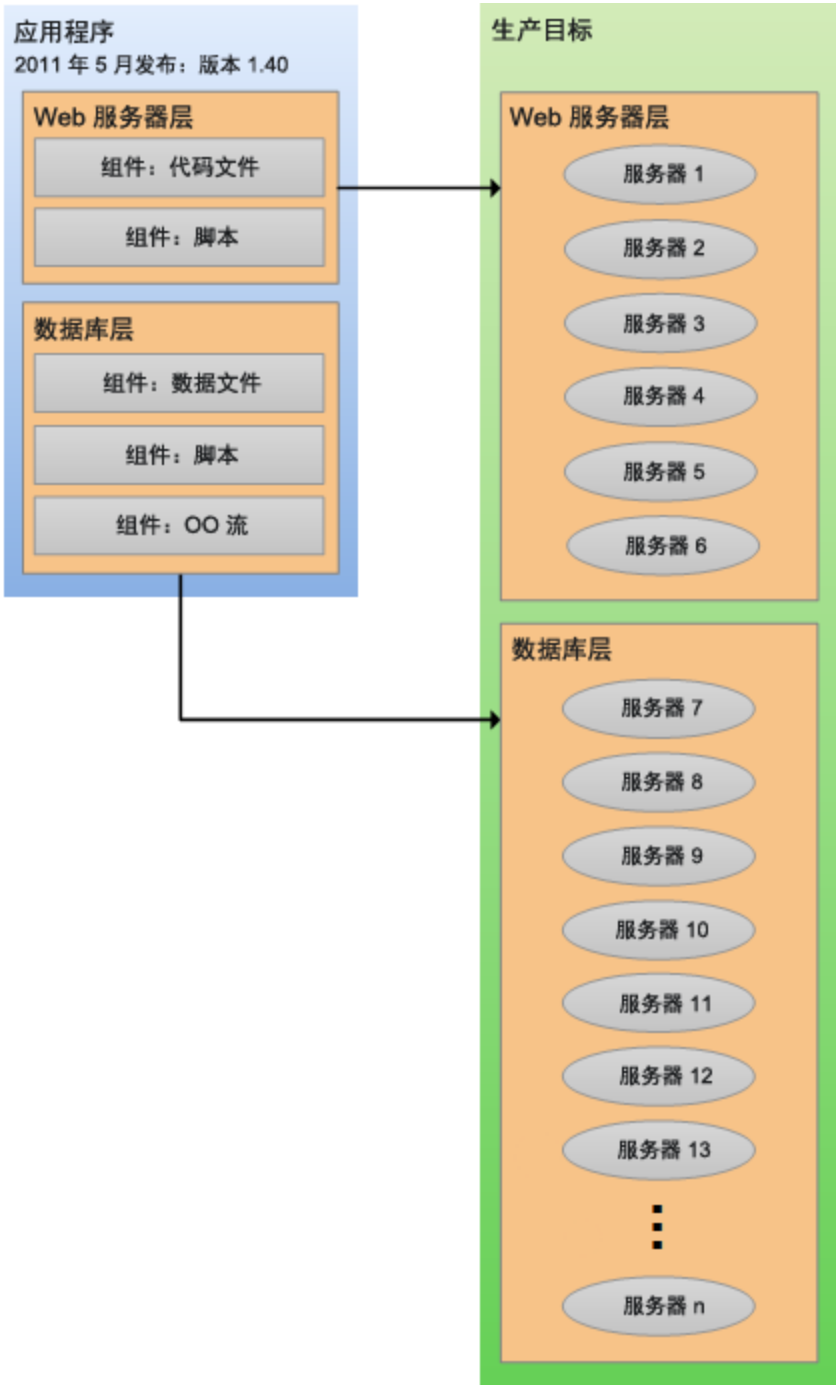
当应用程序提升到 QA 环境中之后，迭代次数将会减少。该环境规模略大一些。在 QA 阶段中，当您检测和解决问题时，可能会将应用程序同时部署在开发环境和 QA 环境中。

部署到略大一些的 QA 目标环境



最终，当应用程序变得稳定并经过全面测试后，将提升到生产环境。这一环境通常更大，如上图所示。

部署到生产环境



请注意，在此示例中，生产数据库托管在服务器群集上，例如 Oracle Real Application Cluster (RAC)。

应用程序如何完成软件生命周期

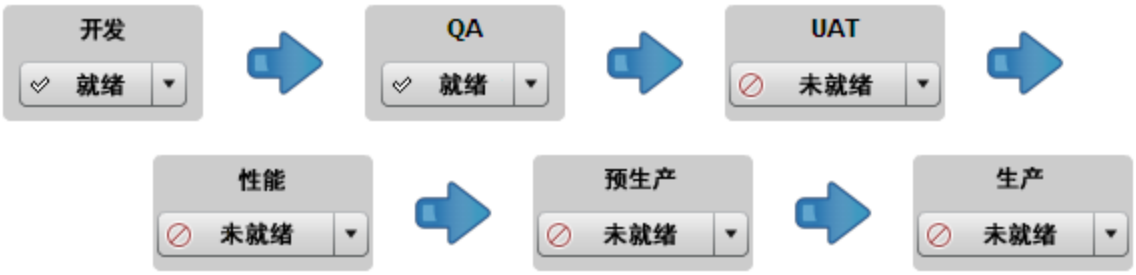
应用程序部署支持您创建自定义生命周期，极为精细地控制应用程序如何从开发阶段完全部署到生产环境。

生命周期由一系列环境组成。每个环境均包含一组服务器。您可以出于不同的目的使用不同的生命周期。例如，对于非常小的应用程序或次要修补程序，可以使用简单的生命周期，如下所示：

修补程序生命周期

对于较为复杂的应用程序 (例如，新产品或重大升级)，可以使用较为复杂的生命周期，如下所示：

完整的生命周期



通过使用生命周期，应用程序部署有助于确保应用程序开发与操作之间的无缝过渡。

在生命周期的最初阶段，应用程序通常会经常更改。例如，您可以在每晚构建后创建一个新版本，然后将其部署到开发环境的服务器中。

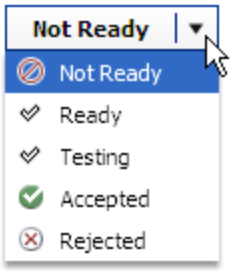
当应用程序变得更稳定后，即可准备进行正式的 QA 测试。此时，您可以每周在 QA 环境中的每台服务器上安装最新内部版本两次。

随着应用程序逐渐完成整个生命周期，更改和所需的部署也变得越来越少。最终，仅需部署少数几个候选发行版，以便进行用户验收测试 (UAT) 和性能测试。

成功通过 UAT 并表现出可接受的性能后，候选发行版即可部署到预生产环境，最终部署到生产环境。

SA 将根据贵公司标准确定不同生命周期中涉及的环境。

环境状态设置



每个环境都有特定的状态。应用程序部署不允许部署到处于“尚未就绪”状态的环境中 (在这种情况下，无法选择该环境中的目标)。

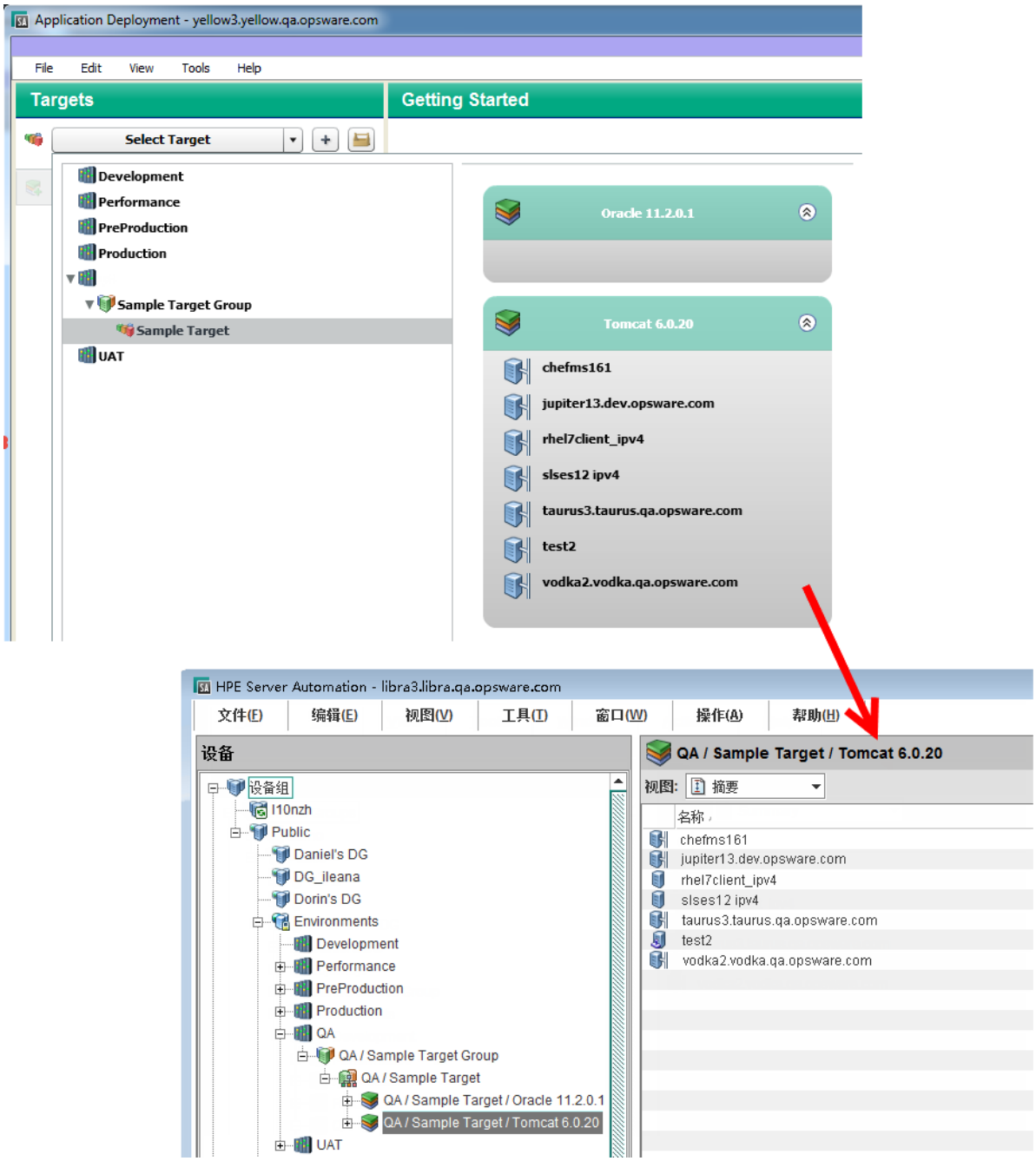
可以使用状态字段控制可部署应用程序的环境。每个环境的初始设置均由应用程序部署管理员设定。如果用户有权部署到特定环境，则可在部署时更改此环境的状态。

应用程序部署管理员将确定哪些用户有权将应用程序部署到哪些环境。有关详细信息，请参见设置权限 (第 546 页)。

环境和设备组

应用程序部署上下文中的环境将在 SA 中镜像为设备组:

应用程序部署和 SA 视图



应用程序部署环境位于“设备”屏幕上的“环境”设备组下。这些设备组为只读 (用户不能修改)，由应用程序部署进行管理。

“环境”下的每个设备组都有一个特殊图标，表示该设备组属于哪种类型的应用程序部署对象：

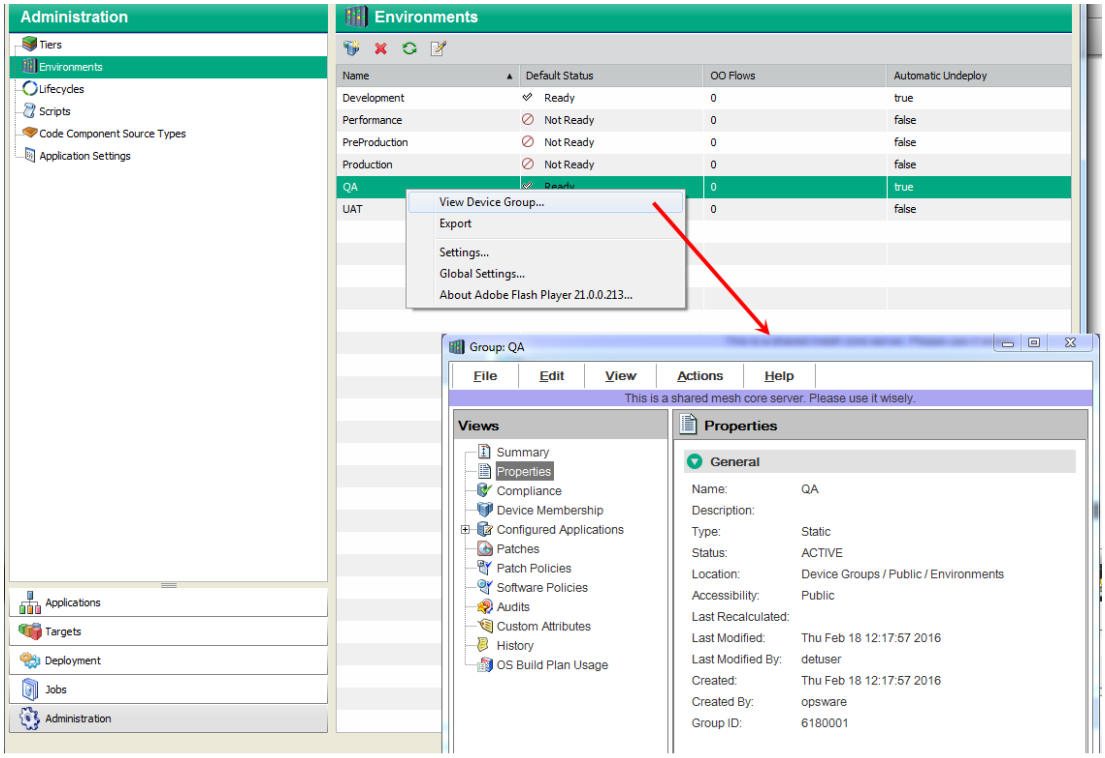
- 环境
- 目标
- 层

目标和层设备组名称是路径，分别指示其父环境和目标。

在应用程序部署用户界面中更改环境或目标 (例如，重命名目标)时，所做的更改也会反映在对应的设备组中。这可能需要一些时间，在此期间，您可能会看到“正在同步”消息。您可能需要在 SA 主用户界面中执行刷新，才能看到所做的更改。

环境所有者或应用程序部署管理员可以使用右键快捷菜单打开“设备组”窗口或强制立即同步，如下所示 (请参见管理环境 (第 563 页))。

环境的设备组窗口



示例：准备环境和部署应用程序

假设您要开设一家网店，您希望尽快向公众开放此站点。负责处理订单的应用程序仍处于开发阶段，但很快可以转到 QA 阶段。

应用程序包括两层: Apache 2.2.9 Web 服务器和 Oracle 11g 数据库。它由存储在源代码控制系统中的多个组件构成。

以下是您团队使用应用程序部署将新应用程序转到 QA 环境进行测试时所需执行的步骤:

1. 应用程序部署管理员设置所需的权限。例如:
   - 应用程序专家需要权限来编辑应用程序并将其部署到开发环境。
   - 部署专家需要权限来查看 (但不能编辑) 应用程序并将其部署到 QA 环境。
2. 应用程序专家创建相关组件并将其分配给 Web 服务器层和数据库层，以此定义应用程序体系结构。
3. 然后，QA 环境所有者检查应用程序体系结构，并准备此应用程序所需的 QA 环境。
   a. 首先，他将创建一个名为 Store Orders 的目标。该目标必须包含应用程序体系结构中指定的相同层。
   b. 接下来，他指定属于每层的服务器，这些服务器必须已由 SA 管理。现在，该网店的 QA 环境已准备就绪。
4. 当软件工程团队确定应用程序的第一个版本已准备好进行测试后，他们将创建一个版本为部署作准备。接下来，他们通知 QA 应用程序已准备好进行测试。
5. 然后，部署专家将应用程序部署到 QA 环境，QA 工程师开始测试。

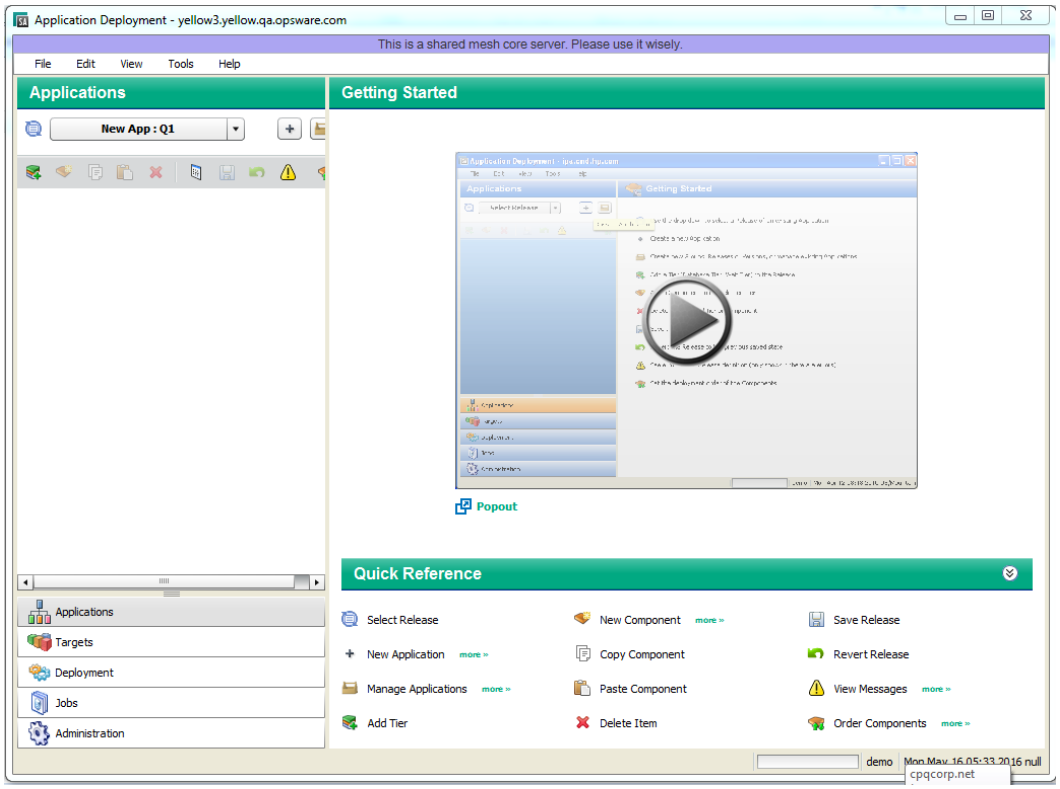
您希望该团队顺利完成测试过程，并最大程度地减少开发与 QA 阶段之间的迭代次数。部署自动化可以减少迭代次数。

一旦应用程序完成 QA 周期，将转到并经过预生产阶段，最后转到生产阶段，届时您即可开店营业。

关于应用程序部署用户界面

首次打开应用程序部署工具以及每次创建新应用程序时，用户界面如下所示:

应用程序部署界面 - 初始视图



在此初始视图中，您可使用两个工具来了解应用程序部署用户界面的详细信息：

- Getting Started 区域包含一个简短的演示视频，其中演示了此屏幕的功能。

  单击 按钮可播放该视频。要在单独的窗口中播放该视频，请单击 Popout。

- Quick Reference 区域列出了此屏幕上的工具栏按钮功能。

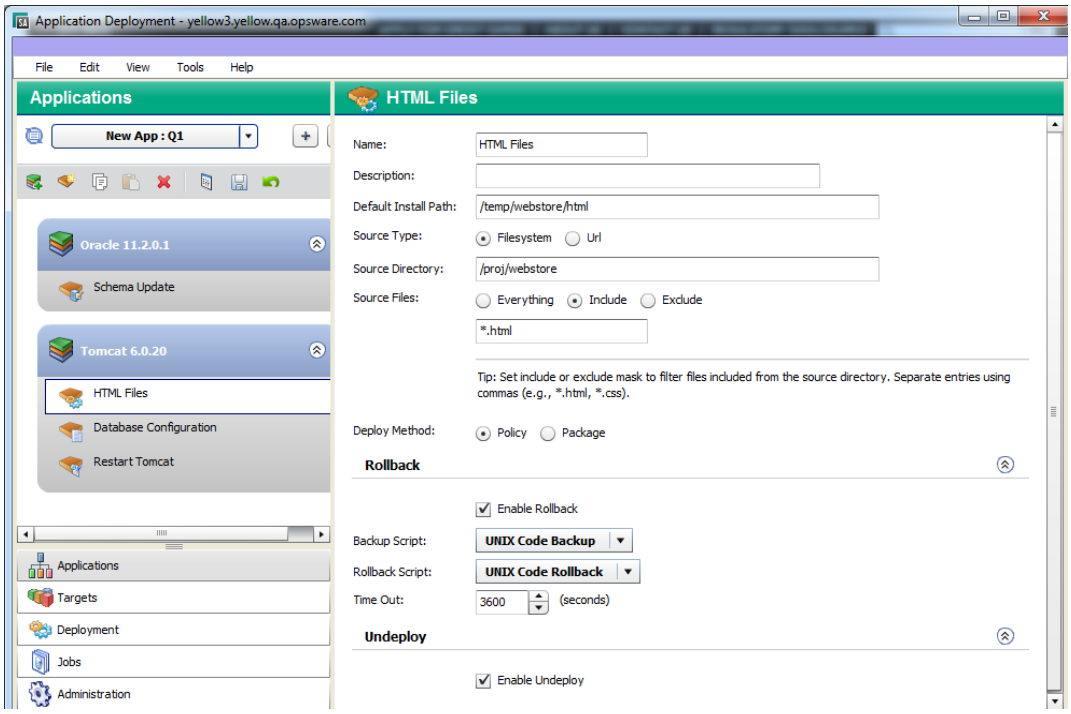
  将鼠标悬停在每个功能上方可查看相应的工具提示。单击 more>> 链接，可在单独的浏览器窗口中打开相关的联机帮助主题。

左下角的按钮可打开应用程序部署用户界面的不同功能区域。突出显示的按钮指示当前的工作区域。您转到其他区域之前，必须先保存或还原所做的任何更改。

> **备注：** 应用程序部署附带了一个示例应用程序和示例目标，旨在帮助您快速入门。

将层添加到应用程序并在层中创建组件之后，该界面类似如下：

应用程序部署界面 - 双层应用程序



左窗格显示结构视图并提供导航控件。

右窗格显示在左侧所选项的内容。在此示例中，选择了 Tomcat 6.0.20 层中的 HTML Files 组件。

命名规则

以下规则适用于所有项的 Name 属性:

- 名称不能以空格开头或结尾。
- 名称不能以标点符号开头，即，首个字符必须为字母或数字。
- 名称不能包含以下任何字符:换行符、制表符、斜杠 (/) 或反斜杠 (\)。

应用程序部署用户界面不允许您违反这些命名规则。如果名称不符合这些规则，则 Name 框周围将出现一个红框。

修改文本框中的信息

修改文本框 (在右窗格或对话框中) 中的信息时，请注意以下几点:

- 选中文本框时，该框周围将出现蓝色边框。
- 如果违反了任何命名规则 (第 412 页)，则该文本框周围将出现红框。

可以使用 Ctrl+C、Ctrl+X 和 Ctrl+V 复制、剪切和粘贴任何可编辑的文本框中的信息。您也可以使用 Edit 菜单上的对应项。

保存和还原更改

每次在应用程序部署用户界面中修改项目时，Save 和 Revert 按钮均会变为活动状态:

- 单击 **Save** 按钮  可保存对应用程序部署数据库所做的更改。
- 单击 **Revert** 按钮  可放弃所做的更改，并从数据库重新加载最近保存的值。

必须先保存或还原所做的更改，才能转到另一个屏幕 (例如，从 Applications 屏幕转到 Targets 屏幕)。

管理应用程序和目标

Applications 屏幕和 Target 屏幕中提供的“管理器”可用于快速评估应用程序和目标的当前状态并执行各种任务。例如，在 Application Manager 中，您可以新建发行版或重命名应用程序。

在这些管理器中，您可以选择多个项，然后对所有这些项执行相同的操作。例如，您可以选择几个应用程序，然后将其删除。

要打开管理器，请在左上角的下拉列表右侧单击  按钮。

有关详细信息，请参见管理应用程序 (第 434 页)和管理目标 (第 492 页)。

验证消息

每次出现验证错误时，工具栏上均会显示“警告”符号  。如果某一项尚未完全配置，也会显示该符号。

单击“警告”符号  ，可了解检测到的特定错误的详细信息。

例如，在创建新应用程序之后、但在添加任何组件之前，将会报告以下验证错误:

“No components in release”

如果您尝试执行的操作失败，也会显示验证错误消息。例如，如果您尝试为组件不完整的应用程序创建版本，将会显示类似如下的消息。

“Cannot create version from this release.Please correct errors below:Script Component ‘Sample Script Component’:No content.

Code Component ‘Sample Code component’:Source Directory must be set.

Code Component ‘Sample Code component’:Default Install Path must be set."

快速入门

借助应用程序部署工具，应用程序开发人员和运营团队可以无缝地协同工作，管理任务关键型业务应用程序从工程到生产的提升。您可以使用应用程序部署定义和部署复杂的多层 Java EE 和 .NET 应用程序。

要从 SA 客户端运行应用程序部署工具，请选择 **Tools > Application Deployment**。

定义应用程序时，您可以:

- 快速创建应用程序定义并使用多个同步版本。
- 使用层定义应用程序的结构。
- 使用软件策略确保应用程序所需的中间件已就绪，或随应用程序一起部署。

使用新的组件类型、软件库对象以及 (可选) HPE Operations Orchestration (OO) 流控制应用程序内容。

指定 OO 流和其他组件类型的应用程序默认参数。如果适合，可以在部署时调整这些参数，使其与不同的环境匹配，例如，对于 QA 与生产环境之间的应用程序，合适的数据库连接设置可能有所不同。

定义目标时，您可以:

定义通常旨在共同作用以支持特定应用程序的服务器、实例和数据库的集合。

使用在应用程序中定义的层的相同集合对目标进行结构化。

使用软件策略确保应用程序所需的中间件已就绪，或随应用程序一起部署。

使用托管服务器、实例和数据库填充目标层。

将 OO 流与目标层关联。

> 此快速入门说明如何创建和部署简单应用程序:

概述 (第 415 页)

快速入门 (第 414 页)

快速入门 (第 414 页)

快速入门 (第 414 页)

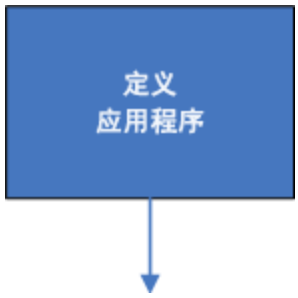
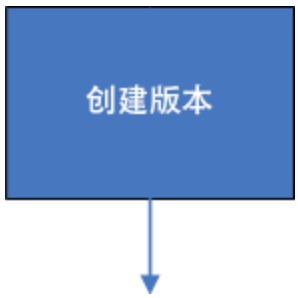
快速入门 (第 414 页)

此快速入门假定您已安装 Server Automation (SA) 并可运行 SA 客户端。有关详细信息，请参见《SA 10.50 安装指南》。

概述

应用程序部署可自动执行部署应用程序的过程。此快速入门教程提供了用户快速入门所需的信息。

基本步骤如下所示:

	应用程序具有一个或多个发行版。 每个发行版组织在层中，例如，应用程序服务器层、Web 服务器层和数据库层。 每层包含一个或多个组件。这些组件表示应用程序的内容并可实现其功能。 请参见步骤 1：定义应用程序 - 层和组件 (第 416 页)。
	创建发行版的版本并部署该版本。 每个版本均是发行版的不变快照。 版本是不变的，这可确保在 QA 环境中测试的组件相同并将其部署到生产环境。 请参见步骤 2：创建要部署的版本 (第 421 页)。

	目标由托管应用程序层的服务器组成，例如，数据库层、应用程序服务器层或 Web 服务器层。 特定层可能在多台服务器上运行。 请参见步骤 3：定义目标 (第 422 页)。
	每个应用程序层中的组件都将部署到环境中具有这些层的选定目标。 请参见步骤 4：将版本部署到目标 (第 426 页)。

有关基本说明，请单击上面的每个链接。本书的后续章节将提供与这些任务有关的更多详细信息。

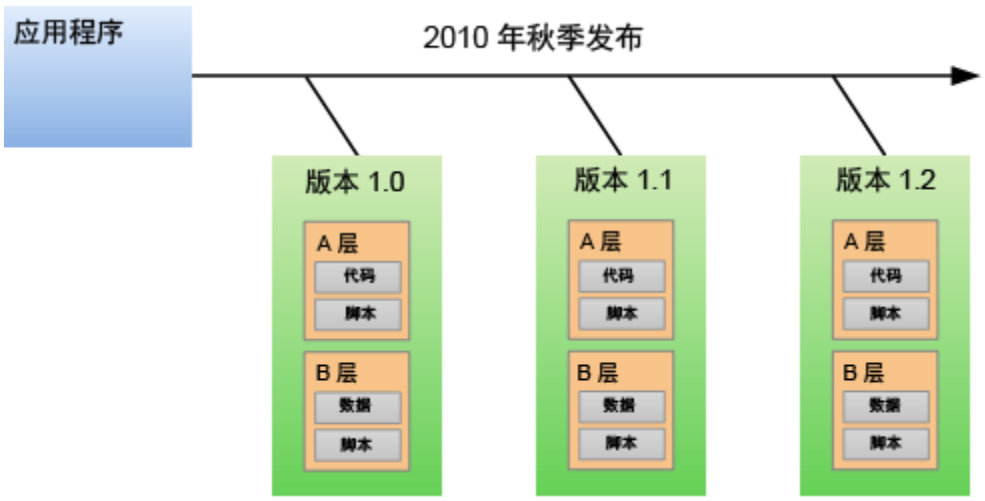
步骤 1：定义应用程序 - 层和组件

应用程序包含一个或多个层。每层包含实现应用程序功能的多个组件。

应用程序以一个或多个发行版的形式存在。每个发行版可以有多个版本。部署应用程序时，创建一个包含各层组件的版本。版本是发行版在特定时间点的不变“快照”。

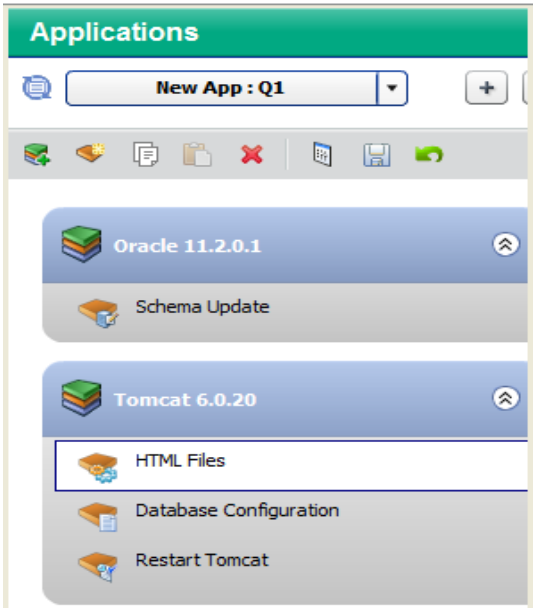
下图显示了具有 Fall 2010 发行版的三个版本的双层应用程序。

如何组织应用程序



例如，以下应用程序具有 Oracle 11.2.0.1 数据库层和 Tomcat 6.0.20 应用程序服务器层。Tomcat 层中的组件是 HTML 文件、配置文件和重新启动脚本。Oracle 层中的组件是数据库脚本。

双层应用程序示例

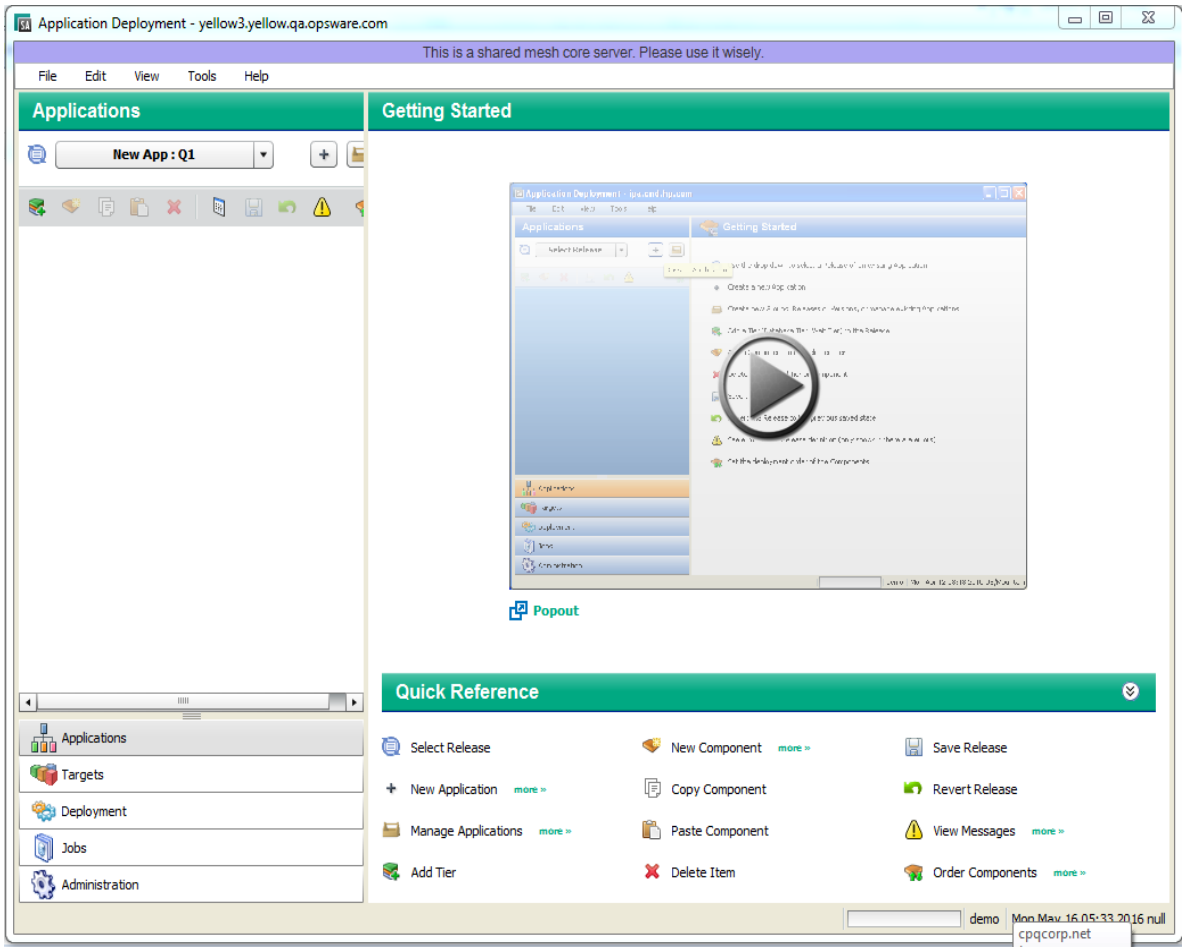


以下说明假定您正在创建一个新应用程序。如果要使用现有应用程序，请参见应用程序概述 (第 428 页)。

在可以创建应用程序之前，SA 管理员必须向您授予访问应用程序部署工具和创建应用程序的权限。请参见设置权限 (第 546 页)。

要启动应用程序部署工具，请执行以下操作:

1. 启动 SA 客户端。有关详细信息，请参见《SA 10.50 用户指南》。
2. 在 SA 客户端中，选择 **Tools > 应用程序部署** 菜单项。将打开界面应用程序部署界面并显示 Applications 屏幕，如下所示:



单击 按钮可启动一个简短的演示视频，其中介绍了 Applications 屏幕的功能。要在单独的窗口中播放该视频，请单击 **Popout**。

Quick Reference 区域列出了此屏幕上的工具栏按钮功能。单击 more>> 链接，可在单独的浏览器窗口中打开相关的联机帮助主题。

要创建和配置新应用程序，请执行以下操作:

1. 转到 Applications 屏幕 (单击左下角的 **Applications**)。
2. 单击 **Create a New Application** 。将打开 Create New Application 对话框。请注意，创建应用程序的功能是 SA 中需要分配给用户所属组的特定操作权限 (请参见设置权限 (第 546 页))。

a. 输入新应用程序的名称。

b. 可选：将新应用程序分配给现有的应用程序组。

c. 可选：输入新应用程序的描述。

d. 在 **Release Name** 字段中，输入对此应用程序使用的当前发行版的名称。

例如，如果您每季度发布一次，则可使用 Q3-2010、Q4-2010、Q1-2011 和 Q2-2011 等作为发行版名称。

e. 选择此发行版的生命周期。该生命周期指定应用程序在从开发到生产的整个过程中经历的阶段。

例如，常见的生命周期是开发 > QA > 预生产 > 生产。这是默认的标准生命周期。

f. 单击 **OK** 保存应用程序信息。

3. 单击 **Add Tier**，可向应用程序添加一层或多层。
   a. 选择要添加的层类型。

   您的软件附带提供了一组标准层。如果所需的层不可用，则SA管理员可能需要添加该层。请参见管理应用程序部署 (第 560 页)。

   b. 单击 **Add**。

   c. 对所需的每个附加层重复步骤 (a) 和 (b)。

4. 要将组件添加到层，请执行以下步骤:
   a. 选择层。

   b. 单击“New Component”按钮，将组件添加到此层。

   c. 输入新组件的名称。

   d. 选择组件类型。有关详细信息，请参见组件 (第 442 页)。

   e. 单击 **OK**。

   f. 输入完全指定新组件的位置和行为所需的信息。

   必须提供的信息因所添加的组件类型而异。某些组件类型需要提供回滚和取消部署场景的指令。有关详细信息，请参见组件类型 (第 443 页)。

   g. 重复上面的步骤 (b) 到 (f)，将更多组件添加到此层。

   h. 单击 **Save Release**，保存所做的更改。

5. 重复上面的步骤 3-4，添加更多层并向每层添加更多组件。

6. 单击 **Edit component deployment order** 。
7. 使用绿色箭头键按正确的部署顺序排列组件。有关详细信息，请参见更改组件的部署顺序 (第 467 页)。

添加所有层和组件后，即已完全定义应用程序。应用程序应类似于下图中所示的示例。

如果工具栏上显示“警告”符号 ，这表明组件未完全定义或定义错误。例如，代码组件的源位置引用的目录可能不存在。
单击 可确定所需的信息或操作。

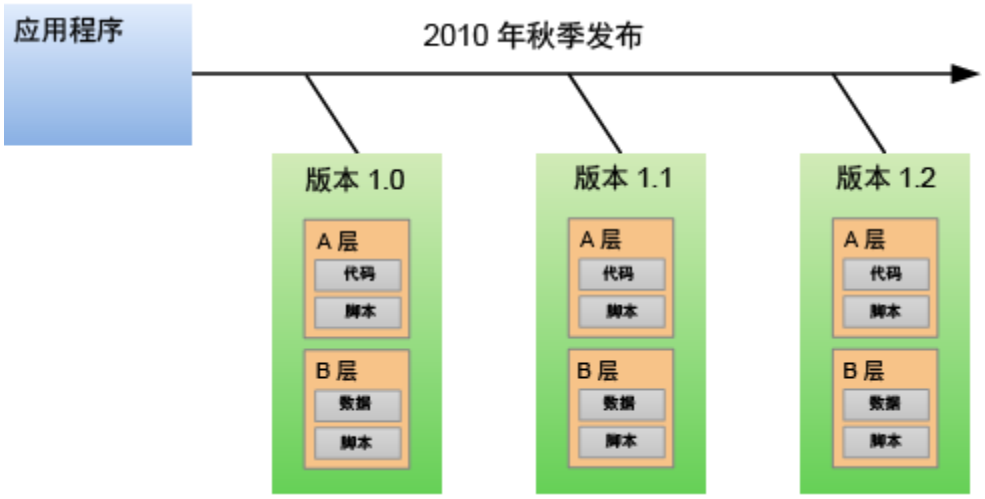
包含七个组件的三层应用程序示例

步骤 2：创建要部署的版本

定义应用程序之后，必须创建应用程序的新版本。然后，将构成该版本的组件部署到目标。

下图显示了为名为 "Fall 2010" 的发行版而开发的应用程序。已构建此应用程序的三个版本：版本 1.0、1.1 和 1.2。可将这三个版本中的任意一个部署到目标。

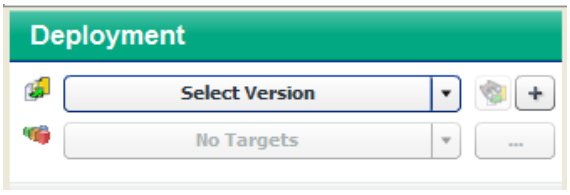
应用程序、发行版和版本



请先确保您已成功创建应用程序的版本，然后再尝试进行部署。

要创建应用程序的新版本，请执行以下操作：

1. 转到 Deployment 屏幕 (单击左下角的 **Deployment**)。
2. 在 Select Version 下拉字段中，导航到您的应用程序。



3. 根据应用程序的相关发行版，单击 **Create New Version** 链接。
4. 在 Create New Version 窗口中，输入版本名称 (或版本号)。

版本名称就是您指定的字符串。如果此发行版的以前版本包含数字，则应用程序部署会自动增加该数字。例如，如果以前版本是 "1"，则 Version 字段中将显示 "2"。如果以前版本是 "V 1.2.1"，则 Version 字段中将显示字符串 "V 1.2.2"。

5. 无论应用程序部署在 Version 字段中输入的内容如何，您均可指定所需的任何版本名称。
6. 可选：在 Description 字段中，输入要为此版本指定的任何其他信息。
7. 可选：如果要查看代码组件中所含文件的列表，请选择 **Show Code Component Changes**。
8. 单击 **Create**。这将通过收集属于应用程序的组件的所有项，创建此应用程序的新版本。
9. 单击 **Close**。应用程序部署工具为准备部署的应用程序创建了新版本。

步骤 3：定义目标

目标是应用程序将部署到的服务器集。每个目标必须包括应用程序所需的层。

三层目标

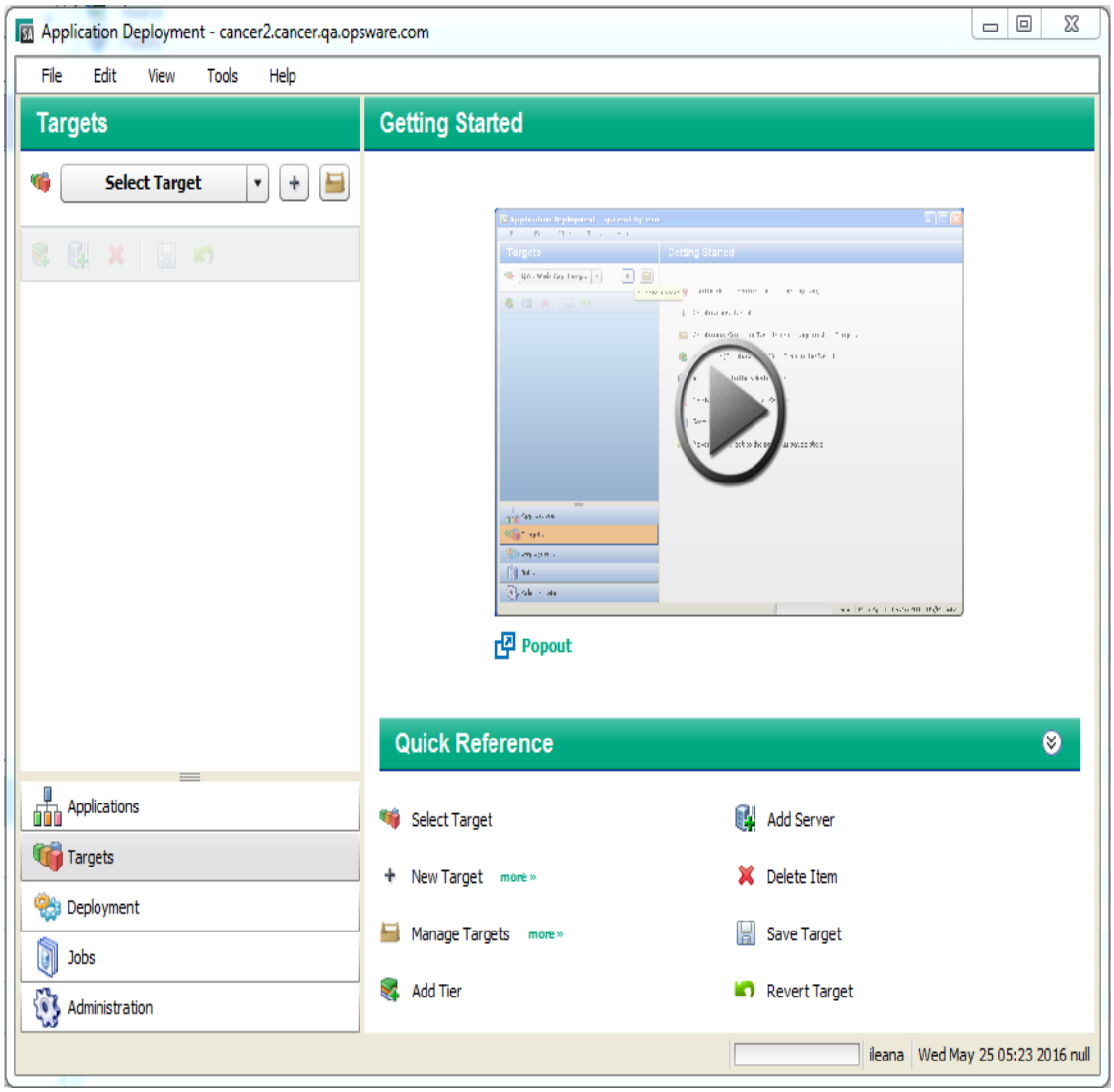
在成功设置目标前，需要两个级别的权限。SA 管理员必须向您授予对相关环境的 Write 权限 (请参见环境权限 (第 559 页))。

您还必须有权使用这些服务器，才可以将服务器添加到层。此权限在 SA 客户端中进行管理 (请参见操作权限 (第 551 页))。

目标在应用程序部署工具中的 Targets 屏幕上定义。

首次访问 Targets 屏幕时，它将如下图中所示:

Targets 屏幕



单击 按钮可启动一个简短的演示视频，其中演示了 Targets 屏幕的功能。要在单独的窗口中播放该视频，请单击 Popout。

Quick Reference 区域列出了此屏幕上的工具栏按钮功能。单击 more>> 链接，可在单独的浏览器窗口中打开相关的联机帮助主题。

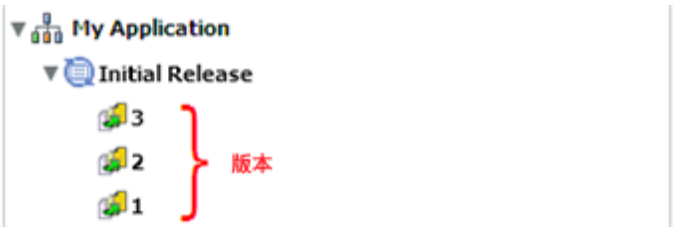
以下说明假定您正在创建一个新目标。这些说明还假定，对于此快速入门，您的目标仅包括服务器。

要创建和配置新目标，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 单击“Create a New Target”按钮: . 将打开 Create New Target 对话框。
3. 输入目标名称。目标名称在环境中必须唯一。
4. 在 Location 下拉字段中，选择将包含此目标的环境。

   可通过两种方法使用层填充目标: 您可以手动添加层 (如步骤 6 中所述)，也可以自动创建层。

   要在目标中自动创建一组用于现有应用程序的相同层，请选择“Use Application's Tiers”，然后选择应用程序的版本。

   > **备注:** 之前必须已创建应用程序的特定发行版的具体版本，该版本才会显示在 Select Version 下拉列表中。

5. 单击 **OK**。
6. 请按照以下步骤添加应用程序所需的每个层:
   a. 单击 **Add Tier** 按钮。
   b. 选择要添加的层类型。您必须创建应用程序所使用的相同层集 (或超集)。

      您的软件附带提供了一组标准层。如果所需的层不可用，则应用程序部署管理员可能需要添加该层。

   c. 单击 **Add** 按钮。

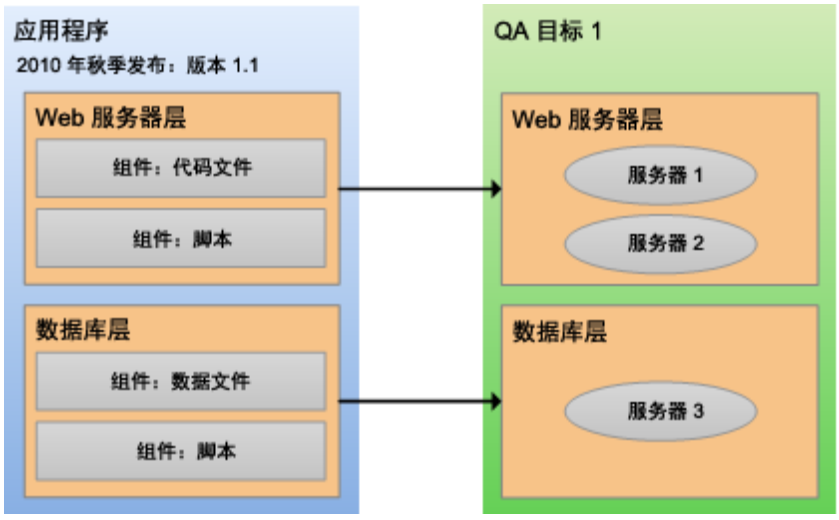
7. 对于在步骤 6 中添加的每个层，按照以下步骤指定将运行此层的服务器：
   a. 选择要使用的层。
   b. 选择 **Add Server** 按钮，将服务器添加到此层。

      可用服务器的列表按平台进行筛选。请确保为层选择合适的服务器。例如：如果该层的组件专门用于 Red Hat Enterprise Linux 5，则仅应选择 Red Hat Enterprise Linux 5 服务器。
   c. 在 Add Servers to Target 对话框中，找到并选择要添加的托管服务器。如果要设置多服务器层，请使用 CTRL 键选择多台服务器。
   d. 单击 **OK**。
8. 重复步骤 7，将服务器添加到其余的所有层中。
9. 选择 **Save Target** 按钮 保存所做的更改。

步骤 4：将版本部署到目标

您可以将应用程序的版本部署到目标。这会将应用程序文件复制到目标服务器并执行应用程序附带的任何脚本。

将应用程序部署到目标



创建版本之后，您可以将该版本部署到目标。

要部署版本，请执行以下操作:

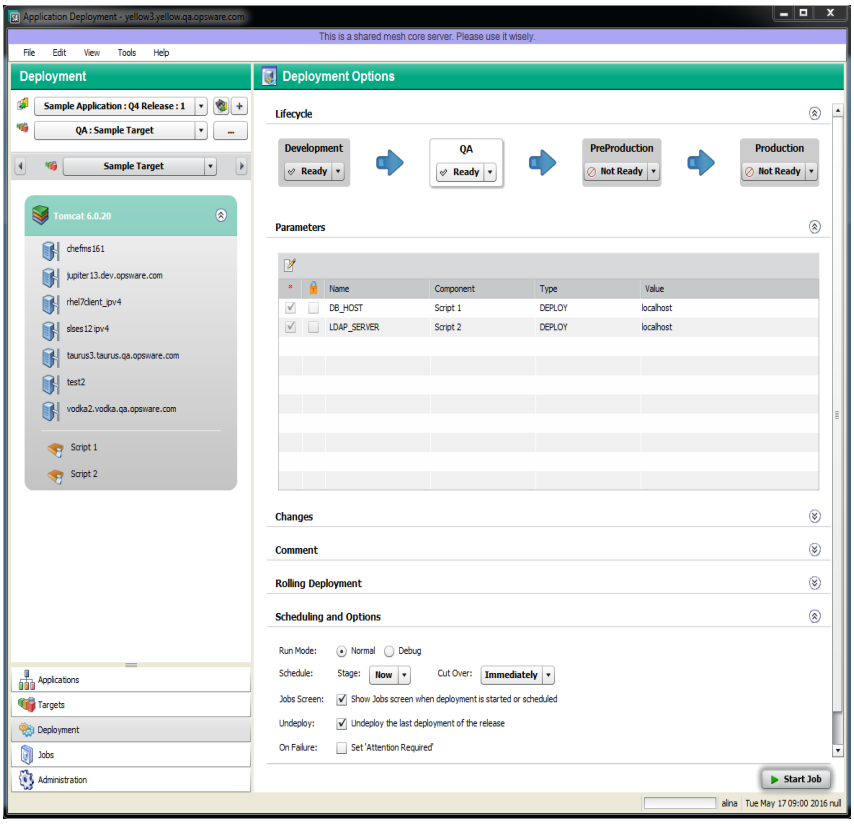
1. 转到 Deployment 屏幕 (单击左下角的 **Deployment**)。
2. 从 Select Version 下拉列表中，选择您创建的版本。如果您刚才创建了此版本，则它已处于选中状态。
3. 从 Select Target 下拉列表中，选择之前创建的目标。这将指定要部署应用程序的目标服务器。

   如果在该列表中看不到您的目标，请确保满足以下条件:
   - 该目标的层匹配应用程序层或是其超集。
   - 您具有此环境的 Deploy 权限。
   - 每层至少包含一台服务器。
   - 此发行版的生命周期包括目标所在的环境。
   - 在生命周期中，该环境没有标记为 Not Ready。

对于其他目标选项，请单击 Select Target 下拉列表右侧的  按钮。有关这些选项的详细信息，请参见部署应用程序 (第 503 页)。

选择版本和目标之后，Deployment 屏幕类似如下: 左侧面板用来预览将部署的内容，而右侧面板显示部署作业设置。

4. 可选：在右侧面板中修改部署作业设置。有关详细信息，请参见部署应用程序 (第 503 页)。



5. 单击 **Start Job**。这会启动部署作业，该作业会将应用程序组件复制到目标服务器，并运行任何必要的代码和脚本。

   可以在 Jobs 屏幕上查看部署的状态。

应用程序概述

应用程序是应用程序部署中的基本构建块之一。它们具有两个主要特性：内容和结构。应用程序的内容决定了其功能。应用程序的结构反映了其组织。在应用程序部署的上下文中，结构有助于将应用程序与结构相同的适当应用程序部署目标关联。

此外，结构还指定组件的部署顺序。在应用程序部署中，组件可实现应用程序的内容。以下类型的组件可用：

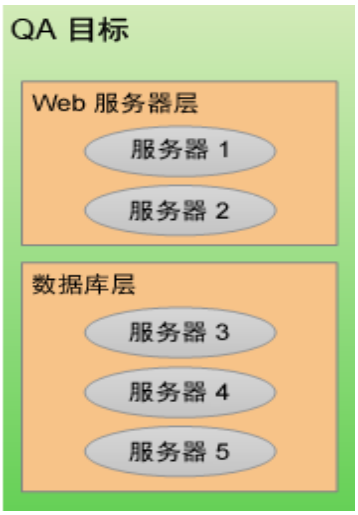
代码组件 (例如：HTML 文件、JAR 文件、WAR 文件、.NET 配置集或数据库)

- 临时脚本
- 配置文件

- 应用程序配置
- 软件策略
- 程序包
- HPE Operations Orchestration (OO) 流
- 注册表组件 (仅限 Windows 目标)

层是用来表示应用程序结构的容器。层包含应用程序的特定部分所需的组件。层的示例包括 Web 服务器、应用程序服务器和数据库。目标也组织在层中。部署应用程序时，每个应用程序层的组件将部署到相应层中的目标服务器。

双层应用程序



单个应用程序可以具有多个发行版。反过来，一个发行版通常具有多个版本。版本是发行版在特定时间点的不变“快照”。将应用程序部署到环境时，需要部署特定版本。

有两种类型的发行版:完全发行版和增量发行版，不同之处在于代码组件的处理。对于完全发行版，属于所有代码组件的所有文件都包括在内。对于增量发行版，只有自发行版的特定版本创建以来添加或修改的这些文件才包括在内。在部署过程中，将在目标服务器上删除已删除的文件。

指定包含在发行版中的组件时，您可以为每个组件同时指定回滚和取消部署指令。如果在部署过程中发生失败，将使用回滚指令。在这种情况下，应用程序部署将追溯其步骤并根据您提供的回滚指令撤消每个部署步骤。取消部署指令用于卸载应用程序。

回滚和取消部署的自动化程度取决于组件类型。例如，代码组件使用由应用程序部署管理员维护的标准脚本。相比之下，脚本和 OO 流组件需要提供显式回滚和取消部署指令。

您可以创建应用程序组，帮助您以最便捷的方式组织应用程序。这是一个完全出于为用户方便而提供的组织工具；它不影响应用程序的部署。

定义新应用程序的过程分为三个主要步骤:

- 创建应用程序和发行版
- 指定包含在发行版中的层
- 指定每层的组件

先决条件

在可以定义应用程序之前，必须先满足以下条件:

此应用程序所需的层在应用程序部署用户界面中可用。

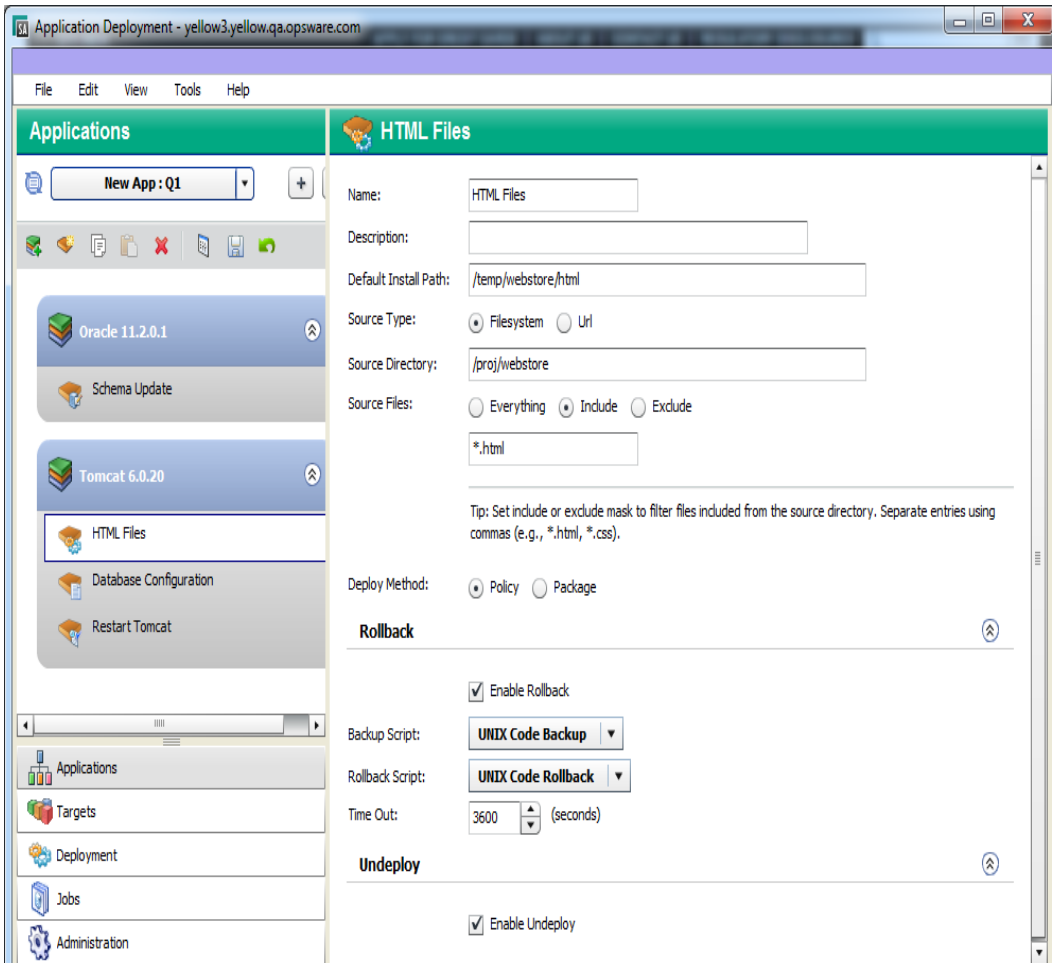
您有权创建应用程序。

有关详细信息，请参见《SA 10.50 管理指南》。

关于 Applications 屏幕

定义应用程序之后，Applications 屏幕类似如下:

包含现有应用程序的 Applications 选项卡



注意事项:

- 应用程序名称必须唯一。
- 您有权查看的所有已定义应用程序和发行版将显示在左上角的下拉列表中。此处选择了 New App:Q1。
- 左窗格显示应用程序的结构。
- 右窗格显示有关在左窗格中选择的项目的信息。您可以选择单个组件或整个层。选择某一项时，该项的背景色将会改变。此处选择了 HTML Files 组件。
- 您可以在层之间拖放组件 (假设平台相同)。
- 您可以更改组件的部署顺序 (请参见更改组件的部署顺序 (第 467 页))。

关于层

层是用来表示应用程序结构的容器。层包含实现应用程序功能的组件。应用程序部署管理员定义了可供您使用的层 (请参见《SA 10.50 管理指南》)。提供了一系列“预置”的常用层。管理员可能需要修改这些层以适应您的环境并添加新层。

要查看简单示例，请参见快速入门 (第 414 页)。

使用应用程序

创建新应用程序

本主题提供有关创建新应用程序的基本说明。假设您具有创建应用程序所需的 HPE Server Automation 权限。有关详细信息，请参见设置权限 (第 546 页)。

要创建和配置新应用程序，请执行以下操作:

1. 转到 Applications 屏幕 (单击左下角的 **Applications**)。
2. 单击 **Create a New Application**。将打开 Create New Application 对话框。请注意，创建应用程序的功能是 SA 中需要分配给用户所属组的特定操作权限 (请参见设置权限 (第 546 页))。
   a. 输入新应用程序的名称 (请参见命名规则 (第 412 页))。
   b. 可选：将新应用程序分配给现有的应用程序组。
   c. 可选：输入新应用程序的描述。
   d. 在 **Release Name** 字段中，输入对此应用程序使用的当前发行版的名称。

      例如，如果您每季度发布一次，则可使用 Q3-2010、Q4-2010、Q1-2011 和 Q2-2011 等作为发行版名称。
   e. 选择此发行版的生命周期。该生命周期指定应用程序在从开发到生产的整个过程中经历的阶段。

例如，常见的生命周期是开发 > QA > 预生产 > 生产。这是默认的标准生命周期。

f. 单击 **OK** 保存应用程序信息。

3. 单击 **Add Tier** 向应用程序添加一层或多层。
   a. 选择要添加的层类型。

      您的软件附带提供了一组标准层。如果所需的层不可用，则应用程序部署管理员可能需要添加该层。请参见《SA 10.50 管理指南》。

   b. 单击 **Add**。
   c. 对所需的每个附加层重复步骤 (a) 和 (b)。
4. 要将组件添加到层，请执行以下步骤:
   a. 选择层。
   b. 单击 **New Component**，将组件添加到此层。
   c. 输入新组件的名称。
   d. 选择组件类型。有关详细信息，请参见组件类型 (第 443 页)。
   e. 单击 **OK**。
   f. 输入完全指定新组件的位置和行为所需的信息。

      必须提供的信息因所添加的组件类型而异。某些组件类型需要提供回滚和取消部署场景的指令。

   g. 重复上面的步骤 (b) 到 (f)，将更多组件添加到此层。
   h. 单击 **Save Release**，保存所做的更改。
5. 重复上面的步骤 3-4，添加更多层并向每层添加更多组件。
6. 单击 **Edit component deployment order**。
7. 使用绿色箭头键按正确的部署顺序排列组件 (请参见更改组件的部署顺序 (第 467 页))。

添加所有层和组件后，即已完全定义应用程序。应用程序应类似于快速入门 (第 414 页)中所示的示例。

如果工具栏上显示“警告”符号 ⚠，这表明组件未完全定义或定义错误。例如，代码组件的源位置引用的目录可能不存在。
单击 ⚠ 图标可确定所需的信息。

管理应用程序

如果您有权执行此操作 (请参见概述 (第 546 页))，则可使用 Manage Applications 工具执行某些任务。此工具支持您创建可部署到目标的版本。它还使您能够指定有关应用程序和发行版的非常精细的信息。

要打开 Manage Applications 工具，请单击 Applications 屏幕上的  按钮。如果您无权管理应用程序 (请参见设置权限 (第 546 页))，则此按钮不可用。

Manage Applications 工具包含一个表，其中列出了此处所示的项目：

应用程序组 (可以嵌套)

应用程序

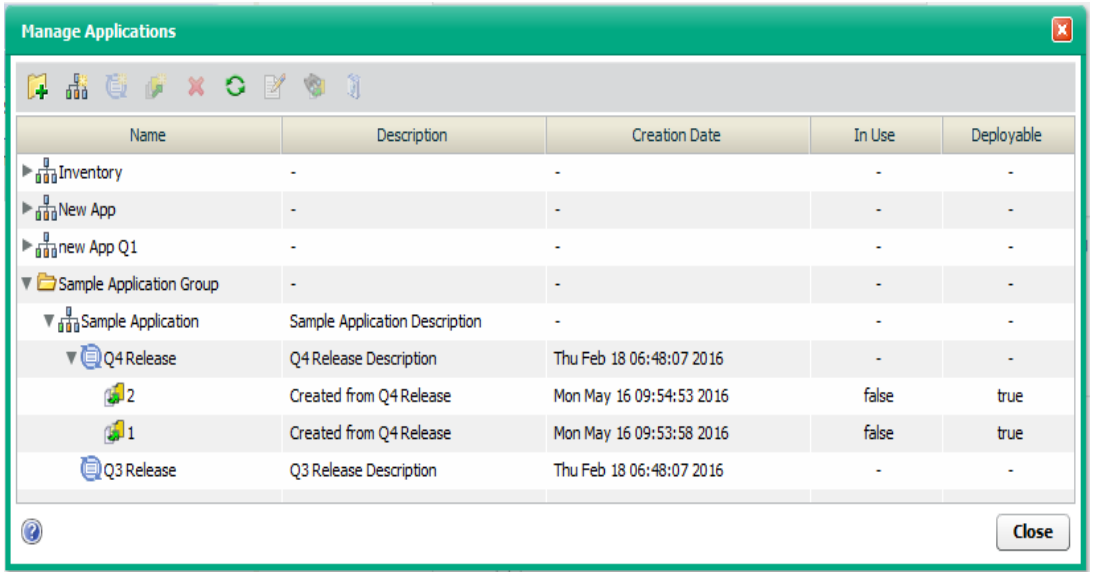
发行版

版本

只有您有权查看的应用程序才会可见。

Name 和 Description 列显示创建 (或最近编辑) 项目时指定的信息。Creation Date 仅针对发行版和版本显示。In Use 列指示版本是否已部署 (未取消部署或回滚)。Deployable 列指示是否已成功创建此版本。

Manage Applications 工具



要编辑特定项目，请选择包含该项目的行，然后单击工具栏上的  按钮，或者直接双击该项目。

由于版本是发行版在特定时间点的不变“快照”，因此版本无法编辑。

创建现有发行版的版本

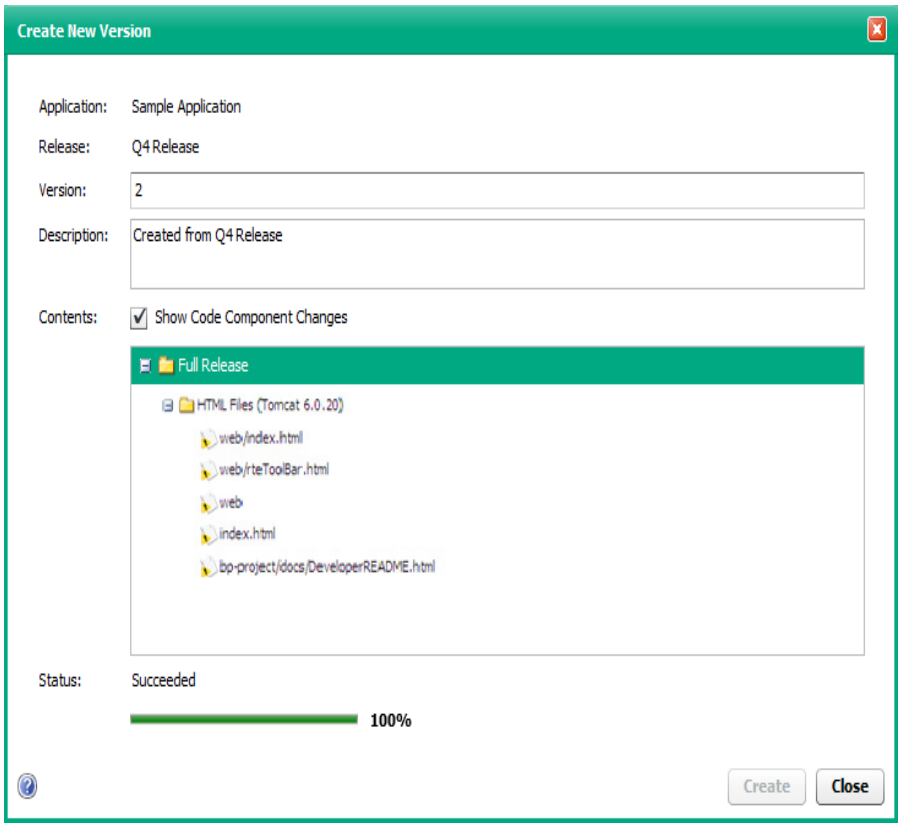
在可以部署发行版之前，必须创建该发行版的版本。版本是发行版以及在创建版本时与该发行版关联的生命周期的“快照”。发行版可能会继续变化，但是版本不会改变。这将确保可以重复部署发行版的特定版本。

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可创建发行版的版本。

请注意，还可以在 Deployment 屏幕上创建版本。有关详细信息，请参见部署应用程序 (第 503 页)。

要创建现有发行版的版本，请执行以下操作：

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择包含要使用的发行版的行。
3. 单击“Create Version”  按钮。将打开 Create New Version 对话框。
4. 指定以下信息：
   a. 显示的 Version 基于以前版本的名称 (通常是数字)。您可以根据需要对此进行更改。
   b. 可选：Description 最初指示版本所基于的发行版。您也可以根据需要对此进行更改。
   c. 可选：如果要查看代码组件文件的列表，请选中 Show Code Component Changes 框。对于完全发行版，将列出所有文件。对于增量发行版，仅列出在“基线“版本之后已更改的文件。例如：



5. 单击 **Create**。应用程序部署将创建版本。
6. 当状态显示为“Succeeded”时，单击 **Close**。
7. 如果作业失败，请单击 **Close**，然后转到 Jobs 屏幕确定原因。

创建新应用程序组

应用程序组是可用于在层次结构中组织应用程序的容器。如果您拥有适当的权限 (请参见 概述 (第 546 页))，则可创建应用程序组。

要创建应用程序组，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 要在顶级创建应用程序组，请勿选择任何内容。要创建嵌套应用程序组，请选择要在其下创建新组的组。
3. 单击 **Create Application Group**。

4. 指定新组的名称 (请参见命名规则 (第 412 页))。此名称在应用程序层次结构的此级别中必须唯一。
5. 单击 **OK**。现在，表中应当会显示新组。组和应用程序按字母顺序在层次结构的每个级别列出。

在 Manage Applications 工具中，您可以在组之间拖放应用程序。

创建新应用程序

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Applications 工具创建一个新应用程序。这是在 Applications 主屏幕上创建应用程序的另一种方法。如果要在添加层和组件之前设置应用程序权限，这将非常有用。

要在 Manage Applications 工具中创建新应用程序，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 要在顶级创建应用程序，请勿选择任何内容。要在现有的应用程序组中创建应用程序，请选择将包含新应用程序的组。
3. 单击 **Create Application**。
4. 指定以下信息:
   a. 提供新应用程序的唯一名称 (请参见命名规则 (第 412 页))。
   b. 可选：输入新应用程序的描述。
   c. 在 Release Name 字段中，输入对此应用程序使用的当前发行版的名称。默认情况下，这将是“Initial Release”。
   d. 选择此发行版的生命周期。该生命周期指定应用程序在从开发到生产的整个过程中经历的阶段。

      例如，常见的生命周期是开发 ® QA ® 预生产 ® 生产。这是默认的标准生命周期。
5. 单击 **OK**。现在，表中应当会显示新应用程序。应用程序和组按字母顺序在层次结构的每个级别列出。

后续步骤:

如果要指定此应用程序的权限，请将 Manage Applications 工具保持打开状态并查看设置权限 (第 546 页)。

如果要指定应用程序的层和组件，请关闭 Manage Applications 工具，然后选择刚才在 Applications 主屏幕上创建的发行版。

创建现有应用程序的新发行版

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Applications 工具创建现有应用程序的新发行版。新发行版是从现有发行版克隆的。它可以是完全发行版或增量发行版。完全发行版包含所有代码组件中所含的全部文件；增量发行版仅包含自指定的“基线”版本创建以来添加或修改的代码组件文件。

有关详细信息，请参见 管理应用程序 (第 434 页)。

要创建现有应用程序的新发行版，请执行以下操作：

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择包含要使用的应用程序的行。
3. 单击“Create Release”按钮。 . 将打开 Create New Release 对话框。
4. 指定以下信息：
   a. 从 Clone Release 下拉列表中，选择要克隆的发行版。新应用程序包含的层和组件将与所选的发行版相同。
   b. 输入此发行版的名称 (请参见命名规则 (第 412 页))。该名称在此应用程序中必须唯一。
   c. 可选：提供发行版的描述。此信息显示在报告和 Manage Applications 工具中。
   d. 可选：指定此发行版的变更请求编号。此信息还显示在报告中。
   e. 可选：在 ROI 框中，估计每台目标计算机预期实现的“投资回报”。这是您对通过部署此发行版将获得多少价值的评估。您可定义此字段的大小和含义。它用于 HPE Server Automation ROI 报告中。您可以指定任意正整数。
   f. 可选：指定 Planned Release Date。此信息还显示在报告中。
   g. 如果是增量发行版，请选中 Delta 复选框，然后从下拉列表中选择“基线”版本。仅部署为每个代码组件指定的源目录中自基线版本创建以来添加或修改的这些文件。

      如果是完全发行版，请确保未选中 Delta 复选框。
   h. 从下拉列表中选择相关的生命周期。
5. 单击 **OK**。

修改现有发行版的属性

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可修改任何发行版的属性。

要修改现有发行版的属性，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择包含要修改的发行版的行。
3. 单击“Edit Properties”按钮:  。
4. 修改发行版的属性。有关这些属性的描述，请参见应用程序概述 (第 428 页)。
5. 单击 **OK**。

删除应用程序组、应用程序、发行版或版本

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Applications 工具删除应用程序、组、发行版或版本。

要删除现有应用程序、组、发行版或版本，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择包含要删除的项目的行。请注意以下几点:
   - 可以使用 Ctrl 或 Shift 键选择多个项。但是，只能删除多个相同类型的项，例如，多个应用程序或多个发行版本。
   - 如果删除一个应用程序，则该应用程序的所有发行版和版本也会被删除，已计划但未运行的所有作业也一样。
   - 不能删除正在使用的的版本。

     必须先从 Jobs 屏幕“取消部署”此版本 (请参见管理应用程序部署作业 (第 525 页))，然后才能将其删除。

     单击 **Show Deployment Jobs** 可查看在删除此版本之前必须取消部署或回滚的作业列表。

请注意，除非您是应用程序部署管理员，否则只能取消部署或回滚您启动的部署。

- 某些应用程序组无法删除。

3. 单击“Delete”。

此操作不能撤消。系统会询问您是否确定要删除此项 (或这些项)。单击 **Yes** 可将其删除。

重命名应用程序或应用程序组

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Applications 工具更改应用程序或应用程序组的名称。

要重命名应用程序或组，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择包含要重命名的应用程序或组的行。
3. 单击 **Edit Properties**。
4. 输入应用程序或组的唯一名称 (请参见命名规则 (第 412 页))。
5. 单击 **OK**。

授予或撤销应用程序权限

创建应用程序时，您可以指定允许删除、编辑和部署应用程序的 SA 用户和用户组。如果不显式指定这些权限，只有您和您的应用程序部署管理员能够访问您的应用程序。有关详细信息，请参见概述 (第 546 页)。

要设置应用程序权限，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择包含要使用的应用程序的行。
3. 单击 **Edit Properties**。
4. 从 Choose User 菜单中，选择一个用户或用户组。
5. 单击 **Add User**。

6. 选择要授予的权限:
   a. View 权限允许用户在 Applications 屏幕上的 Select Release 下拉列表中查看应用程序。
   b. Edit 权限允许用户更改应用程序的内容 (组件)、结构 (层) 或属性 (名称、描述、发行版名称等)。
   c. Deploy 权限允许用户将应用程序部署到一个或多个目标。

   您可以选择或拒绝这些权限的任意组合。选择 Edit 或 Deploy 时，会自动选择 View。
7. 要从权限列表中删除一个用户 (或组)，请选择包含该用户 (或组) 的行，然后单击 **Delete** 。
8. 单击 **OK**，关闭 Edit Applications 对话框并保存所做的更改。**Cancel** 按钮将放弃所做的更改。

查看特定版本的属性

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可从 Manage Applications 工具打开版本查看器 (请参见查看版本的属性 (第 517 页))。

要从 Manage Applications 工具打开版本查看器，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择要查看的版本。
3. 单击工具栏上的 **View version**。将打开版本查看器。

查看使用特定版本的部署作业

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可查看使用发行版的最新版本的部署作业的列表。

要查看使用特定版本的部署作业的列表，请执行以下操作:

1. 打开 Manage Applications 工具 (请参见管理应用程序 (第 434 页))。
2. 在表中选择要查看的版本。
3. 单击工具栏上的 **Show Deployment Jobs**。将打开 Jobs 屏幕，并显示使用此版本的所有部署作业的列表。

组件

概述

您可以在有权编辑的应用程序中创建和修改组件 (请参见 设置权限 (第 546 页))。

要查看包含简单组件的应用程序示例，请参见 快速入门 (第 414 页)。

关于组件

组件表示应用程序的内容，它将实现应用程序的各种功能，并与特定层关联。有八种类型的组件可供您用来填充应用程序层。

组件类型	图标	用途
代码		将文件复制到目标服务器。例如，这可能是一组实现基于 Web 的存储的 HTML 文件。请参见 代码组件 (第 443 页)。
脚本		在应用程序部署期间在目标服务器上运行脚本。例如，您可能有两个脚本: 一个用于在部署之前关闭所有应用程序，另一个用于在部署之后重新启动应用程序。请参见 脚本组件 (第 447 页)。
配置文件		在目标服务器上创建新配置文件。这些文件通常包含特定应用程序的非常具体的信息。例如，您可能需要创建一个配置文件来指定数据库的信息。请参见 配置文件组件 (第 448 页)。
应用程序配置		将现有的应用程序配置从 SA 库部署到目标服务器。请参见 应用程序配置组件 (第 451 页)。
软件策略		将软件策略从 SA 库附加到目标服务器。或者，您可以指定要安装 (而不是附加) 的策略。请参见 软件策略组件 (第 455 页)。
程序包		在目标服务器上安装 SA 库中的软件包。或者，您可以指定要作为策略附加的包。请参见 程序包组件 (第 457 页)。

组件类型	图标	用途
OO 流		启动 **HPE Operations Orchestration (OO)** 工作流。例如，该工作流可能如下：每次部署特定的应用程序时，均向产品所有者发送一封电子邮件。请参见 OO 流组件 (第 460 页)。
DMA 流		在目标服务器、实例或数据库上启动 **HPE Database and Middleware Automation (DMA)** 工作流。例如，您可以使用 DMA 工作流安装关键的数据库修补程序。
Windows 注册表		添加或删除 Windows 注册表项或值。此类组件仅对 Windows 层可用。请参见 Windows 注册表组件 (第 462 页)。

组件针对服务器执行。默认情况下，组件按创建顺序部署。如果您有权编辑应用程序，则可修改此顺序 (请参见更改组件的部署顺序 (第 467 页))。

组件类型

以下是不同类型的应用程序部署组件：

- 代码组件 (第 443 页)
- 脚本组件 (第 447 页)
- 配置文件组件 (第 448 页)
- 应用程序配置组件 (第 451 页)
- 软件策略组件 (第 455 页)
- 程序包组件 (第 457 页)
- OO 流组件 (第 460 页)
- Windows 注册表组件 (第 462 页)

仅当您已将 SA 核心服务器配置为与 HPE OO 服务器集成时，OO 流组件才可用。

Windows 注册表组件只能用于 Windows 层。

代码组件

代码组件用于将文件部署到托管服务器。代码组件就是应用程序部署从指定存储库创建的文件包。此存储库可以是下列任一选项：

- SA 核心服务器的文件系统。
- 通过 URL 访问的单个文件
- 源代码控制系统

代码组件是动态的。每次创建版本时，它们将捕获目录 (或 URL) 的内容。相反，软件策略或程序包组件每次都使用相对静态的相同 SA 库项目。

代码组件可以包含的最大文件数是 100,000。

创建代码组件时，请指定以下信息:

代码组件属性

属性	必需?	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。
描述	否	显示在报告中的描述。
基路径	否	基路径与默认安装路径相结合形成完整安装路径，即:将在托管服务器上放置文件的位置。仅当启用沙盒时，基路径和完整安装路径才可见 (请参见“管理应用程序设置”)。 请参见此表下面的备注。
默认安装路径	是	将在目标上放置文件的目录 (如果启用沙盒，则位于基路径中)。
完整安装路径	只读	基路径 + 安装路径 (仅当启用沙盒时可见)。
源类型	是	源文件所在的位置: 文件系统 - 在 SA 核心服务器的文件系统中查找文件。这包括任何 Samba 和 NFS 安装点。 CVS 或 Subversion - 在指定的源代码控制系统中查找文件 (请参见使用源代码管理系统 (第 446 页))。 URL - 根据在 URL 中指定的协议查找单个文件 (请参见使用 URL (第 447 页))。 这些位置由应用程序部署管理员维护。有关详细信息，请参见管理代码组件源类型 (第 570 页)。
源路径	是	源文件在存储库中所处的目录。此目录必须安装在 SA 核心服务器上。 如果指定 URL 为源类型，则没有源目录。取而代之的是 URL 属性。

代码组件属性(续)

属性	必需?	用途
源文件	是	Everything - 获取源目录中的所有文件。 Include - 获取源目录中名称与指定模式匹配的所有文件。 Exclude - 获取源目录中除与指定模式匹配的文件外的所有文件。 请参见 使用筛选器 (第 446 页)。
部署方法	是	创建新版本时，应用程序部署会创建一个程序包，其中包含指定的文件和所需的任何脚本。 Policy - 应用程序部署在创建新版本时也创建策略。部署版本时，它将在目标中包含的每个托管服务器上附加和修正此策略。 Package - 部署版本时，应用程序部署将在目标中包含的每个托管服务器上“临时”安装程序包。
备份	否	可用的备份和回滚脚本取决于层的平台。这些脚本由应用程序部署管理员维护。有关详细信息，请参见管理脚本 (第 568 页)。
回滚	否	指定 SA 在备份和回滚脚本完成之前应等待的最长时间 (以秒为单位)。如果这些脚本在此期间没有完成，则会被中止。默认值为 3600 秒 (1 小时)。
超时	是	
Undeploy	否	启用或禁用。

> **备注:** 启用沙盒后，您必须在创建代码组件时指定基路径。不能为包含在启用沙盒后未指定基路径的代码组件的发行版创建版本。

如果应用程序部署管理员删除了现有发行版中的代码组件所使用的基目录，您将无法创建该发行版的新版本。但是，您仍然能够部署该发行版的现有版本。

使用符号链接

应用程序部署支持代码组件中的符号链接。如果要使用符号链接，请注意以下几点:

- 符号链接仅适用于 UNIX 目标；它们在 Windows 操作系统上不起作用。
- 如果源目录是符号链接，SA 将访问该链接以查找代码组件的文件。

- 如果在目录结构中找到源目录指向的符号链接，则将存档而不访问该符号链接。在目标服务器上安装这些文件时，将使用原始值创建该符号链接。
- 就包含项和排除项而言，符号链接的处理方式与文件相同。

使用筛选器

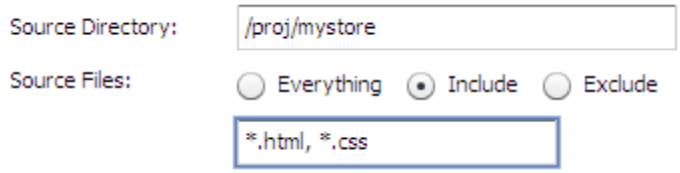
您可以使用含 Include/Exclude 选项的筛选器指定代码组件中包含的文件。

- 对于完全发行版，满足筛选条件的所有文件和目录包括在内。
- 对于增量发行版，仅包括满足筛选条件且时间戳晚于相关版本的文件。如果一个文件已从文件系统删除，则不会出现在组件程序包中。

筛选器可识别 * 通配符。应用程序部署以递归方式搜索源目录的子目录中是否存在与指定模式匹配的文件。例如：

- */src/* 将所有文件与路径中名为 src 的目录相匹配
- Fred* 将名称以 Fred 开头的所有文件与路径中名为 Fred 的目录相匹配

请确保用逗号分隔筛选器模式，如下所示：



例如，此模式将所有文件与 *.html 或 *.css 文件扩展名相匹配，其中包括：

/proj/mystore/index.html

/proj/mystore/products/dogs/dachshunds.html

/proj/mystore/main.css

使用源代码管理系统

例如，CVS 和 Subversion 便属于源代码控制系统。应用程序部署与最常用的源代码控制系统相集成。

在本例中，源目录是指 SA 核心服务器上存在的源存储库的客户端副本所在的位置。您可以使用 Include/Exclude 筛选器指定要包含在组件中的文件。

使用源代码控制系统的源类型具有以下优点：

- 每次创建发行版的版本时，都使用源代码控制系统刷新客户端副本中的文件。这样，您可以确保代码组件中包含最新的更新。
- 对于增量发行版，如果一个文件从源代码控制系统删除，在部署新版本时，它也会从目标服务器删除。

如果您有多个代码组件访问同一源代码控制存储库，请务必为每个组件指定不同的源目录。这可防止在部署应用程序时出现并发错误。

应用程序部署管理员必须先集成源代码控制系统，然后您才能在应用程序中使用此源类型。有关详细信息，请参见管理代码组件源类型 (第 570 页)。

使用 URL

使用 URL 源类型创建代码组件时，需指定一个 URL，指向可由该 URL 中指定的任何协议 (通常是 HTTP) 访问的单个文件。例如，您可以从自动化构建系统放置应用程序的位置中检索其每晚的最新内部版本。

如果使用 HTTPS 协议，您必须确保所需的证书安装在正确的位置中。

脚本组件

部署期间，脚本组件在应用程序的目标上运行。例如，您可能有两个脚本:一个用于在部署之前关闭所有应用程序，另一个用于在部署之后重新启动应用程序。

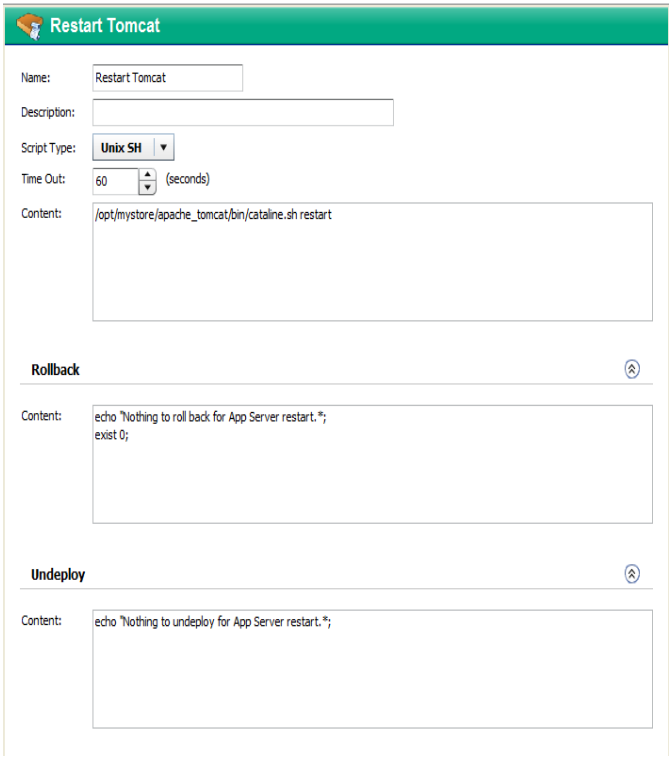
创建脚本组件时，请指定以下信息:

脚本组件属性

属性	必需?	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。
描述	否	可用于为其他应用程序部署用户记录组件的字段。
脚本类型	是	脚本的类型。可用的选项取决于层所需的平台。
超时	是	SA 在脚本完成运行之前应等待的时间 (以秒为单位)。
内容	是	部署期间将在目标上运行的实际指令。
回滚内容	否	如果要启用回滚或取消部署，您必须为每个场景提供显示指令。使用您为脚本本身内容采用的相同脚本语言。
取消部署内容	否	

以下这一简单示例将在目标上重新启动 Tomcat 应用程序服务器。

UNIX 目标的脚本组件示例



您可以在脚本组件中使用参数。有关详细信息，请参见参数和特殊变量 (第 474 页)。

配置文件组件

配置文件组件将在目标上创建一个新的配置文件。此文件包含应用程序所需的特定信息。例如，您可能需要创建一个配置文件来指定要用于特定应用程序的数据库。

配置文件组件最适合简单的键-值型文件。

对于复杂的参数化或特定于应用程序的配置，请改用应用程序配置组件 (请参见应用程序配置组件 (第 451 页))。

创建配置文件组件时，必须指定以下信息:

配置文件组件属性

属性	必需？	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。

配置文件组件属性(续)

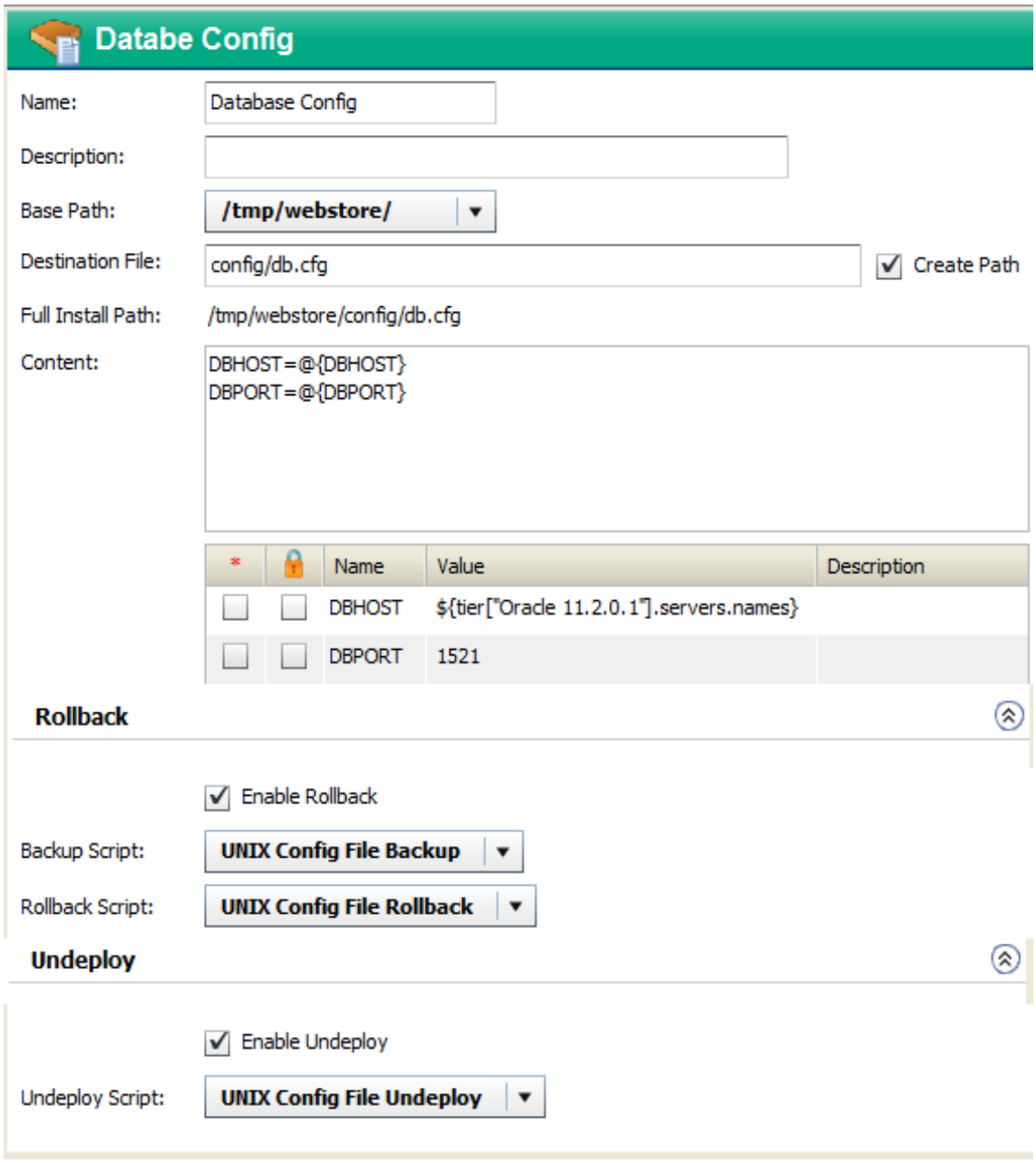
属性	必需?	用途
描述	否	报告中的描述。
基路径	否	基路径与目标文件相结合形成完整安装路径，即:将在托管服务器上放置配置文件的位置。 仅当启用沙盒时，基路径和完整安装路径才可见(请参见"管理应用程序设置")。 请参见此表下面的备注。
目标文件	是	将在目标服务器上创建的配置文件的名称。 如果启用沙盒，基路径将与目标文件相结合形成完整安装路径，即:将创建配置文件的位置。 如果未启用沙盒，您必须指定将在目标服务器上创建配置文件的完整绝对路径。请勿指定相对路径。例如: `/tmp/webstore/config/db.cfg`
完整安装路径	只读	基路径 + 目标文件 (仅当启用沙盒时可见)。
创建路径	否	选择 Create Path 时，将在目标服务器上创建将目标文件放在正确位置所需的目录层次结构 (如果该层次结构尚不存在)。
内容	是	配置文件的内容。
回滚	不适用	可用的备份、回滚和取消部署脚本取决于层的平台。这些脚本由应用程序部署管理员维护。有关详细信息，请参见管理脚本 (第 568 页)。
Undeploy	不适用	

> **备注:** 启用沙盒后，您必须在创建配置文件组件时指定基路径。不能为包含在启用沙盒后未指定基路径的配置文件组件的发行版创建版本。

如果应用程序部署管理员删除了现有发行版中的配置文件组件所使用的基目录，您将无法创建该发行版的新版本。但是，您仍然能够部署该发行版的现有版本。

配置文件组件可以使用在部署时分配了值的参数。有关详细信息，请参见组件类型 (第 443 页)。

UNIX 目标的配置文件组件示例



这一简单的示例将创建一个名为 db.cfg 的文件，其中包含数据库服务器主机名和端口。将组件部署到目标时，会分配 DBHOST 和 DBPORT 参数的值。DBPORT 的值为 1521。DBHOST 的值是属于目标中 Oracle 11.2.0.1 层一部分的服务器名称的逗号分隔列表。

应用程序配置组件

SA 允许您从一个中心位置管理配置文件，并在数据中心的多台服务器中轻松传播更改和更新。您可以管理单个配置文件 (如 UNIX 系统中的 /etc/hosts 文件)，也可以管理与某个应用程序关联的多个复杂的配置文件，例如与某个大型业务应用程序 (如 Web Logic 或 WebSphere) 关联的配置文件。

SA 用来管理配置文件的机制称为“应用程序配置”。它由两种类型的元素组成: 一个或多个模板和值集。

应用程序部署允许您使用 SA 库中的现有应用程序配置创建组件。您可以在部署时修改值集中的项。例如，可能有一个 XML 文件必须包含特定于应用程序部署目标的信息，例如，哪台服务器托管数据库。

创建应用程序配置组件时，请指定以下信息:

应用程序配置组件属性

属性	必需?	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。
描述	否	报告中的描述。
应用程序配置名称	是	这三个属性在 SA 库的应用程序配置中构建。 您不能在应用程序部署用户界面中修改这些属性。但是，如果您对包含应用程序配置的文件夹具有适当权限，则可打开 SA Application Configuration 窗口并在其中修改这些属性。要打开此窗口，请单击此应用程序配置的 Name 链接。
应用程序配置描述	否	表中列出了允许您在部署时修改应用程序配置的值集中的任何项。 如果特定应用程序配置存在多个模板，则可从所有模板进行编辑的所有参数将显示在应用程序部署用户界面中。
实例名称	是	
Parameters	否	
启用回滚	不适用	您可以对此应用程序配置启用或禁用回滚和取消部署。实际的回滚和取消部署行为取决于应用程序配置本身。

应用程序配置组件属性(续)

属性	必需？	用途
启用取消部署	不适用	

以下示例显示了可在多个不同网络上部署，并且每个网络具有不同的数据库服务器的应用程序的应用程序配置。此应用程序用于从 XML 配置文件 dbinfo.xml 检索连接信息。要使应用程序能够连接到数据库，需要 dbinfo.xml 中的四个值:

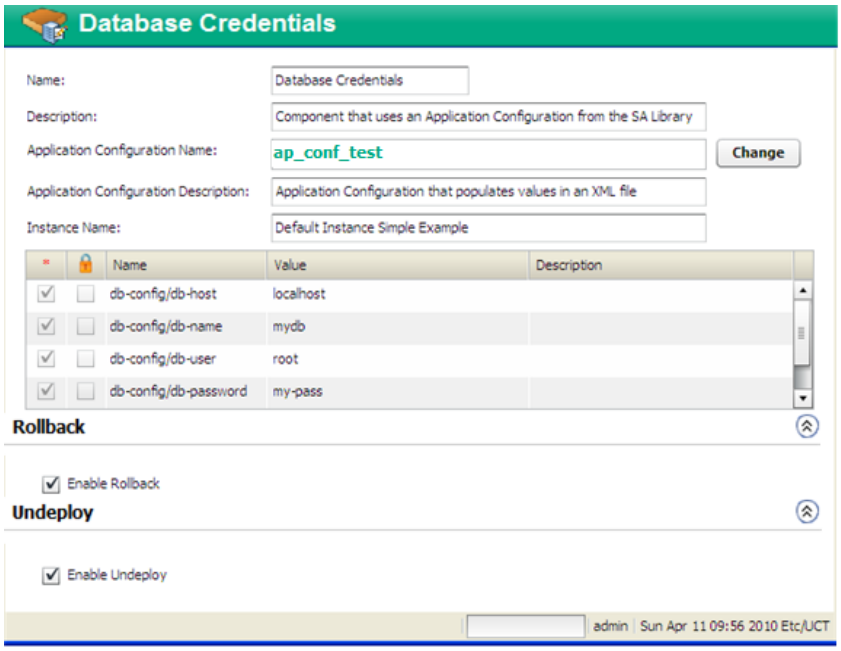
- **主机**：安装数据库的服务器的主机名
- **名称**：主机服务器上的数据库名称
- **用户**：用于打开数据库连接的用户名凭据
- **密码**：打开数据库连接所需的密码

以下是 dbinfo.xml 文件的示例:

```
<?xml version="1.0" ?>
<db-config>
<db-host>localhost</db-host>
<db-name>mydb</db-name>
<db-user>root</db-user>
<db-password>my-pass</db-password>
</db-config>
```

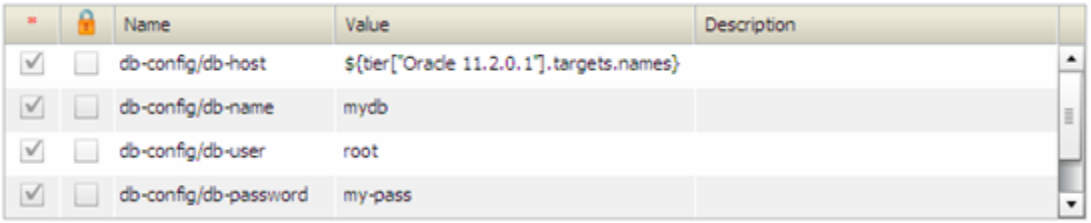
以下是基于此应用程序配置的组件。此处，db-host 参数将在部署时修改为使用目标的名称。

应用程序配置组件示例



要配置应用程序配置组件，请执行以下操作：

1. 创建类型为应用程序配置的新组件。
2. 从 SA 库中查找并选择要使用的应用程序配置。
3. 要修改参数的值，请在表中单击该参数的 Value 单元格。您可以指定常数值或选择要在部署时分配值的特殊变量，如下所示。

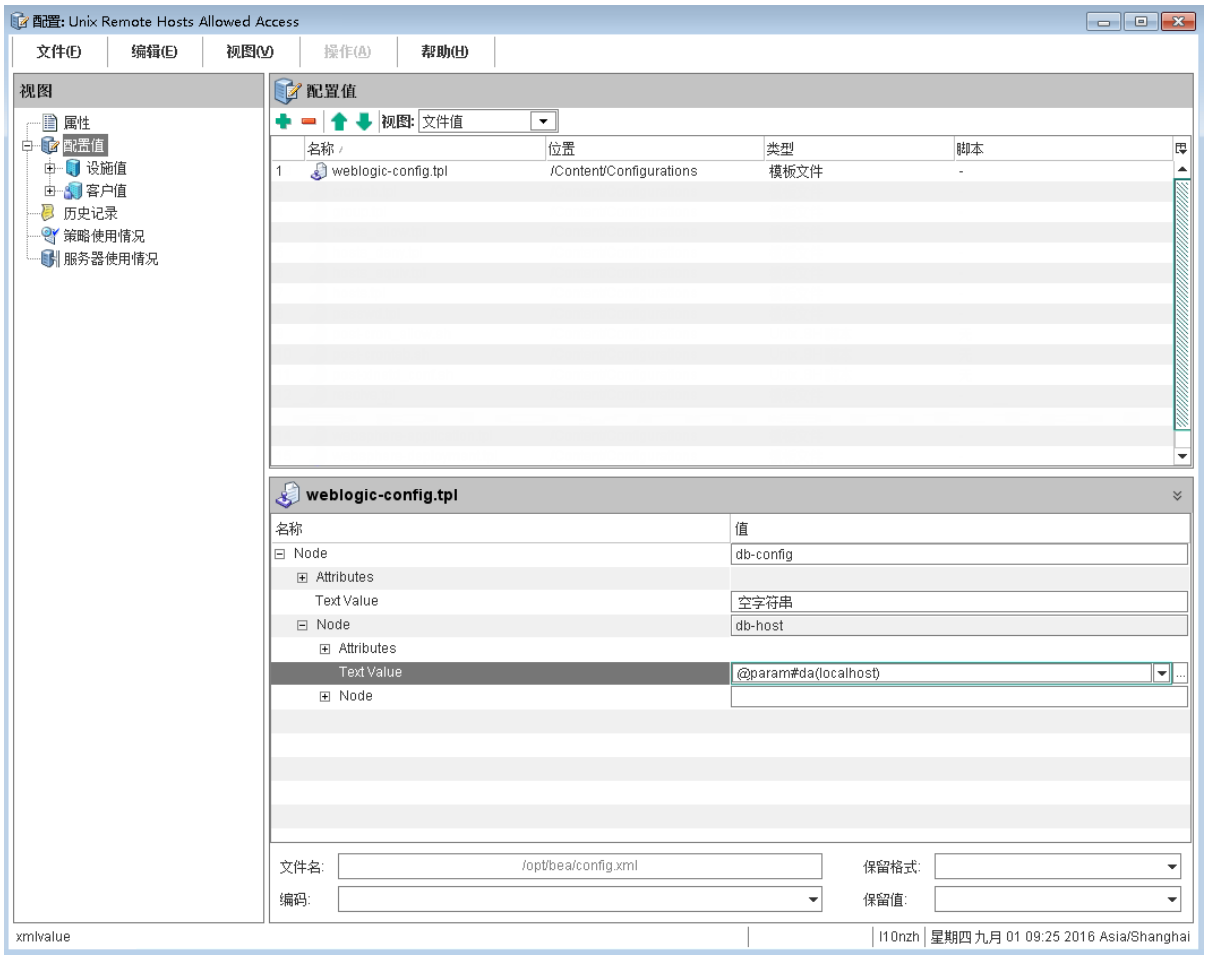


有关详细信息，请参见 参数和特殊变量 (第 474 页)。

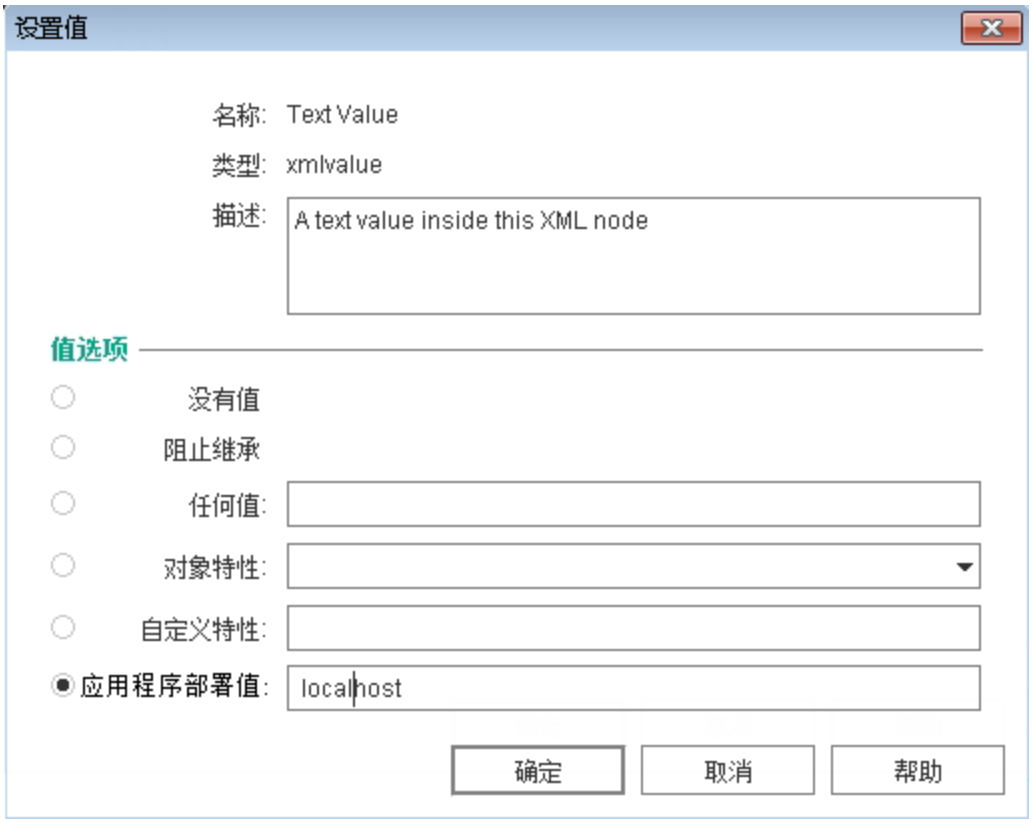
4. 启用或禁用 Rollback 和 Undeploy。

请务必先保存组件，然后再导航到另一个屏幕。

要使应用程序配置参数在应用程序部署用户界面中可用，您必须在 SA 服务器上为该参数指定 Deployment Automation Value 选项，如下所示：



为此，请选中该参数的 Value 框，然后单击 按钮。将打开 Set Value 对话框:



选择 Deployment Automation Value 选项。如果要提供默认值，请在文本框中键入该值。

有关应用程序配置的详细信息，请参见《SA 10.50 用户指南》。

软件策略组件

Server Automation (SA) 使用软件策略自动在托管服务器上安装软件和配置应用程序。软件策略中可以包含程序包、RPM 程序包、修补程序、应用程序配置、脚本和服务器对象。它也可以与 OS 序列关联。创建软件策略后，可以将它附加到服务器或服务器组。然后，您可以评估哪些托管服务器符合或不符合特定策略。

您可以根据 SA库中的软件策略创建应用程序组件。

创建软件策略组件时，请指定以下信息:

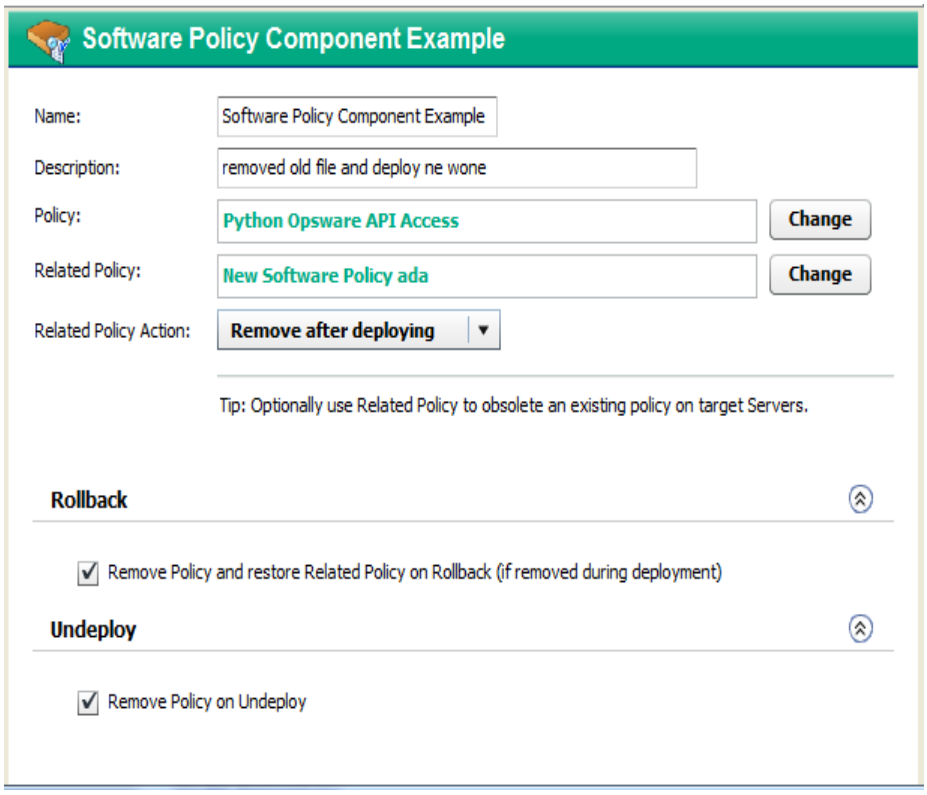
软件策略组件属性

属性	必需?	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。

软件策略组件属性(续)

属性	必需？	用途
描述	否	报告中的描述。
Policy	是	部署应用程序时将在目标服务器上附加和修正的策略。
相关策略	否	要被组件中指定的策略取代的现有策略 (已与服务器关联)。
相关策略操作	如果指定了 Related Policy	在部署后删除 (默认值)、在部署前删除或不删除。如果指定 Do Not Remove，仅在回滚期间使用 Related Policy。
回滚	不适用	如果选中 Rollback 框，在回滚应用程序部署时将出现以下情况： 删除在 Policy 框中指定的软件策略。 如果在部署期间删除 Related Policy 框中指定的策略，将还原此策略。
Undeploy	不适用	如果选中 Undeploy 框，将在取消部署此应用程序时删除在 Policy 框中指定的软件策略。

软件策略组件示例



要配置软件策略组件，请执行以下操作:

1. 创建类型为 Software Policy 的新组件。
2. 从 SA 库中查找并选择要使用的策略和相关策略。
3. 选择所需的相关策略操作。
4. 指定回滚和取消部署行为。

请务必先保存组件，然后再导航到另一个屏幕。

有关软件策略的详细信息，请参见 SA 联机帮助中的“创建和管理软件策略”。

程序包组件

SA 软件库中的应用程序组织在程序包中。要在 SA 中安装程序包，请将该程序包添加到软件策略，将该策略附加到一个或多个托管服务器，然后执行服务器修正。

您可以创建包含 SA 库中一个或多个程序包以及可选相关策略的应用程序部署组件。您还可以“即时”创建新程序包并将该程序包导入到 SA 库中。您可以选择附加策略并使用常规的 SA 服务器修正工作流，或者仅安装程序包而不附加策略。

创建程序包组件时，请指定以下信息:

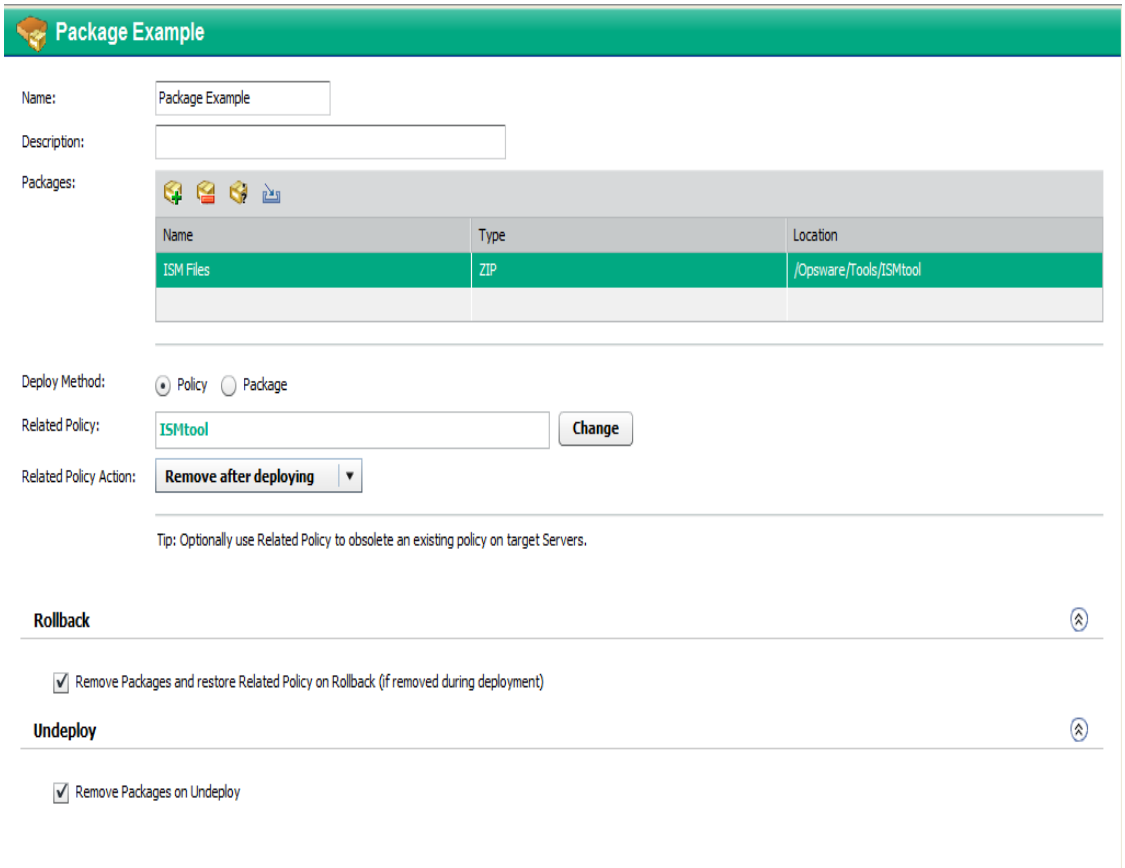
程序包组件属性

属性	必需?	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。
描述	否	报告中的描述。
程序包	是	SA 库中的程序包列表。
部署方法	是	创建新版本时，应用程序部署会创建一个程序包，其中包含指定的程序包和所需的任何脚本。 Policy - 应用程序部署在创建新版本时也创建策略。部署版本时，它将在每个目标服务器上附加和修正此策略。 Package - 部署版本时，应用程序部署将在每个目标服务器上“临时”安装程序包。
相关策略的相关软件包	否	现有策略或要被组件中指定的程序包取代的现有程序包。
相关软件包操作	如果指定了相	在部署后删除 (默认值)、在部署前删除或不删除。

程序包组件属性(续)

属性	必需？	用途
	关程序包	如果指定 Do Not Remove，仅在回滚期间使用相关程序包。
回滚	不适用	如果选中 Rollback 框，在回滚应用程序部署时将出现以下情况: 删除程序包。 如果删除了指定的相关策略，将会还原。始终会还原指定的相关程序包。
Undeploy	不适用	如果选中 Undeploy 框，将在取消部署此应用程序时删除在 Packages 表中列出的程序包。

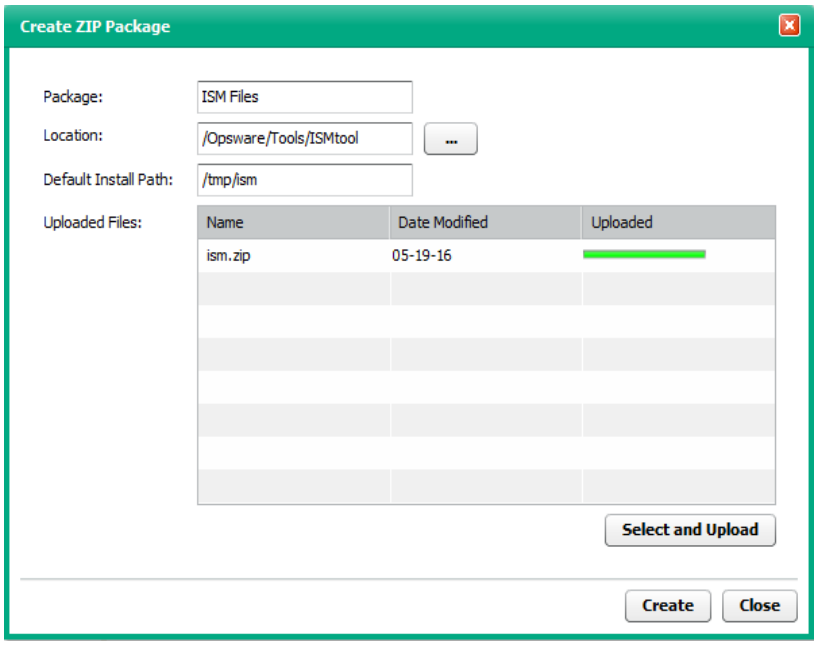
程序包组件示例



可以使用下列工具指定组件中的程序包:

- 单击 “Add Package” 按钮，从 SA 库中选择一个程序包。此时将打开 Select Package 对话框。请参见 使用组件 (第 464 页)。
- 使用通配符选择与搜索字符串匹配的所有程序包。为此，请执行以下步骤：
  a. 在 Packages 表中选择一个程序包。
  b. 单击 “Wildcard Package Name” 按钮。将打开 Set Wildcard 对话框。
  c. 修改 Name 框中的文本，以使用 * 和 ? 通配符创建搜索字符串。
  d. 单击 **OK**。

  创建新版本时，应用程序部署将包括 SA 库的同一位置 (文件夹) 中与搜索字符串和程序包类型匹配的最新程序包。
- 单击 **Create ZIP Package from Files** 以创建新的 ZIP 包。此时将打开 Create ZIP Package 对话框：



Location 是新程序包将在 SA 库中驻留的文件夹。

Default Install Path 是部署应用程序时将在目标上安装应用程序的位置。

要创建新程序包，请执行以下步骤：

1. 单击 **Select and Upload**。
2. 在本地系统上浏览到要包含在程序包中的文件。
3. 单击 **Create** 以创建 ZIP 包并将其导入到 SA 库中。

现在，您可像部署 SA 库中的任何其他程序包一样部署您的程序包。

要配置程序包组件，请执行以下操作:

1. 创建类型为程序包的新组件。
2. 使用 **Add Package** 工具或 **Import Package** 工具指定要包含的程序包。
3. 指定部署方法 (请参见软件策略组件属性 (第 455 页))。
4. 使用 **Add Package** 工具或 **Import Package** 工具指定要使用的相关程序包，然后选择 **Related Policy Action**。
5. 指定回滚和取消部署行为。

请务必先保存组件，然后再导航到另一个屏幕。

有关软件策略的详细信息，请擦肩 SA 联机帮助中的“管理软件包”。

OO 流组件

这种类型的组件将启动 **HPE Operations Orchestration (OO)** 工作流。这在您想要与其他应用程序集成时非常有用。例如：

- 发出 HPE Service Manager 问题工单，指示某组服务器已更新
- 配置负载平衡器以了解部署
- 与应用程序监控系统交互

如果 SA 核心服务器配置为与 HPE OO 服务器集成，您可以创建包含 HPE OO 工作流的应用程序部署组件。

也可以将 HPE OO 工作流与特定环境关联。通过使用此策略，您可以创建以下类型的场景:

1. 在环境级别，启动“部署前”OO 工作流。
2. 部署应用程序。
3. 在环境级别，启动“部署后”OO 工作流。

例如: 对 10 minutes®deploy the application®re-enable monitoring 禁用监控。

在 QA 环境中，您可以在每晚构建后运行冒烟测试 (即验证)。然后，您可能希望向产品所有者发送一封电子邮件，看看测试结果是否可接受。如果收到子邮件回复，您可以部署新构建。

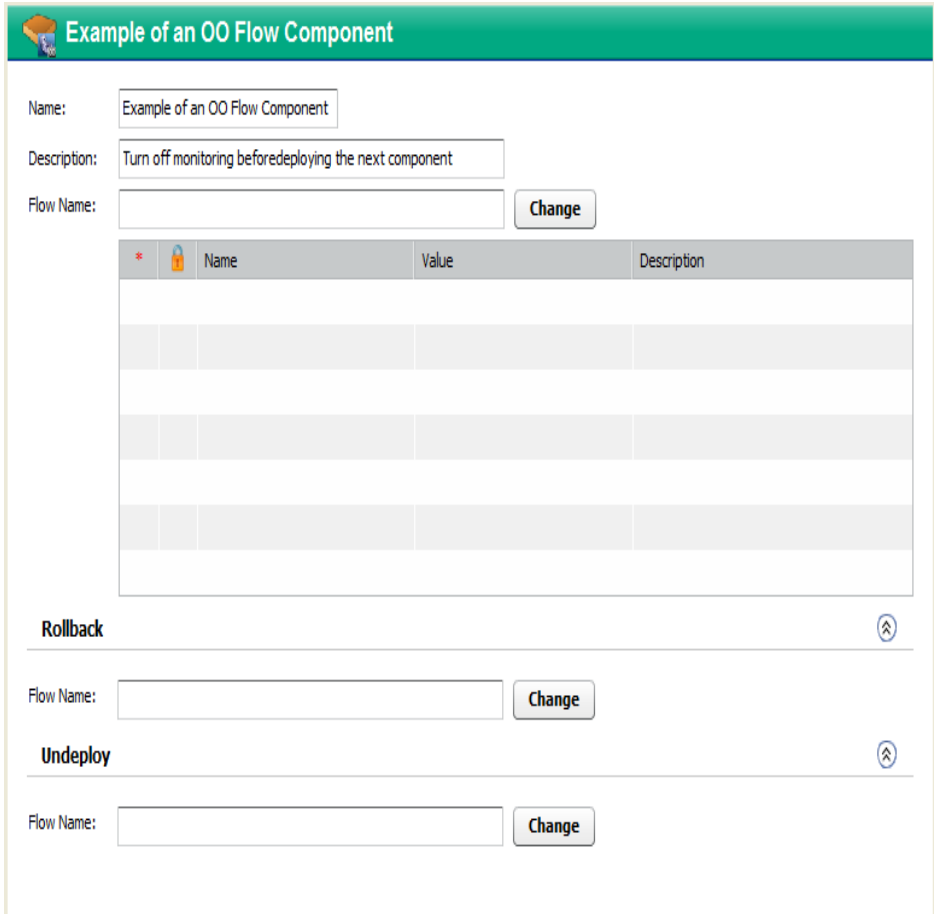
有关详细信息，请参见管理环境 (第 563 页)。

创建 OO 流组件时，请指定以下信息：

OO 流组件属性

属性	必需？	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。
描述	否	报告中的描述。
流名称	是	对 HPE OO 工作流的引用。
流参数	取决于选定的流	选定的 HPE OO 工作流所需的参数。 请注意，同名参数跨函数、回滚和取消部署区段执行。
回滚流	否	在部署失败时用于回滚的 HPE OO 工作流。
取消部署流	否	在取消部署应用程序时使用的 HPE OO 工作流。

OO 流组件示例



要配置 OO 流组件，请执行以下操作：

1. 创建类型为 OO 流的新组件。
2. 从 HPE OO 库中选择 HPE OO 工作流。

   如果您正在搜索 OO 流，则可通过在文本框中指定完整字词来提高搜索速度。OO 流搜索仅匹配完整字词。例如，在此上下文中，"start" 与 "restart" 不匹配。
3. 要修改参数的值，请在表中单击该参数的 Value 单元格。您可以指定常数值或选择要在部署时分配值的特殊变量 (请参见参数和特殊变量 (第 474 页))。
4. 指定回滚和取消部署行为。

请务必先保存组件，然后再导航到另一个屏幕。

Windows 注册表组件

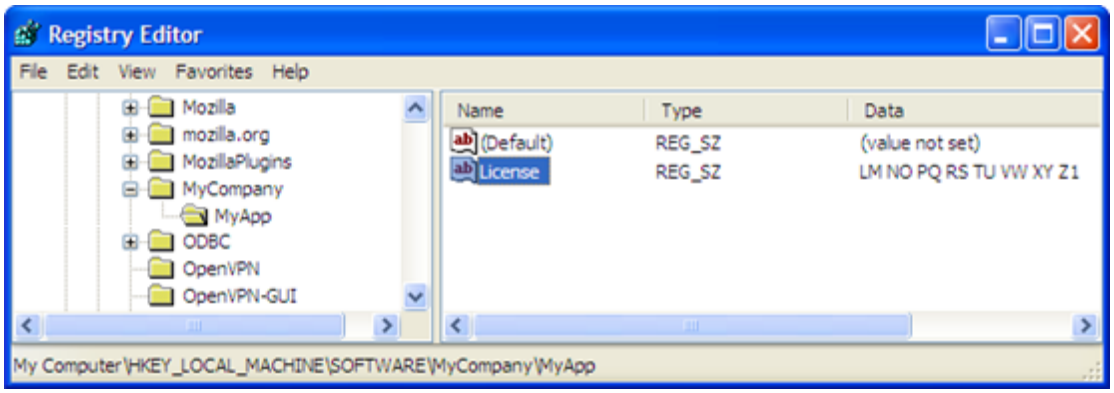
使用这种类型的组件可添加或删除 Windows 注册表项或值。此类组件仅对 Windows 应用程序可用。

创建 Windows 注册表组件时，请指定以下信息：

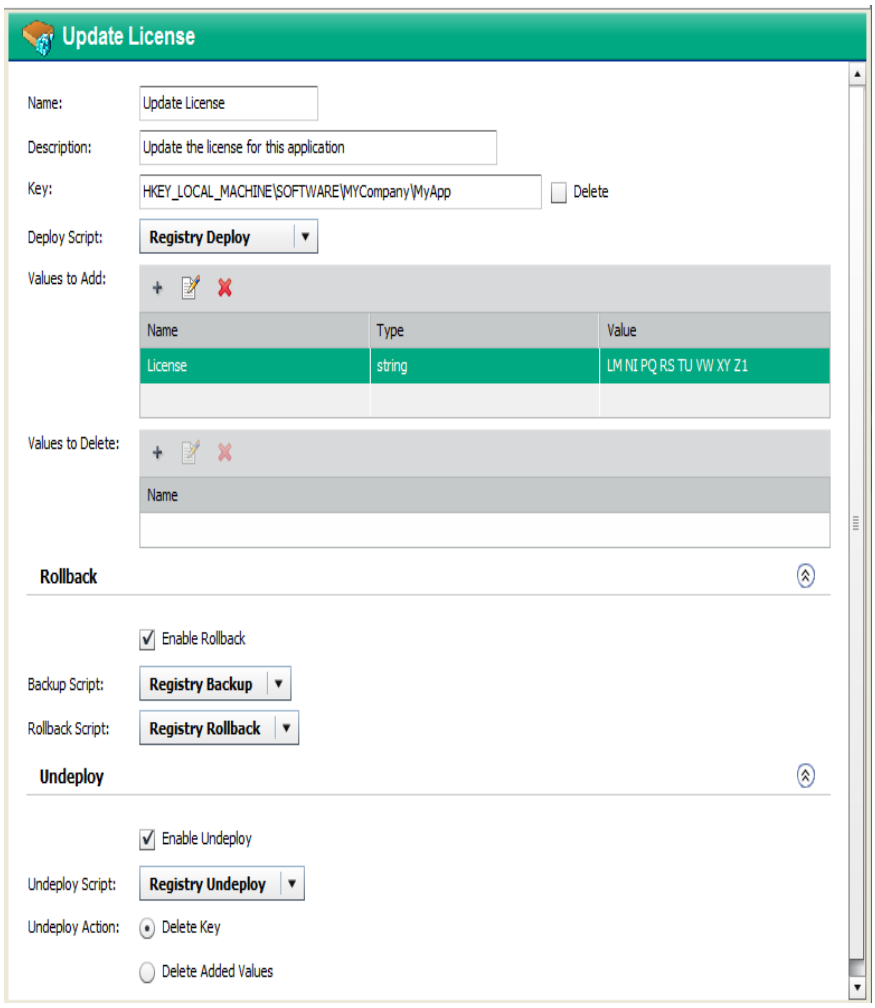
Windows 注册表组件属性

属性	必需？	用途
名称	是	组件的名称。此名称将显示在层中。它在应用程序中必须唯一。
描述	否	报告中的描述。
密钥	是	要修改的注册表项。
值	是	要在指定项中添加或删除的值。如果要删除该项，则不需要值。
备份脚本	如果启用回滚	备份、回滚和取消部署脚本由应用程序部署管理员维护。有关详细信息，请参见管理脚本 (第 568 页)。
回滚脚本		
取消部署脚本	如果启用取消部署	
取消部署操作	如果启用取消部署	删除整个密钥和任何值或仅删除添加的值。

以下注册表组件会将此许可证密钥值添加到注册表：



Windows 注册表组件示例



必须以正确的格式指定密钥名称和密钥值。请注意以下几点:

- Key 字段必须以有效的配置单元名称开头。以下配置单元名称有效:HKEY_CLASSES_ROOT、HKCR、HKEY_CURRENT_USER、HKCU、HKEY_LOCAL_MACHINE、HKLM、HKEY_USERS、HKU、HKEY_CURRENT_CONFIG、HKEY_PERFORMANCE_DATA 和 HKEY_DYN_DATA。
- 只能将反斜杠用作密钥组件分隔符。例如：
- HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\HP Virtual Rooms 8.0
- 二进制类型密钥值的 Value 字段必须指定为一系列空格分隔的两位十六进制数。例如：01 A2 B3 C4
- 双字类型密钥值的 Value 字段必须指定为以 0x 开头的十六进制数或十进制正数。

要配置 Windows 注册表组件，请执行以下操作:

1. 创建类型为 Windows 注册表的新组件。
2. 要添加新密钥值，请执行以下步骤。
   a. 在 Values to Add 表中，单击 **Add**。
   b. 在 Add Value 对话框中，指定 **Name**、**Type** 和 **Value**。

      您可以键入 **Value**，或选择在部署时分配了值的特殊变量 (请参见组件类型 (第 443 页))。
   c. 单击 **OK**。
3. 要删除现有密钥值，请执行以下步骤。
   a. 在 Values to Delete 表中，单击 **Add**。
   b. 在 Add Value to Delete 对话框中，指定其 **Name**。
   c. 单击 **OK**。
4. 要修改任一表中的值，请选择该值，然后单击 **Edit Properties**。
5. 要从任一表中删除值，请选择该值，然后单击 **Delete**。
6. 指定回滚和取消部署行为。

请务必先保存组件，然后再导航到另一个屏幕。

使用组件

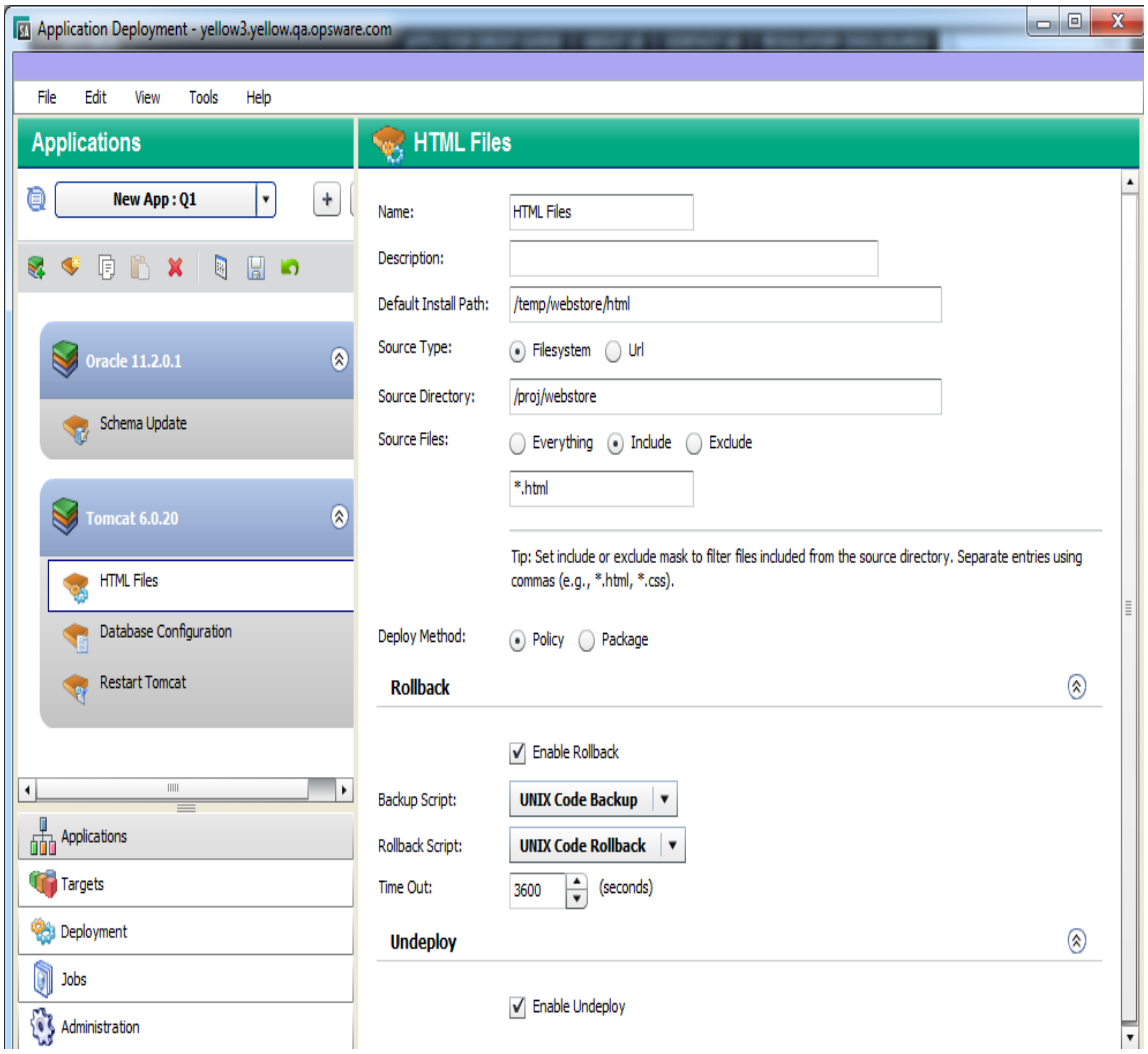
组件表示应用程序的内容，每个组件均会部署应用程序的一部分。

组件使用 Applications 屏幕的右窗格来配置。组件属于表示应用程序结构的特定层。

尽管属性随所配置的组件类型而变化，但创建和访问组件的过程类似。

仅当您有权编辑应用程序时，才可以添加和修改组件。

代码组件示例



所有组件都包含三组属性:

函数属性 (位于窗格顶部)唯一地标识组件、指定其行为并告知应用程序部署可在何处找到此组件包含的任何项。

回滚属性告知应用程序部署在部署失败且需要有序回滚时对该组件采取何种操作。

对于某些类型的组件，在启用回滚时必须指定回滚操作。这是一个脚本、一个流或一组显式指令，具体取决于组件类型。您可能还需要指定备份操作。在这种情况下，需在部署组件前进行备份；如果部署失败，回滚操作将会还原已备份的项。

取消部署属性指示应用程序部署在收到取消部署应用程序的指令时如何卸载此组件。

回滚和取消部署脚本由应用程序部署管理员进行管理 (请参见管理脚本 (第 568 页))。

创建新组件

New Component 对话框用于创建新的应用程序组件。您可以为有权编辑的应用程序创建新组件 (请参见应用程序权限 (第 557 页))。

并与特定层关联。仅当选择层后，才能创建新组件。与层有关的平台决定了您可以创建的组件类型。

要创建新组件，请执行以下操作:

1. 选择将包含新组件的层。可使用两种方法选择层:
   - 单击层名称。
   - 选择该层内的任何组件。
2. 单击 **New Component tool bar**。
3. 在 New Component 对话框中，指定 **Component Name** 和 **Component Type**。组件名称在应用程序中必须唯一。
4. 单击 **OK**。
5. 为新组件的属性填充适当的值。有关详细信息，请参见组件类型 (第 443 页)。
6. 单击“Save”，保存所做的更改。

如果在未保存所做更改的情况下尝试退出 Applications 屏幕，系统将提醒您保存或放弃更改。

修改现有组件

如果您有权编辑某个应用程序，则可修改其组件 (请参见应用程序部署权限 (第 557 页))。在左窗格中选择一个组件后，您可以在右窗格中修改其属性。您可以进行的修改类型取决于组件类型。

要修改现有组件，请执行以下操作:

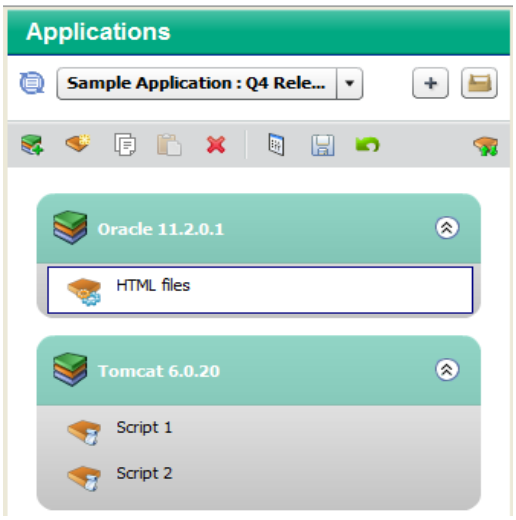
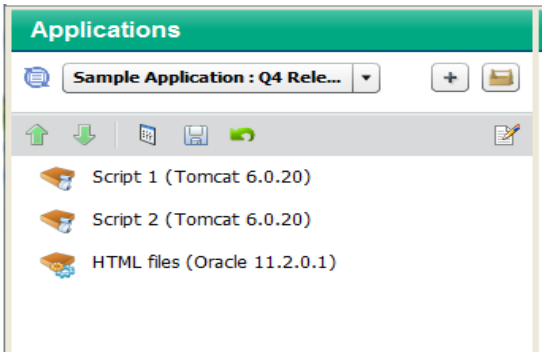
1. 在层中选择该组件。
2. 使用右窗格中的工具修改其属性
3. 单击“Save”，保存所做的更改。

如果在未保存所做更改的情况下尝试退出 Applications 屏幕，系统将提醒您保存或放弃更改。

更改组件的部署顺序

如果您有权编辑应用程序，则可更改将其组件部署到目标的顺序 (请参见应用程序权限 (第 557 页))。为此，您必须将左窗格从层视图 (默认值) 更改为部署顺序视图，如下所示。

组件视图

层视图	部署顺序视图
	

例如，您可能需要在部署应用程序的文件之后启动应用程序。在这种情况下，您需要确保启动应用程序的脚本组件在部署顺序中晚于部署文件的代码组件。

除非更改其顺序，否则将按创建顺序部署组件。

要修改组件部署顺序，请执行以下操作:

1. 单击 **Edit Component Deployment Order**。

   左窗格变为显示与当前应用程序关联的所有组件。它们按部署顺序列出:

2. 选择一个组件，然后使用绿色箭头 ( 和 ) 在列表中向上或向下移动该组件。在此视图中，层显示在组件名称后的括号中。使用箭头按钮移动组件时，它们保留在原始层中。

3. 要返回到层视图，请单击“Edit Properties”按钮: 

在层视图中，您可以将组件从一层拖放到另一层。请注意，某些类型的组件只能用于专为特定操作系统设计的层。例如，注册表组件只能用于 Windows 层。

如果将组件拖放到其他层中，请务必检查部署顺序。默认情况下，组件放置在层的最后。

复制和粘贴组件

您可以复制现有的组件，然后将该组件粘贴到相同或其他应用程序中的兼容层。兼容层在同一操作系统平台上运行。

复制和粘贴组件时，现有组件的所有属性都将复制到新组件。

您每次只能复制和粘贴一个组件，并且只能在有权编辑的应用程序中复制和粘贴组件 (请参见应用程序权限 (第 557 页))。

要复制和粘贴组件，请执行以下操作:

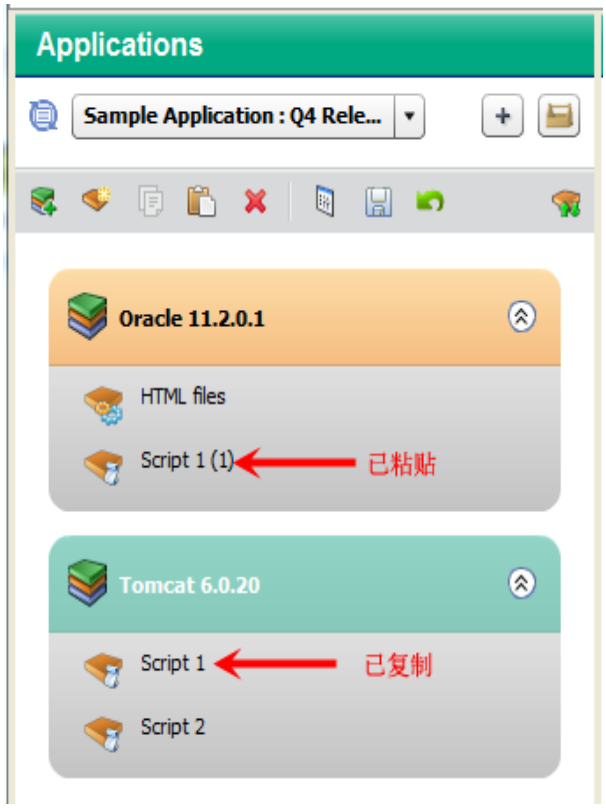
1. 选择要复制的组件。

   这将激活包含此组件的层，然后使 **Copy** 按钮  在工具栏上可用。

   如果 **Copy** 按钮不可用，则您无权编辑此应用程序。
2. 单击工具栏上的 **Copy**。
3. 可选：如果要将组件粘贴到其他应用程序，请打开该应用程序进行编辑。
4. 选择要将组件粘贴到的层。

   如果要粘贴到执行复制的相同层，则无需选择该层。

5. 单击工具栏上的 **Paste**。所粘贴组件的名称会向其附加一个数字，如下图中所示：



如果 **Paste** 不可用，则您无权编辑此应用程序。

6. 单击“Save”，保存所做的更改。

移动组件

您可以使用“拖放”操作将组件从一层移动到相同应用程序中的另一个兼容层。兼容层在同一操作系统平台上运行。

您每次只能移动一个组件，并且只能在有权编辑的应用程序中移动组件 (请参见应用程序权限 (第 557 页))。

要使用拖放操作移动组件，请执行以下操作：

1. 将要移动的组件拖动到兼容层中的组件区域。

   如果新层与原始层兼容，会显示一个绿色加号 。
   如果新层不兼容，或者如果您尝试将组件拖动到相同层中，会显示一个红色的 X 。

2. 将组件拖动到新层中。
3. 单击“Save”，保存所做的更改。

以下示例显示了要从 Tomcat 6.0.20 层移动到 Oracle 11.2.0.1 层的 HTML Files 组件。

将组件移动到不同层的示例

之前：


Oracle 11.2.0.1
HTML files
Tomcat 6.0.20
Script 1
Script 2
脚本 1
组件位于此处

期间：


Oracle 11.2.0.1
HTML files
Script 1
Tomcat 6.0.20
Script 1
Script 2
将其拖动到该兼容层

之后：


Oracle 11.2.0.1
Script 1
HTML files
Tomcat 6.0.20
Script 2
现在，它位于该层

删除组件

您可以删除属于有权编辑的应用程序的组件 (请参见应用程序权限 (第 557 页))。

要删除组件，请执行以下操作:

1. 选择组件。
2. 单击工具栏上的 **Delete Selected Item**。
3. 单击 **Yes** 确认。

请注意，每次只能删除一个组件。

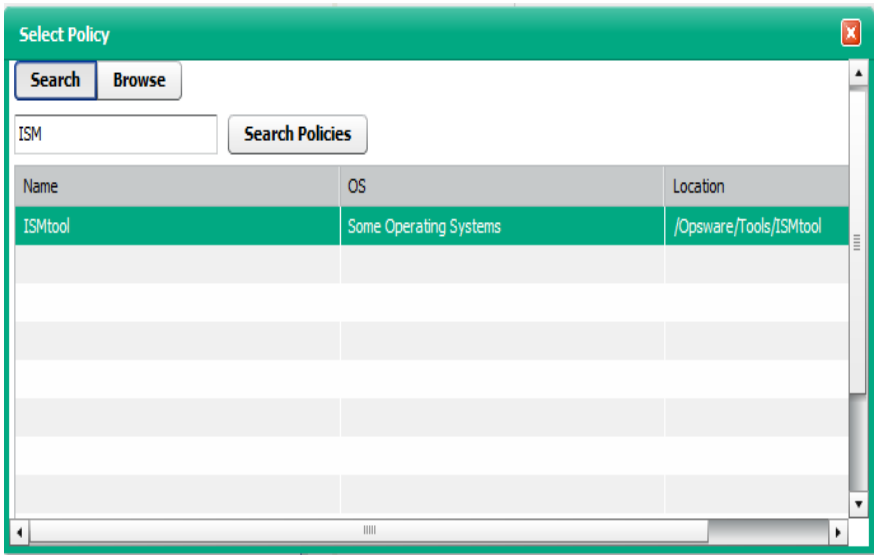
在库中查找并选择项目

以下信息适用于以下类型的组件:

- 应用程序配置组件 (第 451 页)
- 使用组件 (第 464 页)
- 使用组件 (第 464 页)
- 使用组件 (第 464 页)

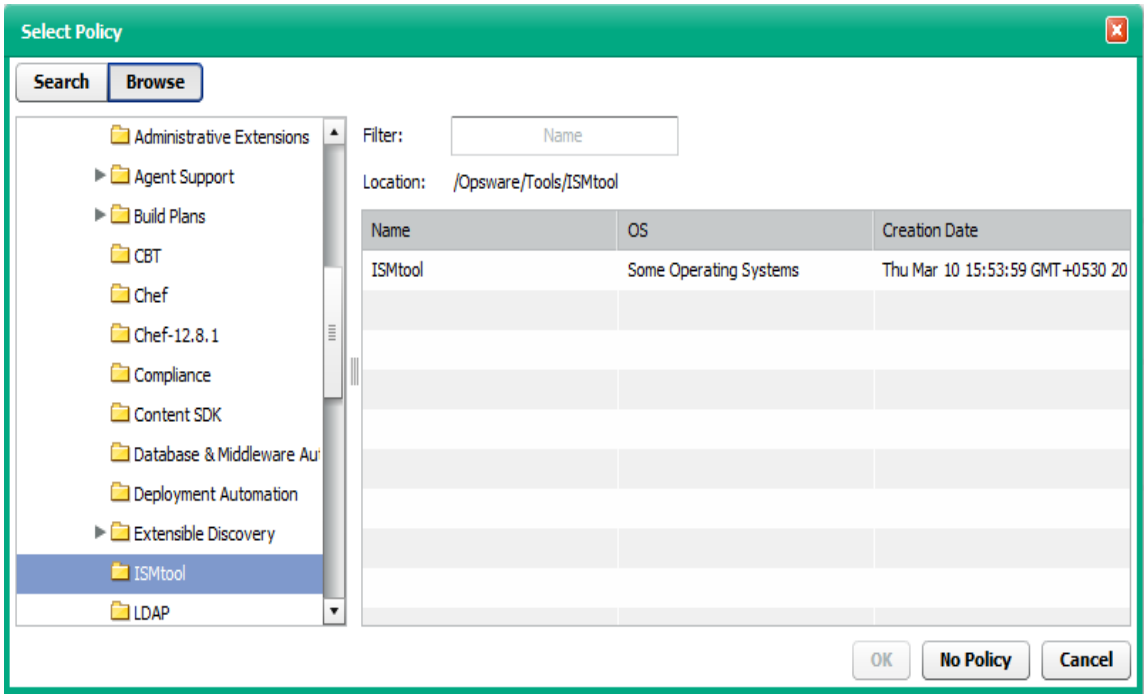
在每种情况下，您可以使用如下所示的对话框在相关库 (SA OO) 中搜索特定项。

Library Item Selection 窗口 - Search 选项卡



在本例中，我们要在 SA 库中搜索名称包含 "ISM" 的策略。您可以使用表视图搜索库 (如上所示)，或者可以浏览库的文件夹层次结构。

Library Item Selection 窗口 - Browse 选项卡



在本例中，搜索结果与 /Opsware/Tools/ISMtool 文件夹中名称包含 "Linux" 的策略匹配。

以下说明假定您在 Applications 屏幕上选择了一个发行版。占位符 "Item" 是指上述库项目之一。

要在库中查找并选择项目，请执行以下操作:

1. 在左窗格中，选择组件类型之一进行编辑 (或创建新组件):
2. 单击项目对应的 **Change** (或者，如果使用程序包组件，则单击  图标)。

   将打开 Select Item 对话框。
3. 可选：如果要指定程序包，请从列表中选择程序包的类型:RPM、ZIP、MSI (仅适用于 Windows 层) 或 All Types。
4. 要使用表视图进行搜索，请执行以下步骤:
   a. 在文本框中，键入要匹配的搜索字符串。要列出库中的所有项目，请将文本框留空。

      如果您正在搜索 OO 流，则可通过在文本框中指定完整字词来提高搜索速度。搜索仅匹配完整字词。例如，在此上下文中，"start" 与 "restart" 不匹配。
   b. 单击 **Search Item**。

   要使用文件夹层次结构进行搜索，请执行以下步骤:
   a. 单击 **Browse** 按钮。您将看到有权查看的所有文件夹。
   b. 可选：指定筛选器。
5. 在表中，选择所需的项目。
6. 单击 **OK**。

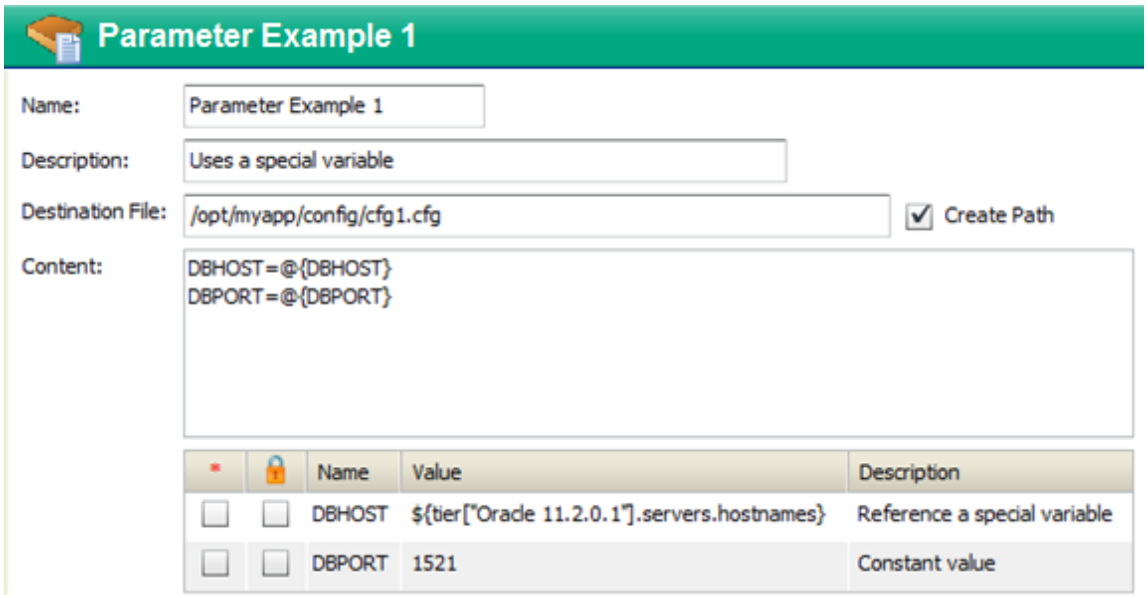
如果要从现有组件中删除库项目，请单击 **No Item** 按钮。此按钮适用于软件策略和 OO 流组件。例如，要删除以前指定的策略，您可以单击 **No Policy**。

参数和特殊变量

在某些类型的组件中，您可以使用组件参数表示可能随应用程序、发行版、版本、目标、层或环境而变化的信息。

在以下配置文件组件中，参数 DBHOST 和 DBPORT 分别用于表示数据库服务器主机名和数据库端口。

参数示例



参数引用的表示法如下:`@{parameter_name}`

通过使用 Content 框中的该表示法引用参数时，参数定义将显示在 Content 框正下方的表中。

- 参数的名称显示在 Content 框中的大括号之间。
- 组件参数名称不能包含以下字符: 空格、\、@、{或 }。
- 如果选中了 * (必需) 列中的框，在部署组件时，参数必须具有值。
- 如果选中了 (已加密) 列中的框，将对参数进行加密，且其值不会显示在 应用程序部署 用户界面中的任何位置。

参数的值可以是一个简单的数字或文本 (例如此示例中的 DBPORT 参数)，也可以是在部署时已分配值的变量。在此示例中，DBHOST 参数引用的变量表示属于指定目标中 Oracle 11.2.0.1 层一部分的服务器的主机名的逗号分隔列表。

您可以使用三种类型的变量指定参数值: 特殊变量、发行版参数和全局参数。每种变量都有特定的用途和功能 (请参见变量类型 (第 476 页))。

当在组件定义中指定某个参数的值时，该值将成为该参数的默认值。您可以在部署时修改该值。这在执行覆盖特定目标或环境的默认值等操作时非常有用 (请参见自定义部署参数 (第 514 页))。

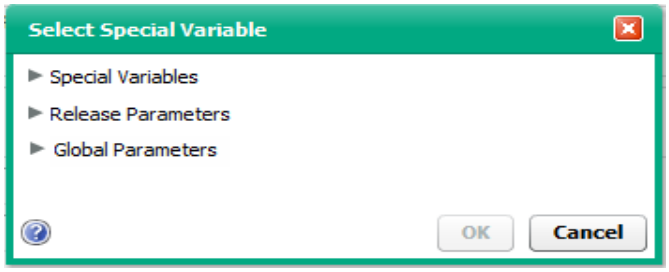
有关创建、管理和引用参数的信息，请参见使用参数和变量 (第 480 页)。有关参数相关规则和约束的信息，请参见特别注意事项 (第 485 页)。

变量类型

您可以在参数定义中引用三种类型的变量:

类型	描述
特殊变量	特殊变量内置到应用程序部署中。它们表示必须在 Deployment 屏幕上指定版本和目标才能确定的信息。 特殊变量与部署内容 (应用程序、发行版或版本)、部署位置 (目标或环境) 或部署者 (SA 用户) 有关。 请参见 特殊变量 (第 477 页)。
发行版参数	发行版参数对发行版中的所有组件可用。它们由应用程序所有者创建和管理，任何有权编辑应用程序的用户均可对其进行修改。除非显式修改，否则发行版参数的值不会更改。 在发行版的特定版本中，发行版参数的值相同。 请参见 发行版参数 (第 478 页)。
全局参数	全局参数适用于所有应用程序。它们由应用程序部署管理员创建和管理。除非由管理员显式修改，否则为全局参数指定的值不会发生更改。 全局参数的实际值可能因环境和目标而异。 请参见 全局参数 (第 479 页)。

所有这三种类型的变量均可通过使用 Select Special Variable 对话框引用:



此对话框可在应用程序部署过程中的不同时间点访问 (请参见访问 Select Special Variable 对话框 (第 484 页))。

在部署时，应用程序部署将通过递归方式跟踪对特殊变量、发行版参数和全局参数的所有引用来解析每个组件参数的值。

例如，假设名为 LogDir 的组件参数定义如下:

Name	Value	Description
LogDir	${release=>LogDir}	References release parameter LogDir

发行版参数 AppDir 和 LogDir 定义如下:

Release Parameters:

Name	Value	Description
AppDir	${global=>AppBaseDir}/ThisApp	References global parameter AppBaseDir
LogDir	${release=>AppDir}/log	References release parameter AppDir

全局参数 AppBaseDir 定义为 /opt:

Global Parameters

Name	Value	Description
AppBaseDir	/opt	All applications are stored under /opt

版本中的默认参数类似如下:

Name	Component	Type	Value
AppBaseDir	-	GLOBAL	/opt
AppDir	-	VERSION	${global=>AppBaseDir}/ThisApp
LogDir	-	VERSION	${release=>AppDir}/log
LogDir	Show Recursive Parameter Resolution	DEPLOY	${release=>LogDir}

假设没有目标或环境覆盖，在部署组件时，组件参数 LogDir 的最终值将为 /opt/ThisApp/log:

Parameters:

Name	Component	Type	Value
LogDir	Show Recursive Parameter Resolution	DEPLOY	/opt/ThisApp/log

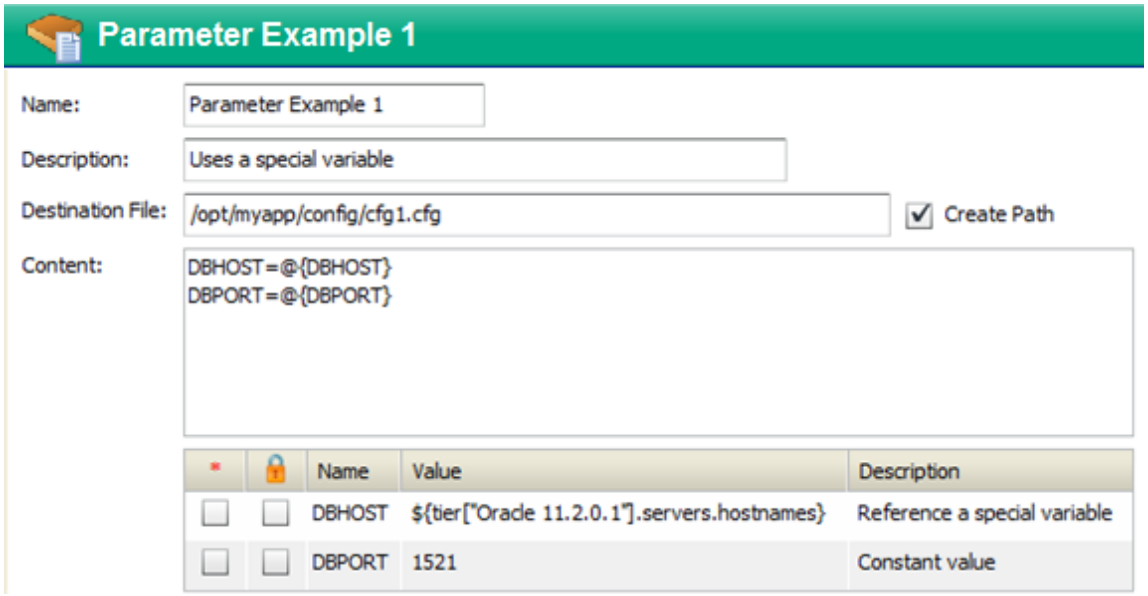
特殊变量

特殊变量表示有关以下项目类型的信息:

- 将部署组件的服务器
- 包含该服务器的层
- 组件所属的应用程序
- 要部署的发行版和版本
- 目标所在的环境
- 启动部署的 SA 用户

在以下示例中，DBHOST 参数引用的变量表示属于 Oracle 11.2.0.1 层一部分的服务器的主机名的逗号分隔列表。

引用特殊变量的参数



尽管之前可能知道某些特殊变量的值，但是其值必须等到部署时才分配 (绑定)。

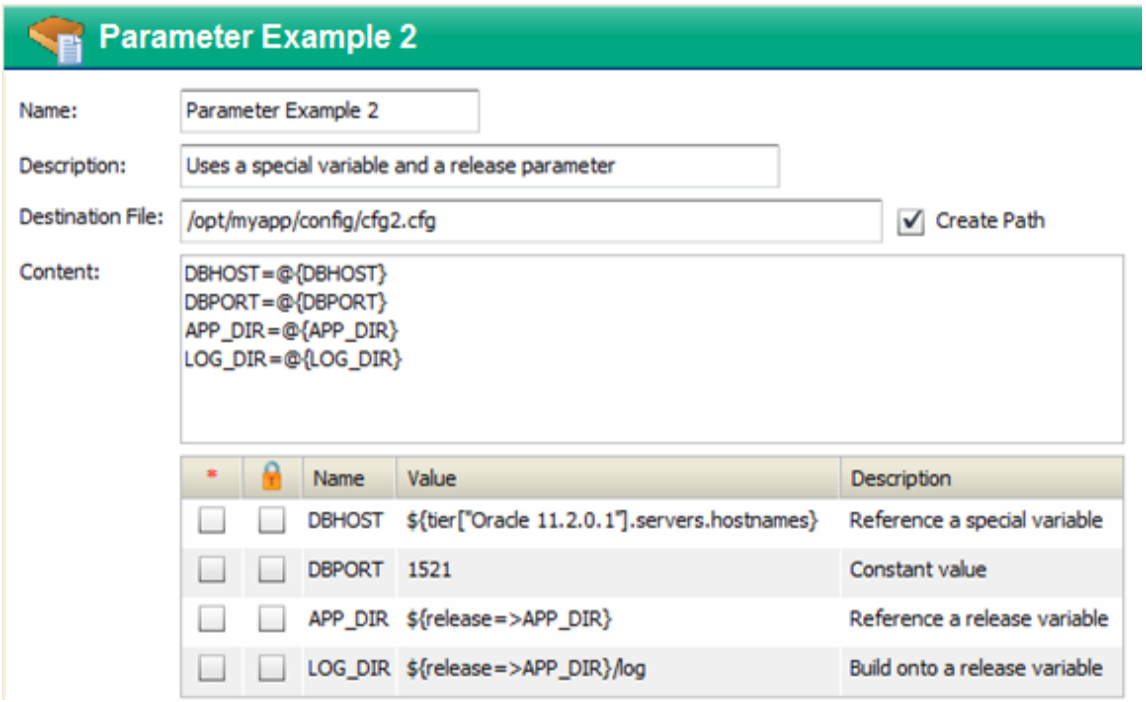
请注意，“Server Name”特殊变量表示 SA 中的服务器名称。可以在 SA 中更改此名称，而无需更改服务器的实际主机名。“Hostname”特殊变量表示服务器的实际主机名。SA 服务器名称可能与主机名相同或不同。

有关在参数定义中引用特殊变量的信息，请参见指定参数的值 (第 482 页)。

发行版参数

发行版参数的值与发行版的特定版本中的值相同。它们可以由发行版中的任何组件引用。在以下示例中，APP_DIR 发行版参数用于表示应用程序的安装目录：

发行版参数示例



您可以通过在发行版参数引用前后添加信息来“构建”参数值。在此示例中，LOG_DIR 参数值将文本 "/log" 添加到 APP_DIR 发行版参数的值。

发行版参数可以引用全局参数、特殊变量和其他发行版参数 (在同一发行版中)。

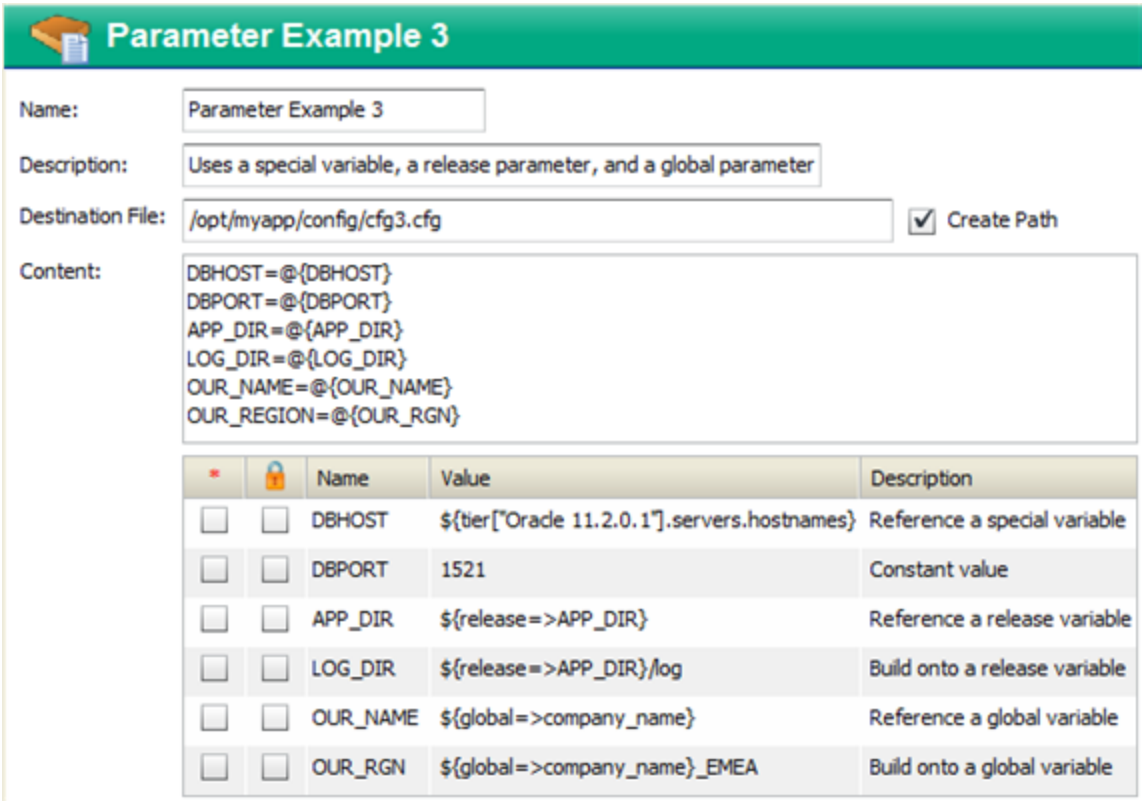
如果您有权编辑应用程序，则可为该应用程序的任何发行版定义或修改发行版参数 (请参见定义发行版参数 (第 483 页))。

从现有发行版克隆新发行版时，也会克隆发行版参数。

全局参数

全局参数在所有应用程序中都相同。它们可以由任何应用程序中的所有组件引用。此外，也可以由发行版参数引用。以下示例使用全局参数 OUR_NAME 表示公司名称:

全局参数示例



与发行版参数一样，您可以通过在全局参数引用前后添加信息来构建参数值。在此示例中，OUR_RGN 参数值将文本 "_EMEA" 添加到 company_name 全局参数的值。

全局参数可以引用特殊变量和其他全局参数。

应用程序部署管理员负责管理全局参数 (有关详细信息，请参见管理应用程序设置 (第 575 页))。

使用参数和变量

本节中的主题提供了“操作步骤”信息帮助您执行以下操作：

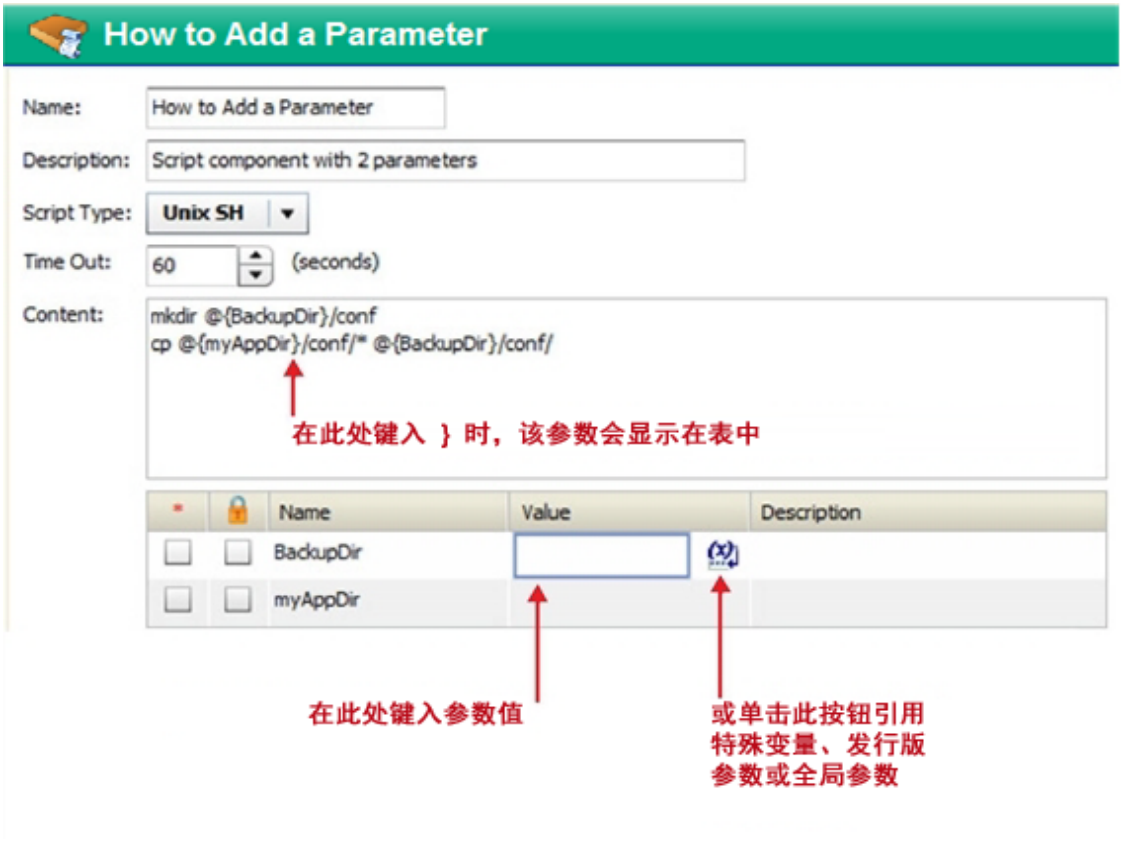
- 指定组件参数 (第 481 页)
- 指定参数的值 (第 482 页)
- 访问 Select Special Variable 对话框 (第 484 页)
- 定义发行版参数 (第 483 页)
- 删除发行版参数 (第 484 页)

指定组件参数

如果您有权编辑 (或创建) 应用程序，则可以向该应用程序包含的脚本或配置文件组件的内容添加参数。

OO 流和应用程序配置组件也可以使用参数。您可以修改这些组件类型的参数值，但是不能添加或删除这些参数本身。

脚本组件中的参数引用示例



要向脚本或配置文件组件的内容添加参数，请执行以下操作:

1. 在 Content 框中，通过使用以下表示法键入参数的名称:
   `@{parameter_name}`

   参数名称不能包含以下字符: 空格、@、\、{ 或 }。
2. 可选：如果在部署组件时，参数必须具有值，请选中 * (必需) 列中的框。

3. 指定参数值。您可以指定常数值和/或引用变量 (有关说明，请参见指定参数的值 (第 482 页))。
4. 可选：如果要对参数进行加密，请选中 (已加密) 列中的框。
5. 单击工具栏上的“Save Release”，保存所做的更改。

指定参数的值

如果您有权编辑应用程序，则可更改其所有组件参数的默认值。您还可以更改其发行版参数的值。您可以指定常数值 (文本或数字)，也可以引用下表中的任何项。

变量引用格式

变量类型	格式	示例
特殊变量	${special_variable_name}	${application.name}
发行版参数	${release=>parameter_name}	${release=>APP_DIR}
全局参数	${global=>parameter_name}	${global=>APP_BASE_DIR}

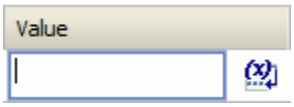
通过使用 Select Special Variable 对话框，可以从特殊变量、发行版参数和全局参数的相关列表中选择变量。

您可以将文本与变量引用组合，如以下示例中所示：

```
${release=>APP_DIR}/log

/var/${application.name}/my_data

${global=>TestBaseDir}/${application.name}
```

要指定参数的值，请执行以下操作：

1. 在参数表中，单击要为其指定参数的 Value 列中的任意位置。Value 列中将显示一个文本框和按钮：



2. 按照以下步骤从列表中选择特殊变量、发行版参数或全局参数引用:
   a. 单击  按钮。将打开 Select Special Variable 对话框。
   b. 选择要引用的特殊变量、发行版参数或全局参数。
   c. 单击 **OK**。现在，变量引用显示在 Value 列中。
3. 可选：指定要包含在参数值中的任何附加文本。
4. 单击工具栏上的 **Save Release**，保存所做的更改。

如果您具有应用程序的 Deploy 权限和相关环境的 Write 权限，则在部署时可以通过在 Parameter Editor 中执行类似的步骤覆盖默认参数值 (请参见自定义部署参数 (第 514 页))。

定义发行版参数

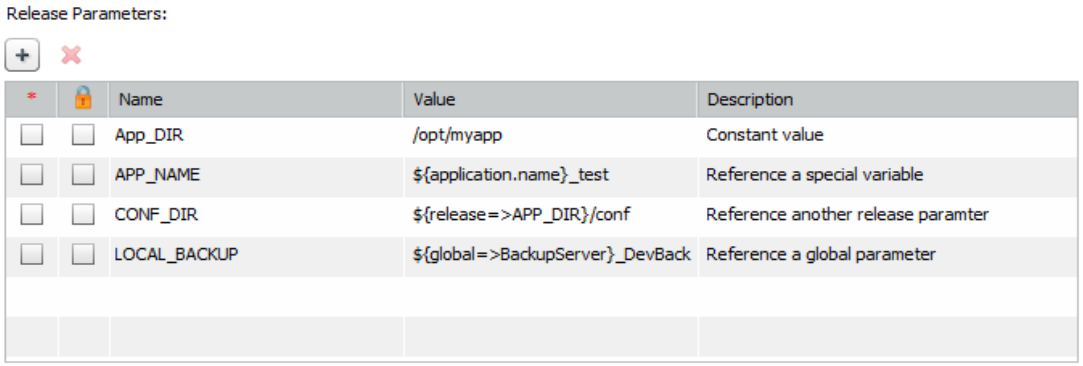
如果您有权编辑应用程序，则可定义或修改其发行版参数。这些参数对发行版中包含的所有组件可用；它们在发行版之外不可用。

要定义或修改发行版参数，请执行以下操作:

1. 在 Application 屏幕上，选择要使用的发行版。
2. 在工具栏上，单击 **Edit release properties**。将打开 Release Attributes 对话框。

   您还可以从 Manage Applications 工具打开此对话框 (请参见管理应用程序 (第 434 页))。
3. 要添加新的发行版参数，请执行以下步骤:
   a. 单击 **Add Release Parameter** 按钮。
   b. 输入新参数的名称。该名称在发行版参数中必须唯一。该名称不能包含以下任何字符:@、\、{或 }。
   c. 单击 **OK**。
4. 单击此参数的 **Value** 列中的任意位置。
5. 指定参数的值 (有关详细说明，请参见指定参数的值 (第 482 页))。

   您可以在值指定中引用特殊变量、其他发行版参数和全局参数。例如：


Release Parameters:

*		Name	Value	Description
☐	☐	App_DIR	/opt/myapp	Constant value
☐	☐	APP_NAME	${application.name}_test	Reference a special variable
☐	☐	CONF_DIR	${release=>APP_DIR}/conf	Reference another release paramter
☐	☐	LOCAL_BACKUP	${global=>BackupServer}_DevBack	Reference a global parameter

6. 可选：要对参数值进行加密，请选中 (已加密)列中的框。
   请注意，在此上下文中，* (必需)列不相关。
7. 单击工具栏上的 **Save Release**，保存所做的更改。

删除发行版参数

如果您有权编辑应用程序，则可以删除未由任何组件参数或其他发行版参数引用的任何发行版参数。不能删除正在使用的发行版参数。

要删除发行版参数，请执行以下操作:

1. 在 Application 屏幕上，选择要使用的发行版。
2. 在工具栏上，单击 **Edit release properties**。将打开 Release Attributes 对话框。
3. 在 **Release Parameters** 表中，选择要删除的参数。
4. 单击 **Delete Parameter**。此时将显示确认对话框。
5. 单击 **Yes** 确认删除。
6. 单击工具栏上的 **Save Release**，保存所做的更改。

访问 Select Special Variable 对话框

可以从应用程序部署用户界面中的多个位置打开 Select Special Variable 对话框。

如何打开 Select Special Variable 对话框

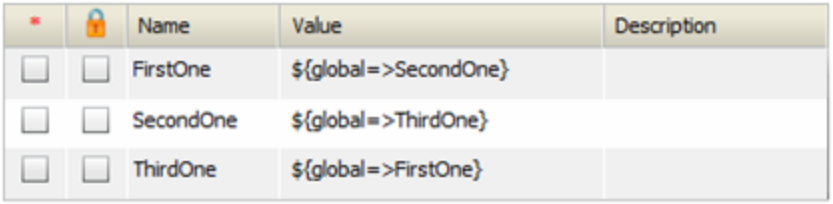
位置	参数类型	说明
Application 屏幕	组件	在组件编辑器中，单击任何组件参数的 Value 列中的任意位置。
发行版	1. 单击 **Edit release properties**。 2. 单击任何发行版参数的 Value 列中的任意位置。	
Deployment 屏幕	任何	1. 展开 Deployment Options 区域的 Parameters 部分。 2. 单击 Parameters 区域左上角的 **Edit** 按钮。将打开 Parameter Editor 对话框。 3. 在最右侧的列中，双击列出的任何参数的值。将打开 Parameter Values 对话框。 4. 单击 Target Value 或 Environment Value 框右侧的  按钮。
Administration 屏幕	全局	1. 转到 Application Settings 页面。 2. 单击任何发行版参数的 Value 列中的任意位置。

特别注意事项

指定参数定义时，请注意以下限制和事项。

周期

不能在参数定义中创建循环引用 (周期)。例如，当您尝试保存发行版时，以下规范会导致出错:



如果在保存应用程序之后创建周期 (例如，如果以后修改全局参数)，您将无法为此发行版的任何新版本创建部署作业。Deployment 屏幕将报告 "infinite loop" 错误。

自引用

不允许参数引用自身。

删除

如果现有版本中的组件或发行版参数引用某个全局参数，则不能删除该全局参数。

如果当前发行版中的组件或发行版参数引用某个发行版参数，则不能删除该发行版参数。

如果全局参数 A 引用全局参数 B，则不能删除全局参数 B。

如果版本包含对已删除全局参数的引用，则不会创建部署作业。

如果保存的发行版包括引用已删除发行版参数的组件，则应用程序工具栏上将显示警告图标  。单击警告图标可确定已删除的参数。

命名

不能在组件参数、发行版参数或全局参数的名称中使用以下字符:@、\、{或 }。

也不能在脚本或配置文件组件的组件参数名称中使用空格。OO 流组件参数名称、发行版参数名称和全局参数名称中允许使用空格。

发行版参数名称在特定发行版的发行版参数列表中必须唯一。

全局参数名称在全局参数列表中必须唯一。

发行版参数的名称可能与组件参数或全局参数相同。

全局参数的名称可能与组件参数或发行版参数相同。

导入和导出

可以导入和导出全局参数和发行版参数 (请参见导入和导出数据 (第 581 页))。

执行部分导入时，将仅导入由一个或多个发行版或组件参数引用的全局参数。

覆盖

为给定发行版参数创建的环境或目标值覆盖还将用于从原始发行版克隆的所有发行版。

全局参数和发行版参数的目标和环境覆盖遵循与组件参数相同的优先顺序。目标覆盖取代了环境覆盖，而环境覆盖取代了默认值。

目标

要查看简单示例，请参见快速入门 (第 414 页)。

概述

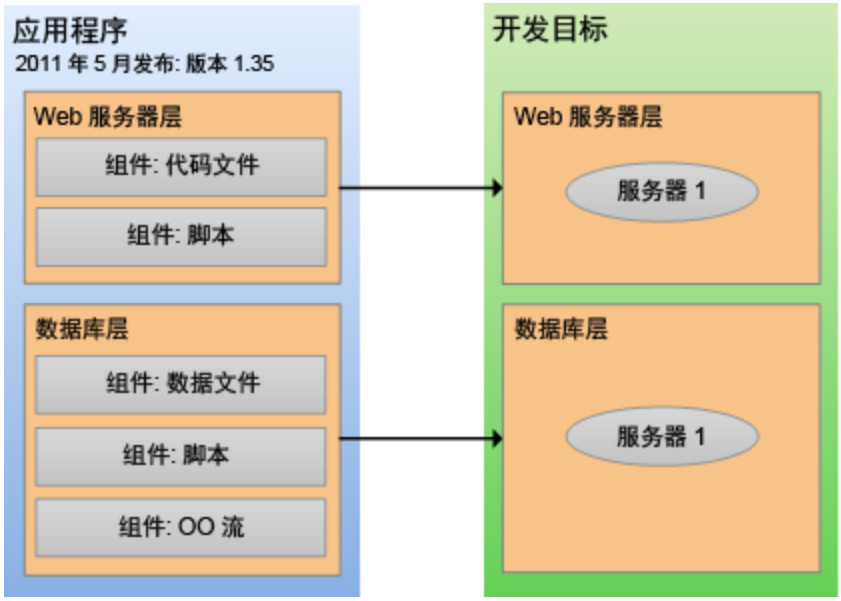
应用程序将部署到托管服务器集。这些服务器集称为目标。每个目标通常适合托管一个或多个应用程序，具体取决于其结构。

目标属于特定环境，例如 QA 或生产。应用程序部署用户界面中使用的环境和目标在 HPE Server Automation 主用户界面中镜像为“设备组”。

目标使用层进行结构化。层提供应用程序的特定部分 (通常是中间件) 所需的功能。层的示例包括 Web 服务器、应用程序服务器和数据库。部署应用程序时，给定应用程序层中的所有组件都将部署到相应目标层中的每台服务器。

SA 软件策略可能与层关联，以确保适当的中间件在层上已就绪，或者随应用程序部署管理的应用程序一起部署。

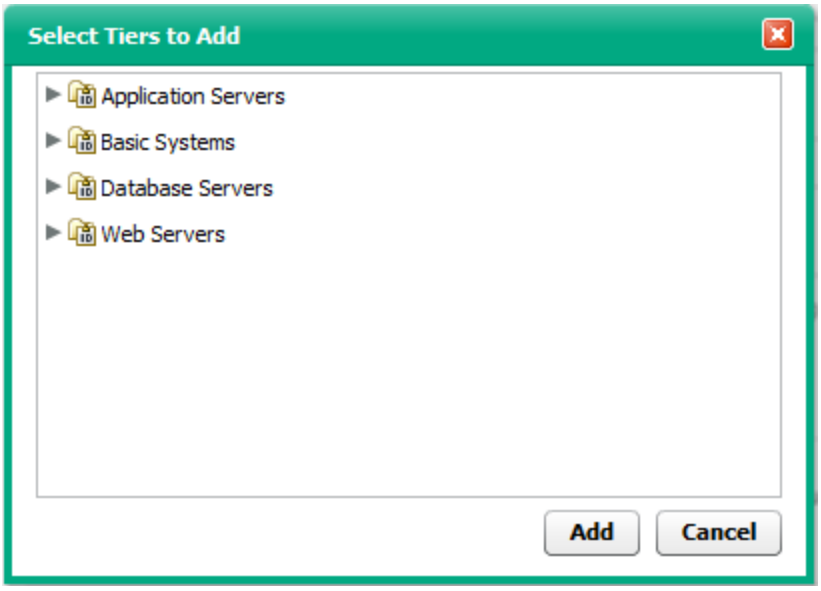
双层目标



如果您在创建目标时考虑到特定应用程序，则可使用该应用程序定义目标的结构。为此，必须已存在应用程序的特定发行版的可部署版本。有关详细信息，请参见部署应用程序 (第 503 页)。

您也可以创建一个新目标并使用现有层进行结构化。SA 提供了以下“预置”层:

提供的层



您可以使用这些层对目标进行结构化，也可以定义您自己的层 (请参见管理应用程序部署 (第 560 页))。

每个目标与一个特定环境关联，例如 QA 或生产。您的环境映射到软件开发生命周期。

您可以在有权编辑的任何环境中构造目标 (请参见设置权限 (第 546 页))。

通过在环境中一致地构造目标，您可以确保测试环境准确地反映生产环境。

一个环境中的目标中所包含的服务器数量可以、而且通常与在不同环境中将同一应用程序部署到的目标中所包含的项目数量不同。

例如，在简单的 QA 环境中，相同的物理计算机 (或虚拟机) 可能会用来运行应用程序服务器和数据库。但在生产环境中，这些责任将分配给多台服务器。可能有一个专用数据库服务器和应用程序服务器群集。

目标的名称在应用程序部署环境中必须唯一。

先决条件

在可以创建或修改目标之前，必须满足以下条件:

- 此目标所需的层可用。
- 填充此目标中的层所需的任何服务器由 SA 管理。
- 您有权在此环境中创建目标。
- 您有权查看要添加的服务器。

有关 SA 权限的详细信息，请参见《SA 10.50 管理指南》。

关于 Targets 屏幕

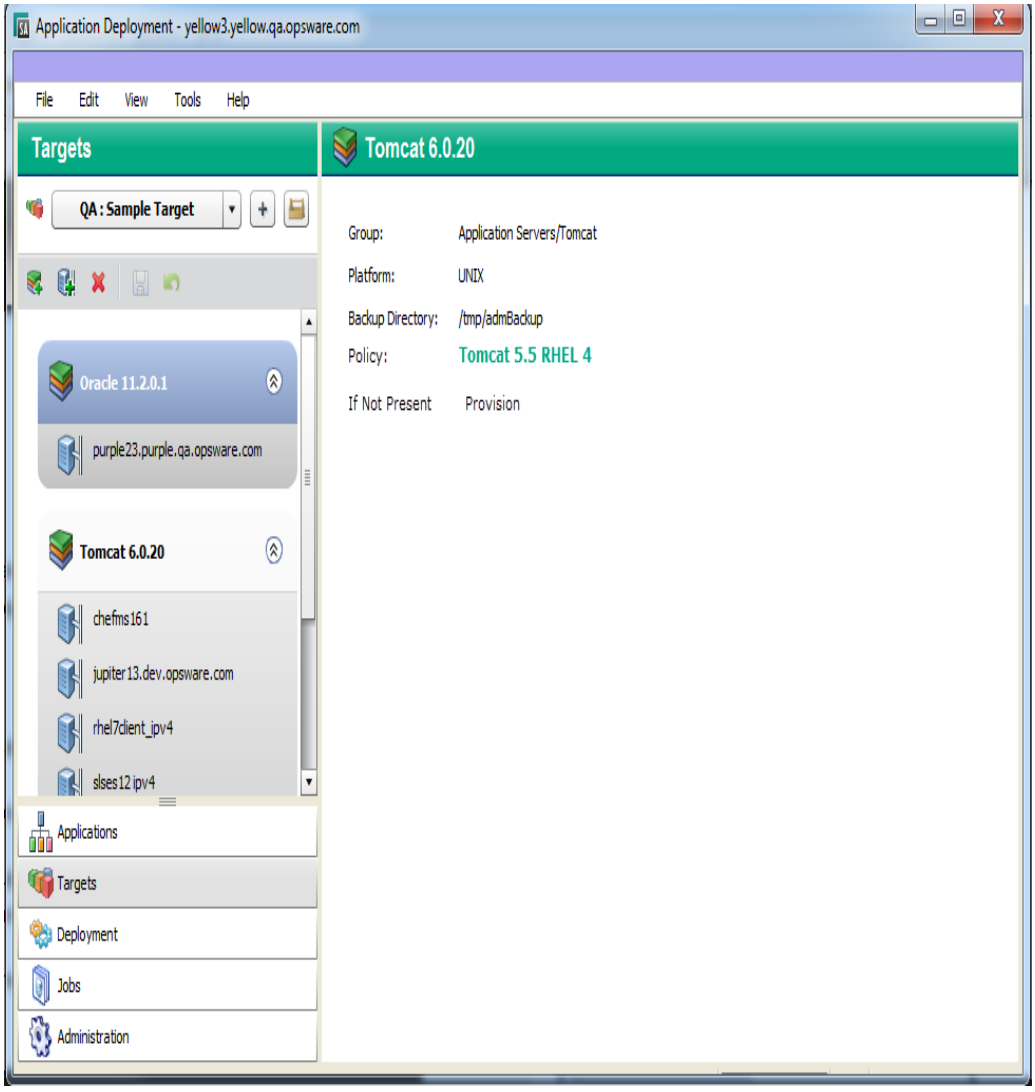
在选择目标之前，Targets 屏幕类似于快速入门 (第 414 页)中所示的图像。下图显示了 Targets 屏幕。选择目标之后，该屏幕如下所示。

注意事项:

- 所有现有目标显示在左上角的下拉列表中。此处选择了 Production:Web Store 目标。
- 左窗格显示目标的结构。

- 右窗格显示有关在左窗格中选择的项目的信息。您可以选择单台服务器或整个层。选择某一项时，该项的背景色将会改变。此处选择了 Oracle 11.2.0.1 层。
- 通过显示在右侧面板中的任何蓝色文本，可以链接到更多详细信息。在此示例中，该链接是与此层关联的策略。如果在左侧面板中选择一台服务器，您将看到该服务器详细信息的链接。

显示现有目标的 Targets 屏幕



使用目标

以下主题提供有关创建和维护目标的基本说明。

- 创建新目标 (第 491 页)
- 管理目标 (第 492 页)

有关更多详细信息，请访问每个主题中提供的链接。

创建新目标

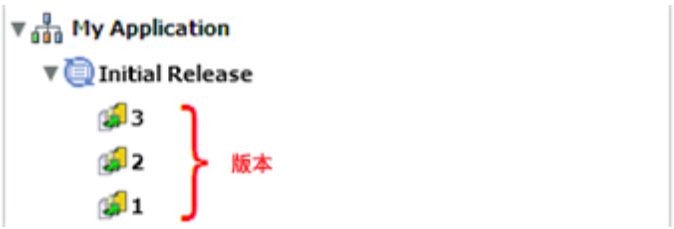
本主题提供有关创建新目标的基本说明。假设您具有创建应用程序部署目标所需的 SA 权限，并有权编辑目标所在的环境。请参见 先决条件。

要创建和配置新目标，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 单击 **Create a New Target**。将打开 Create New Target 对话框。
3. 输入目标名称。名称在环境中必须唯一。
4. 在 Location 下拉字段中，选择将包含此目标的环境。

   可以通过两种方法使用层填充目标: 手动添加层或自动创建层。

   要在目标中自动创建一组用于现有应用程序的相同层，请选择“Use Application's Tiers”，然后选择应用程序的版本。

   请注意，之前必须已创建应用程序的特定发行版的具体版本，它才会显示在 Select Version 下拉列表中。
5. 单击 **OK**。
6. 要将层添加到此目标，请执行以下步骤:
   a. 单击 **Add Tier**。
   b. 选择要添加的层类型。您必须创建应用程序所使用的相同层集 (或超集)。

您的软件附带提供了一组标准层。如果所需的层不可用，则应用程序部署管理员可能需要添加该层。请参见 管理层 (第 560 页)。

c. 单击 **Add**。

d. 对应用程序所需的每个层重复这些步骤。

7. 对于在步骤 6 中添加的每个层，请按以下步骤指定在此层中包含的服务器：
   a. 选择要使用的层。
   b. 向层中添加服务器。
8. 重复步骤 7，将服务器添加到其余的所有层。
9. 单击 **Save Target** 保存所做的更改。

添加服务器

1. 单击 **Add Server**。

   可用服务器的列表按平台进行筛选。请确保为层选择合适的服务器。例如：如果该层的组件专门用于 RedHat EL 5，您应该只选择 Red Hat EL 5 服务器。
2. 在 Add Servers to Target 对话框中，找到并选择要添加的托管服务器。您可以使用 CTRL 键选择多台服务器。

   如果要添加未配置的服务器，请参见在部署时配置服务器 (第 499 页)。
3. 单击 **OK**。

管理目标

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Targets 工具执行某些任务。使用此工具可执行以下操作：

- 创建新目标组。 (第 493 页)
- 删除目标或目标组 (第 494 页)
- 重命名目标或目标组 (第 494 页)

要打开 Manage Targets 工具，请单击 Targets 屏幕上的 **Manage targets** 。如果您无权查看目标，则此按钮不可用。

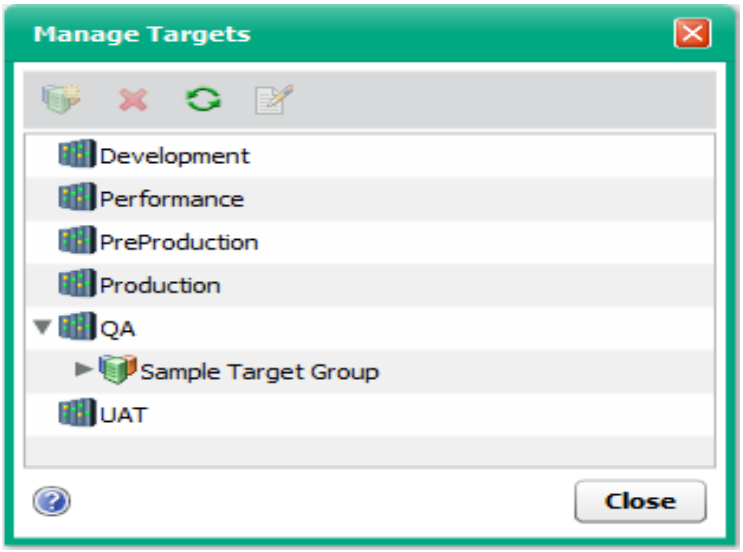
Manage Targets 工具包含一个表，其中列出了以下项：

 环境

目标组 (可以嵌套)

目标

Manage Targets 工具



表中具有“子项”的每个项前都带有箭头。单击 ▶ 箭头可展开项；单击 ▼ 箭头可折叠项。要刷新视图并折叠所有项，请单击“Refresh”按钮 。

要编辑特定项目，请选择包含该项目的行，然后单击工具栏上的“Edit Properties” 按钮，或者直接双击该项目。但只能使用 Manage Targets 工具修改目标或目标组的名称。要修改目标的结构或内容，请返回到 Targets 屏幕。

创建新目标组。

目标组是可用于在层次结构中组织目标的容器。如果您拥有适当的权限 (请参见概述 (第 546 页))，则可创建目标组。

要创建目标组，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 打开 Manage Targets 工具 (请参见管理目标 (第 492 页))。
3. 展开要在其中创建目标组的环境。
4. 要在顶级创建目标组，请选择该环境。要创建嵌套目标组，请选择要在其下创建新组的现有组。

5. 单击 **Create Target Group**。
6. 指定新组的名称。该名称在层次结构中的此容器内必须唯一。
7. 单击 **OK**。现在，表中应当会显示新组。组和目标按字母顺序在层次结构的每个级别列出。

您可以将目标和目标组拖放到同一环境中的不同位置。但是，不能在环境之间移动它们。

删除目标或目标组

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Targets 工具删除目标或空目标组。

要删除现有目标或目标组，请执行以下操作:

1. 打开 Manage Targets 工具 (请参见管理目标 (第 492 页))。
2. 在表中选择包含要删除的项目的行。请注意以下几点:
   - 可以使用 Ctrl 或 Shift 键选择多个项。但是，只能删除多个相同类型的项，例如，多个目标或多个目标组。
   - 如果删除目标，已计划但未运行的任何作业也会被删除。
   - 只能删除空目标组。
3. 单击“Delete”。

系统会询问您是否确定要删除此项 (或这些项)。单击 **Yes** 以删除。

重命名目标或目标组

如果您拥有适当的权限 (请参见概述 (第 546 页))，则可使用 Manage Targets 工具更改目标或目标组的名称。

要重命名目标或组，请执行以下操作:

1. 打开 Manage Targets 工具 (请参见管理目标 (第 492 页))。
2. 在表中选择包含要重命名的目标或组的行。
3. 单击 **Edit Properties**。

4. 输入目标或组的新名称。目标名称在环境中必须唯一。另请参见命名规则 (第 412 页)。
5. 单击 **OK**。

将层添加到现有目标

本主题提供有关将层添加到现有目标的基本说明。本主题假设您具有创建应用程序部署目标所需的 SA 权限，并有权编辑目标所在的环境。有关详细信息，请参见先决条件。

要将层添加到现有目标，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 从 Select Target 下拉列表中，选择要将层添加到的目标。
3. 单击 **Add Tier**。
4. 选择要添加的层类型。您必须创建应用程序所使用的相同层集 (或超集)。

   您的软件附带提供了一组标准层。如果所需的层不可用，则应用程序部署管理员可能需要添加该层。请参见 管理层 (第 560 页)。
5. 单击 **Add** 按钮。
6. 将一台或多台服务器添加到新层 (请参见将服务器添加到现有层 (第 496 页))。
7. 单击 **Save Target** 保存所做的更改。

从现有目标中删除层

本主题提供有关从现有目标中删除层的基本说明。假设您具有编辑应用程序部署目标所需的 SA 权限，并有权编辑目标所在的环境。请参见 先决条件。

要从现有目标删除层，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 从 Select Target 下拉列表中，选择要从中删除层的目标。
3. 选择要删除的层。
4. 单击 **Delete Selected Item**。

5. 单击“是”。
6. 单击 **Save Target** 保存所做的更改。

将服务器添加到现有层

本主题提供有关将服务器添加到现有层的基本说明。假设您具有编辑应用程序部署目标所需的 SA 权限，并有权编辑相关目标所在的环境。请参见 先决条件。

要从现有目标删除层，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 从 Select Target 下拉列表中，选择要使用的目标。
3. 选择要将服务器添加到的层。
4. 单击 **Save Target** 保存所做的更改。

从层中删除服务器

本主题提供有关从现有层中删除服务器的基本说明。假设您具有编辑应用程序部署目标所需的 SA 权限，并有权编辑目标所在的环境。请参见 先决条件。

要从现有层中删除服务器，请执行以下操作:

1. 转到 Targets 屏幕 (单击左下角的 **Targets**)。
2. 从 Select Target 下拉列表中，选择要使用的目标。
3. 选择要删除的服务器。
4. 单击 **Delete Selected Item**。
5. 单击“是”。
6. 单击 **Save Target** 保存所做的更改。

请注意，此过程仅从目标中删除服务器，对服务器本身没有影响。

实时目标

可以通过两种方法创建应用程序部署目标:

- 您可以在部署之前使用 Targets 屏幕上的工具创建这些目标 (如本节前面所述)。
- 您可以在部署应用程序时创建这些目标。这称为“实时”目标创建。

如果要委托分配和/或选择目标资源，实时目标将非常有用。利用实时目标，您可以重点关注应用程序的定义，而不是其部署位置。通过使用实时目标和 HPE Operations Orchestration (HPE OO) 流，您可以制定具体的资源决策并连接第三方分配和计划工具。

应用程序部署使用您提供的 HPE OO 流填充实时目标。在大多数情况下，此流将选择一组现有的托管服务器并将其分配给新目标。而在其他情况下，此流将包含选择一台服务器、在该服务器上执行任何必要的操作和将其交付给 SA 所需的全部逻辑，这样它可以成为有效的托管服务器。

创建实时目标之后，您可以使用管理目标 (第 492 页)下所述的过程管理该目标。

先决条件

在可以创建实时目标之前，您必须执行以下操作:

1. 将 SA 配置为与 HPE OO 集成。
2. 定义一个接受以下参数的 HPE OO 流:

参数	描述
host	SA 核心服务器的主机名或 IP 地址。
username	将创建新目标的 SA 用户的名称 (需要它来确保服务器分配受此 SA 用户的权限约束)。
environment	应用程序部署中将包含新目标的环境的 ID。
target	该流将服务器添加到的新目标的 ID。
tierCounts	包含层和服务器计数参数的分隔字符串。 格式为 `tier,count;tier,count...` `tier` 是该流将服务器添加到的层的 ID。 `count` 是新目标 (克隆或从头开始创建) 中的服务器数量。

3. 在应用程序部署用户界面中的 Administration 屏幕上，将此 HPE OO 流指定为实时目标流。

要创建实时目标，您必须对目标所在的环境具有“编辑”权限。

创建实时目标

要创建实时目标，请执行以下操作：

1. 选择左下角的 **Deployment** 按钮。
2. 选择要部署的应用程序版本。有关详细信息，请参见 部署特定版本 (第 512 页)。
3. 单击 Select Target 下拉列表右侧的 **Advanced Target Selector**。将打开 Advanced Target Selection 对话框。
4. 选择 **Just-In-Time Target**。
5. 单击 **Continue**。将打开 Create New Just-In-Time Target 对话框。在此对话框中，指定以下信息：
   a. Name - 键入该目标的名称。
   b. Location - 选择该目标所在的环境或目标组。
   c. Based On
      - 选择 Version 可使用所部署的版本包含的相同层创建该目标。
      - 这将选择基于该版本的层，从头开始创建一个真正的“按需”目标。
      - 选择 Target 可通过复制现有目标创建该目标。

      这将创建一个“克隆”目标，它使用另一个目标的层结构作为模板。如果有时需要“灵活调整”资源，这将非常有用。例如，如果您经营一家网店，则可能希望在假日购物旺季扩增接待能力。
   d. 如果您在上一步中选择了 Target，则从下拉列表中选择现有目标。
   e. 在表中，指定将包含在每层中的服务器数量 (双击 New Size 列中的值)。
6. 单击 **Create**。应用程序部署将创建新目标，并调用实时目标流来用服务器填充该目标。此流在 Administration 页面上指定 (请参见管理应用程序设置 (第 575 页))。

   如果应用程序部署无法创建目标，将提供一条错误消息。
7. 单击 **Close** 返回到 Deployment 屏幕。

   现在，新目标已自动选择。单击 **Start Job** 可启动部署。

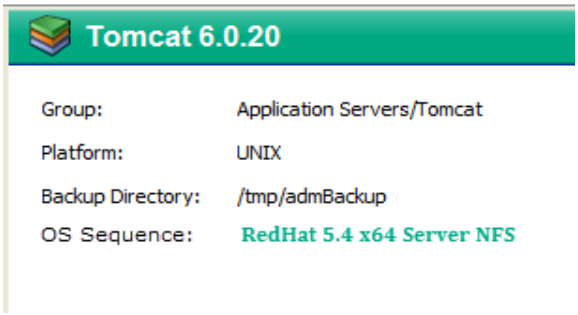
在部署时配置服务器

通过应用程序部署，您可以在部署时通过使用 OS 序列在未配置的服务器上安装操作系统 (OS)。

OS 序列用于定义要在未配置的服务器上安装的内容，包括来自安装配置文件的 OS 构建信息和选定的软件策略。

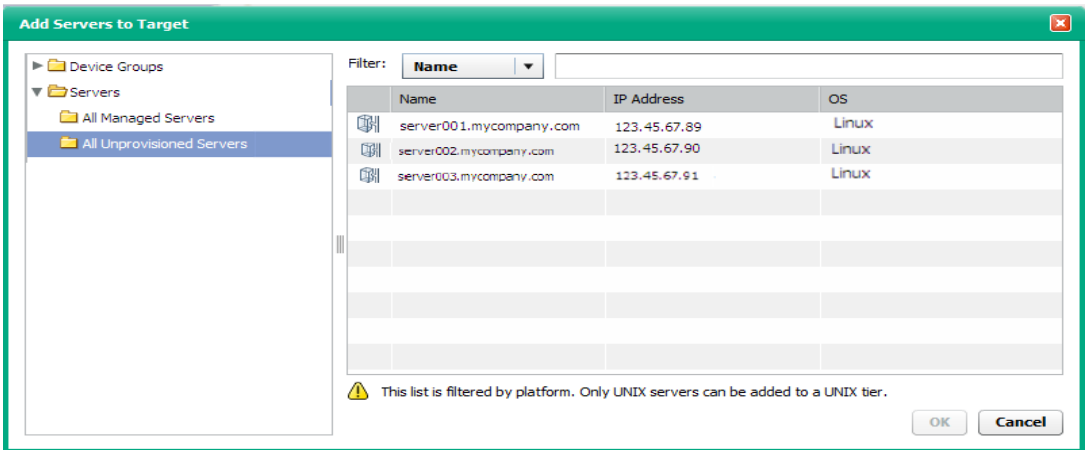
在部署时配置服务器需要执行三个基本步骤：

1. 应用程序部署管理员将 OS 序列 (以及任何必需的中间件) 与特定层关联：



   有关详细信息，请参见 管理层 (第 560 页)。

2. 在定义目标时，将未配置的服务器池中的一台或多台服务器添加到目标中的此层：

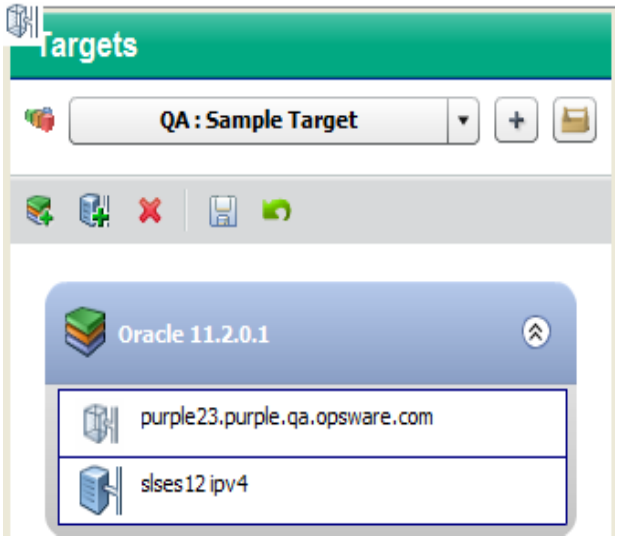


3. 要查看 All Unprovisioned Servers 文件夹，您必须具有 SA 核心的“服务器池”权限。SA 管理员必须向您授予此权限 (请参见《SA 10.50 管理指南》中的“权限参考”一节)。

   您也可以通过浏览 Device Groups 文件夹来查找未配置的服务器。

   - 作为 SA 微代理启动的未配置服务器具有空心图标 。
   - 托管服务器具有实心图标 。

   这些图标显示在 Add Servers to Target 对话框 (如上所示) 中以及 Targets 屏幕和 Deployment 屏幕的左窗格中:



4. 将应用程序的版本部署到目标时，应用程序部署可识别目标中未配置的一台或多台服务器。在暂存时，它根据 OS 序列中指定的信息将指定 OS 部署到这些服务器。

先决条件

在应用程序部署可以将操作系统部署到未配置的服务器之前，必须满足以下条件:

- 服务器记录必须位于 SA 可用生命周期中并处于 SA 无法访问的状态。这可以使用非 OGFS PXE 或非 OGFS CD 启动完成。

  有关详细信息，请参见《SA 10.50 用户指南》中的“代理管理”一节。

- 要查看应用程序部署用户界面内部的 All Unprovisioned Servers 文件夹，您必须具有 SA 核心的“服务器池”权限。
- SA 核心上的库必须包含可用于适当配置服务器的 OS 序列。

  负责将 OS 序列与层关联的用户 (假定是应用程序部署管理员) 必须具有以下 SA 权限:
  - 管理 OS 序列功能权限
  - 对保存 OS 序列的文件夹的“文件夹读取”权限

  这些权限必须来自单个 SA 用户组。

  但在部署时，无需这些权限即可配置服务器。
- 此服务器所属的层必须具有与之关联的 OS 序列。

无法使用此过程配置通过 OGFS PXE 或 OGFS CD 启动功能启动的服务器 (在此版本的 SA 中)。有关详细信息，请参见 SA 联机帮助中的“OS 配置”。

配置工作原理

部署应用程序的版本时，应用程序部署将在暂存期间执行以下步骤:

1. 运行与环境关联的部署前流 (如有)。
2. 运行与指定目标中的层关联的 OS 序列 (如有)。这将在该目标中的任何未配置服务器上安装和配置指定 OS。
3. 附加并修正与指定目标中的层关联的任何策略。

然后，应用程序部署将部署该版本。

您可以将未配置的服务器包含在传统定义的目标和实时目标中。

请注意以下事项:

平台匹配

您只能将构建代理类型正确的未配置服务器添加到层。例如，只能将包含 Linux 构建代理的服务器添加到具有 Linux OS 序列的层。同样，只有 Solaris 构建代理可以安装 Solaris。

OS 序列的存在

只能将未配置的服务器添加到具有 OS 序列的层。如果将未配置的服务器添加到某层，随后从该层中删除 OS 序列，应用程序部署将无法创建部署作业，并将显示一条错误消息。

冲突

有时可能会发生 OS 序列冲突。如果单台服务器包含在两个不同的层中，并且这些层具有不同的 OS 序列，则应用程序部署不知道将服务器安装在哪个 OS 上。在这种情况下，应用程序部署将无法创建部署作业，并将显示一条错误消息。

回滚和取消部署

回滚和取消部署不适用于 OS 序列。如果回滚或取消部署作业，在部署该作业时未配置、随后使用 OS 序列配置的任何服务器仍将是托管服务器。您必须先显式停用服务器，然后才能将其返回到未配置的池。

不重新配置

如果部署到的目标具有一个包含已配置和未配置服务器的层，并且 OS 序列与该层关联，则 OS 序列将仅应用于未配置的服务器。换句话说，不能使用此过程重新配置服务器。

服务器重命名

根据服务器的配置方式，其名称可能在使用 OS 序列进行配置时发生更改。例如，通常情况下，裸机将具有通用名称，并且 OS 序列可以分配一个更有意义的名称。同样，在停用以前托管的服务器并将其返回到未配置的池时，其名称可能在使用 OS 序列重新进行配置时发生更改。

尽管作业日志将反映发生的任何重命名操作，但是应用程序部署 Jobs 屏幕起初可能不显示正确的服务器名称。您可能需要关闭并重新打开 Jobs 屏幕才能看到新名称。

配置失败

如果 OS 序列安装出于任何原因而失败，则部署作业将失败，并且服务器的状态将设置为 Provision Failed。如果发生这种情况，必须先停用 (或删除) 服务器，然后在该服务器上重新安装 SA 代理。有关说明，请参见《SA 10.50 用户指南》中的“代理管理”一节。

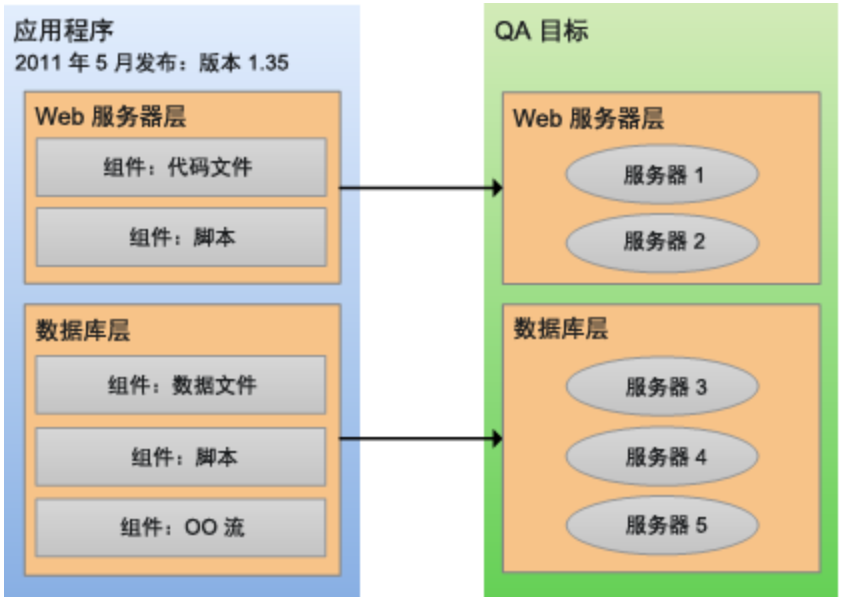
部署应用程序

要查看简单示例，请参见快速入门 (第 414 页)。

概述

在应用程序部署中，您可以将应用程序的特定版本部署到目标。应用程序的组件将部署到指定目标的相关层中的所有服务器。这些组件按 Deployment Order 视图中显示的顺序部署 (请参见组件 (第 442 页))。除非配置了滚动部署，否则必须先完成所有服务器，然后才能部署下一个组件。

将应用程序部署到目标



组件的部署过程分为四个阶段：

1. 将任何必要的文件从 SA 核心复制到目标服务器上的临时位置。这称为“暂存”。对于软件策略组件以及使用“策略”部署方法的代码和程序包组件，将在暂存期间附加策略。
2. 应用程序部署将等到指定的“切换”时间。
3. 如果在组件中启用了回滚，则会在定义层时指定的位置中备份目标服务器上的相关文件和目录 (请参见管理应用程序部署 (第 560 页))。
4. 执行部署组件所需的操作。此操作的性质取决于组件类型:

- 对于代码组件:
  - 如果部署方法是 Package，则从暂存的 ZIP 文件中提取这些文件，并放入目标服务器上的指定位置。
  - 如果部署方法是 Policy，则修正在暂存期间附加的策略。
- 对于脚本组件，在目标服务器上执行指定脚本。
- 对于配置文件组件，将在目标服务器上创建该文件。
- 对于软件策略组件，将修正在暂存期间附加的策略。
- 对于程序包组件，将发生以下两种情况之一:
  - 如果部署方法是 Package，则程序包安装在目标服务器上。
  - 如果部署方法是 Policy，则修正在暂存期间附加的策略。
- 对于应用程序配置组件，应用程序配置将部署到目标服务器。
- 对于 OO 流组件，该流在 HPE Operations Orchestration 服务器上运行至完成。
- 对于 Windows 注册表组件，添加或删除了相关的注册表项或值。

对于一个或多个指定目标中的每台服务器的选定版本中包含的每个组件，应用程序部署创建的部署作业将执行这些步骤。可以在 Jobs 屏幕上查看作业的进度和状态。

如果任何组件无法正确部署或取消部署作业，则部署将自动从故障点回滚。对于启用了回滚的每个组件，将执行该组件中指定的回滚操作，并删除存储在临时目录中的任何暂存文件。

您也可以从 Jobs 屏幕手动回滚部署。回滚操作会将服务器恢复到在该部署之前所处的状态。

回滚的替代方法是“取消部署”。取消部署过程将卸载版本并删除它在服务器上的任何痕迹。这个过程的性能比回滚更高。它不会尝试将服务器恢复到其先前状态。

部署版本时，您可以指示应用程序部署自动取消部署以前版本。在这种情况下，将创建两个作业: 以前版本的取消部署作业和当前版本的部署作业。

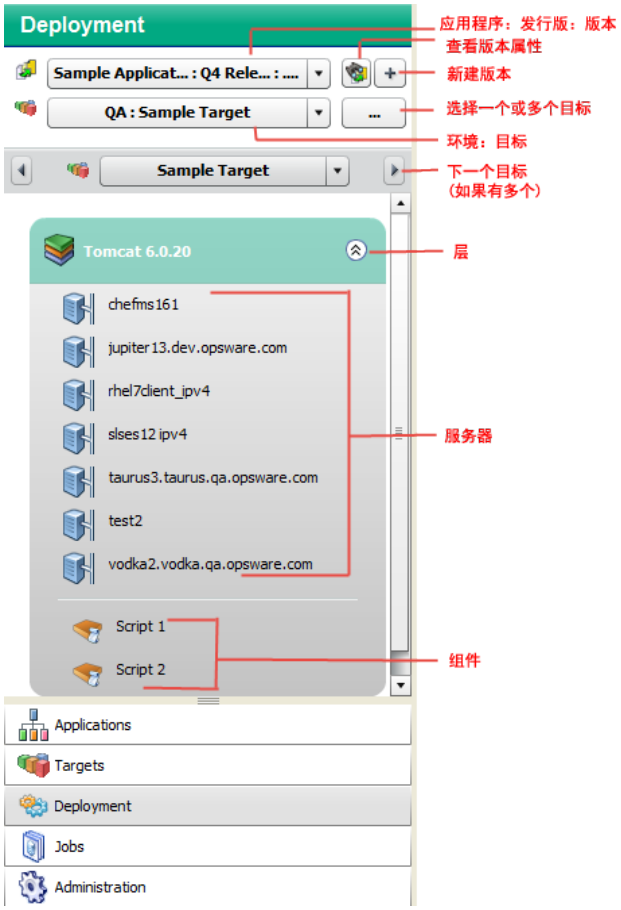
应用程序部署支持滚动部署。这意味着您可以将特定层中的组件部署到该层中的一部分目标服务器，同时保持应用程序在该层中的其余服务器上处于运行 (实时) 状态。

- 您可以自定义组件在部署时使用的参数。如果要覆盖特定目标或环境的默认值或者直接对特定部署使用不同值，这将非常有用。
- 自 SA 版本 9.01 起，SA 支持“实时”目标，即：在部署时使用提供的 HPE OO 流创建的目标。有关详细信息，请参见目标 (第 487 页)。

Deployment 屏幕

Deployment 屏幕使您能够将发行版的特定版本部署到一个或多个目标。Deployment 屏幕的左侧面板表示结构视图。

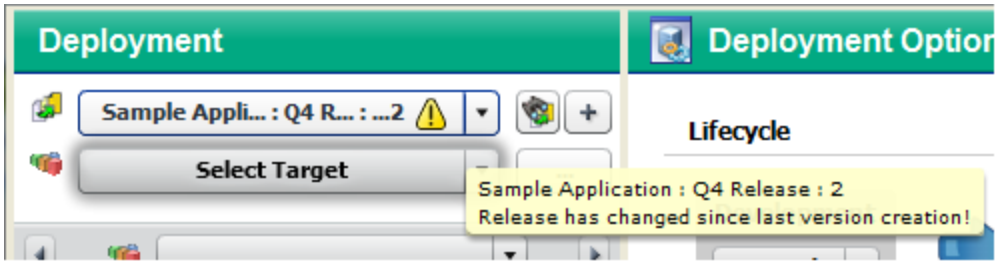
Deployment 屏幕的左侧面板



在此示例中，选择了单个目标 (Advantage123)，并且每层只有一台服务器。您可以选择多个部署目标，但所有目标必须位于同一环境中。

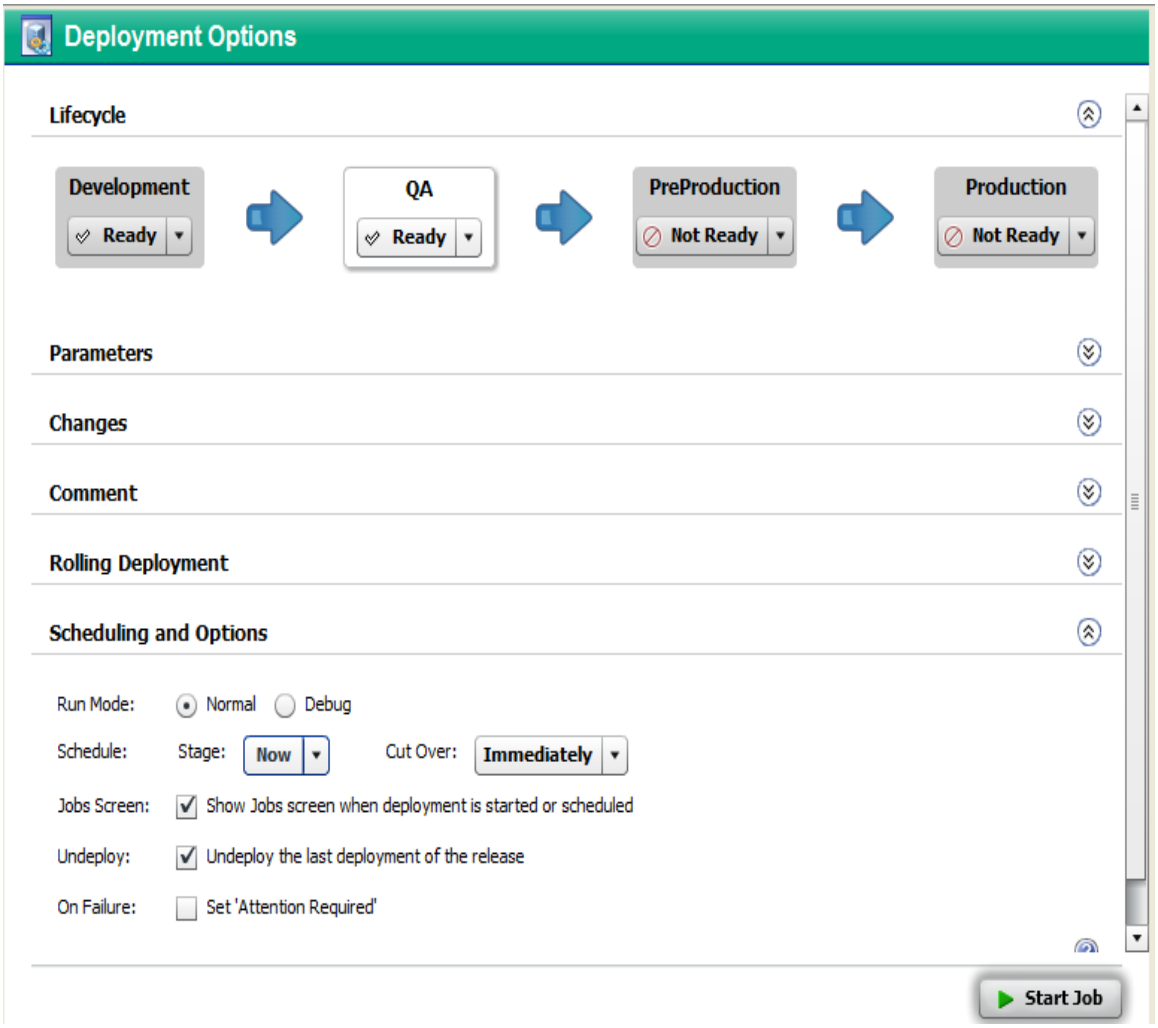
“警告”符号 ⚠ 在此处可见，因为 Green Checking 1.0 发行版自 RC 3 版本创建后已发生更改。出现此图标时，将光标悬停在其上方可查看详细信息。

与警告符号关联的工具提示



Deployment 屏幕的右侧面板显示内容视图，即，特定于此部署作业的设置。

Deployment 屏幕的右侧面板



右侧面板中的以下部分显示单击 **Start Job** 按钮时将创建的部署作业的信息：

- Lifecycle
- Parameters (第 507 页)
- 变更 (第 508 页)
- Comment (第 509 页)
- Rolling Deployment (第 509 页)
- Scheduling and Options (第 510 页)

要展开一个部分，请单击 图标 (或双击蓝色条)。

Lifecycle

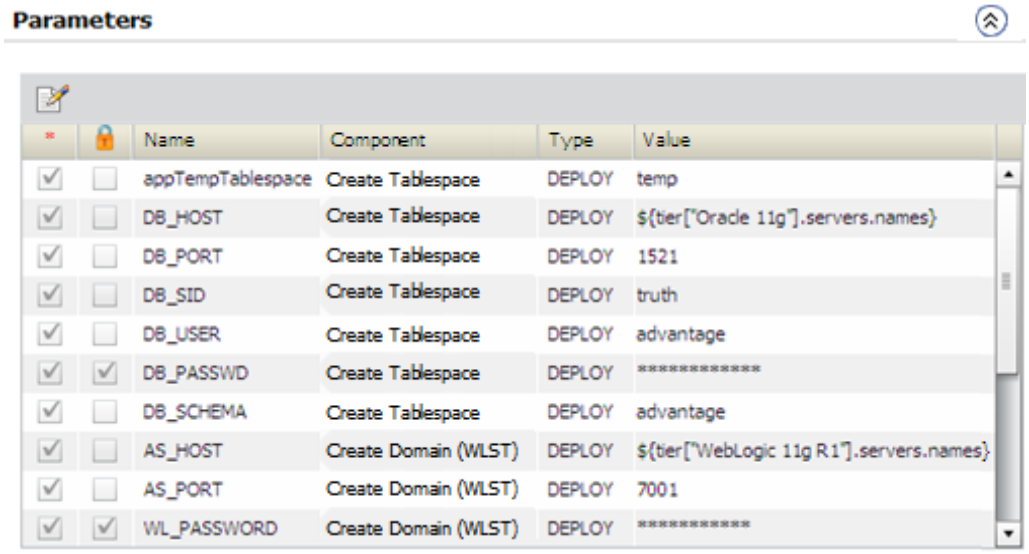
该面板顶部的 Lifecycle 部分显示了此生命周期中包含的所有环境的状态。要部署到的环境将突出显示。

只能部署到没有标记为 Not Ready 的环境。应用程序部署管理员或环境所有者将确定哪些环境可用。对此环境具有 Deploy 权限的任何用户均可在部署时更新环境的状态。

Parameters

Parameters 部分允许您查看和自定义应用程序组件所使用的任何可编辑参数的值。在部署时，应用程序部署将检查并确保所有需要的参数具有值，如果没有，则不会启动部署作业。

Deployment Job 屏幕上的参数信息



在此示例中，这些参数已在应用程序的组件中指定值，这些是“应用程序默认”值。DB_HOST 和 AS_HOST 参数使用“特殊变量”，并从各自层中服务器的名称获取其值。特殊变量是直到部署时才能知道的数据的占位符 (如果选择目标服务器)。

您可以在部署时覆盖特定目标或整个环境的应用程序值 (请参见自定义部署参数 (第 514 页))。

变更

Changes 部分显示与此发行版关联的代码组件文件发生了哪些更改。对于完全发行版，将列出所有文件 (如下所示)。对于增量发行版，仅列出自发行版的“基线”版本部署后添加或修改的任何文件。

Deployment Job 屏幕上列出的代码组件更改

Comment

Comment 部分可供您输入文本注释。这将显示在报告和 Job Logs 中。

Rolling Deployment

Rolling Deployment 部分使您能够确定为了方便部署，每层中有多少服务器同时脱机。Number of Servers per Group 设置用于控制这一点。默认情况下，该设置是层中的服务器数量。默认情况下，所有服务器将同时部署。

例如，编写的应用程序应在更新时从负载平衡池中删除服务器，然后重新添加这些服务器。这可确保在更新滚动批处理时，大多数服务器还能继续运行。

如果某一层共有九台服务器，您可以选择先将相应应用程序层中的每个组件部署到其中的三台服务器，然后再从剩下的六台服务器中选择三台服务器执行相同操作。在这种情况下，您应将 Number of Servers per Group 设置为 3。

如果使用 Rolling Deployment，则同一层中的几组组件将部署到该层中指定数量的服务器中。如果不使用 Rolling Deployment，除非前一个组件已成功部署到层中的所有服务器，否则不会部署组件。

Scheduling and Options

Scheduling and Options 部分允许您指定如何以及何时执行部署，包括用于调试的选项 (请参见管理应用程序部署作业 (第 525 页))。

部署作业选项

选项	用途
Run Mode	确定部署作业应正常运行还是在调试模式下运行。如果在此处选择“Debug”，则 Debugger 窗口将在您单击 **Start Job** 按钮时打开。
Stage	确定应用程序部署应在何时将部署所需的任何文件从 SA 核心复制到目标服务器上的临时目录。 当 Run Mode 为“Debug”时，此选项将自动设置为“Now”。
Cut Over	确定应在目标上部署组件的实际时间。在指定的切换时间，将执行相关文件和目录的备份 (如果已启用 Rollback)，然后部署组件。 计划切换时间时，请务必小心。例如，如果暂存需要 20 分钟才能完成，则计划的切换时间至少应在 30 分钟后。应用程序部署不允许您计划的切换时间早于暂存时间。 如果指定“Immediately”，则备份和部署步骤将在文件暂存后立刻开始。 当 Run Mode 为“Debug”时，此选项将自动设置为“Immediately”。
Jobs 屏幕	确定是否在您单击 **Start Job** 按钮后显示 Jobs 屏幕。 如果选中“Show job status when deployment starts”框，则将打开 Jobs 屏幕。应用一个筛选器，以便只有部署作业可见。您可以在执行部署中的每一步时观察 Job Logs。 如果不选择此选项，Deployment 屏幕将保持显示状态。 当 Run Mode 为“Debug”时，会自动选择此选项。
Undeploy	在执行当前部署之前，确定是否已取消部署此版本的以前部署。 如果未选中“Undeploy the last deployment of this release”框，应用程序部署不会尝试取消部署以前的部署。 当 Run Mode 为“Debug”时，此选项不可用。
On Failure	确定如果作业失败，是否应在回滚之前将部署作业置于 Attention Required 状态。 当部署作业失败时，将发生以下两种情况之一： 如果未选择此选项，应用程序部署会将作业状态设置为 Failed 并执行自动回滚。

部署作业选项(续)

选项	用途
	如果已选择此选项，作业将处于已挂起状态 (Attention Required)，从而使您可以检查托管服务器的状态，以便排除故障。 收集了足够的故障信息之后，您可以恢复作业。然后，应用程序部署将作业状态设置为 Failed 并启动自动回滚。 当 Run Mode 为“Debug”时，会自动选择此选项。

Start Job 按钮

同时选择版本和目标之后，**Start Job** 按钮将变为启用状态。单击该按钮，可根据您指定的选项计划或启动部署作业。

如果任何所需的参数没有值，或者如果版本和目标不匹配 (例如，组件为 Red Hat Linux，但是目标服务器正在运行 SUSE Linux)，则不会启动部署作业。

部署版本

部署应用程序时，应部署该应用程序的特定版本。版本与特定发行版相关联。部署版本需要执行两个步骤:

- 创建新版本 (第 511 页) (或使用现有版本)
- 部署特定版本 (第 512 页)

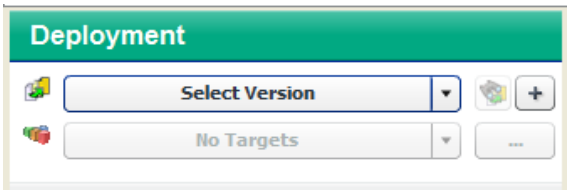
如果您具有环境的 Deploy 权限，则可部署您有权部署的任何应用程序 (请参见概述 (第 546 页))。

创建新版本

如果您有权部署应用程序 (请参见概述 (第 546 页))，则可创建该应用程序的特定发行版的新版本。

要创建应用程序的版本，请执行以下操作:

1. 选择左下角的 **Deployment** 选项卡。
2. 在 Select Version 下拉字段中，导航到您的应用程序。



3. 根据应用程序的相关发行版，单击 **Create New Version** 链接。
4. 在 Create New Version 窗口中，输入版本名称 (或版本号)。

   版本名称就是您指定的字符串。但是，如果此发行版的以前版本包含数字，则应用程序部署会自动增加该数字。例如，如果以前版本是 "1"，则 Version 字段中将显示 "2"。如果以前版本是 "V 1.2.1"，则 Version 字段中将显示字符串 "V 1.2.2"。

   无论应用程序部署在 Version 字段中输入的内容如何，您均可指定所需的任何版本名称。但是，版本名称在发行版中必须唯一。
5. 可选：在 Description 字段中，输入要为此版本指定的任何其他信息。
6. 可选：如果要查看自前一版本创建以来更新的代码组件的列表，请选择 **Show Code Component Changes**。
7. 单击 **Create**。这将通过收集属于应用程序的组件的所有文件，创建此应用程序的新版本。

   这将启动作业创建该版本。左下方的进度条显示了作业的状态和进度。如果作业失败，请转到 Jobs 屏幕找出原因。
8. 选择 **Close**。现在，即可部署应用程序的新版本。

部署特定版本

您可以将应用程序的版本部署到一个或多个目标。应用程序部署可将组件定义 (来自应用程序定义)转换成将在选定目标中指定的服务器上执行的部署步骤。部署步骤将执行以下操作:

将应用程序文件复制到目标服务器。

执行应用程序附带的脚本。

应用 SA 库中的指定项。

执行应用程序引用的 OO 流。

要部署版本，请执行以下操作:

1. 选择左下角的 **Deployment** 按钮。
2. 从 Select Version 下拉列表中，导航到并选择要部署的应用程序版本。选择现有版本，或创建新版本 (请参见创建新版本 (第 511 页))。

   将光标悬停在列表中的现有版本上方时，其层将显示在列表的右侧。如果发行版标有星号 (*)，这意味着应用程序的此发行版自上一版本创建以来经过修改。
3. 从 Select Target 下拉列表中，选择要部署此应用程序版本的目标。

   如果在该列表中看不到您的目标，请确保满足以下条件:
   - 该目标的层匹配应用程序层或是其超集。
   - 您具有此环境的 Deploy 权限。
   - 每层至少包含一台服务器。
   - 在生命周期中，该环境没有标记为 Not Ready。
   - 包含您版本的发行版的生命周期必须包括您的环境和目标。

   对于其他目标选项，请单击 Select Target 下拉列表右侧的 **Advanced Target Selector**:
   - 如果要将此版本部署到多个目标，您可以指定多个目标。只能选择满足上述所有要求的目标。
   - 如果 SA 核心与应用程序部署配置为支持这些目标，则您可以指定一个实时目标为此部署创建新目标。仅当满足以下条件时，此选项才可用:
     - SA 核心配置为运行 HPE Operations Orchestration (HPE OO) 流。
     - 在 Administration 屏幕上的 Application Settings 区域中指定了实时目标流。

   有关详细信息，请参见实时目标 (第 496 页)。
4. 可选：在右侧面板中查看部署作业设置。您可以修改其中一些设置 (请参见Deployment 屏幕 (第 505 页))。要展开某一部分，请单击  图标。
5. 单击 **Start Job**。这会启动部署作业，该作业会将应用程序组件复制到目标服务器并运行任何必要的代码、脚本和流。

可以在 Jobs 屏幕上查看部署的状态。

自定义部署参数

参数是用来传递信息的占位符，可能随部署应用程序的环境或目标而变化。参数的应用程序默认值可能因版本而异。例如，您可能在不同的版本中使用不同的数据库用户。

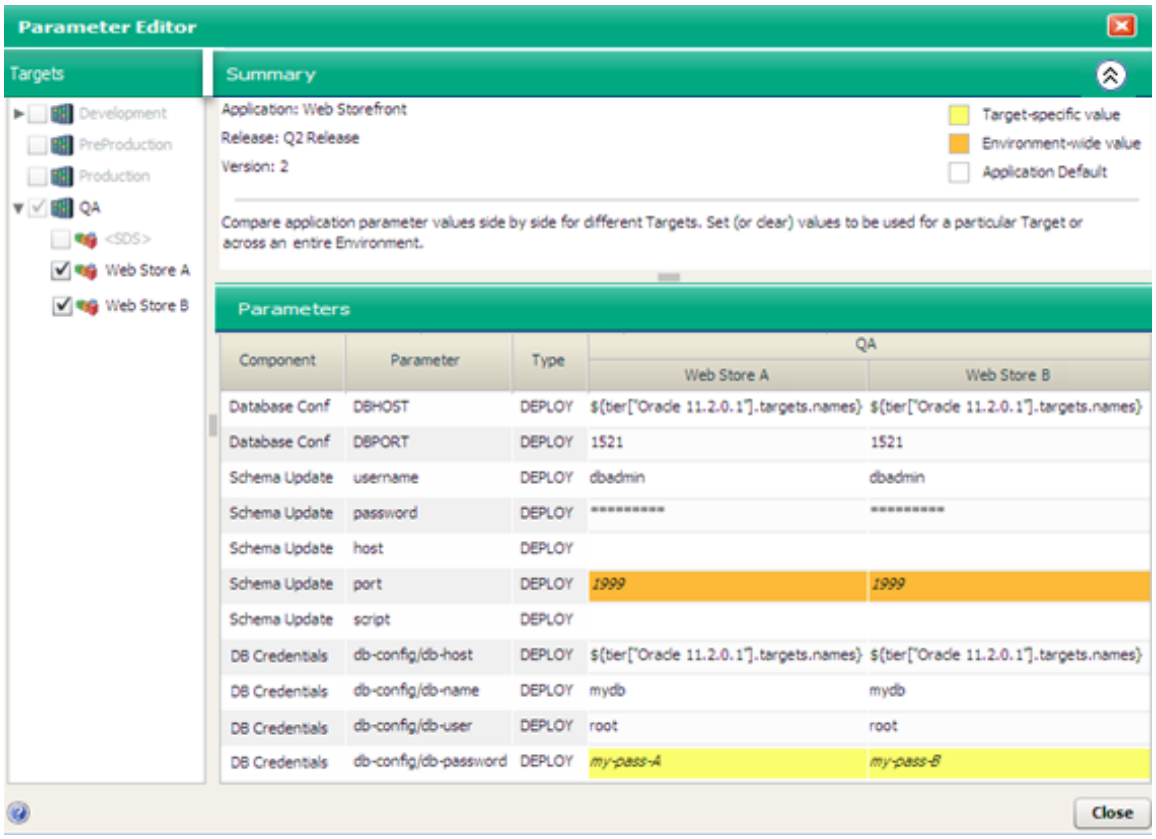
参数最初是在指定应用程序功能的组件 (请参见组件 (第 442 页)) 中定义的。可以通过三种方式设置其值:

- 在应用程序本身中 (当指定组件时)。
- 在将部署应用程序的环境中 (由部署应用程序的用户)。
- 在部署时针对特定目标。应用程序部署会记住目标特定的值，这样您无需每次部署时设置这些值。

参数与应用程序内的组件特定相关。只要参数的名称不变，且该组件在部署时从原始组件克隆，系统便会对每个新版本重用该值。

您可以使用参数编辑器在部署时确定应用程序的参数值。如果要部署到多个目标，则可为每个目标自定义这些值。以下示例包含 QA 环境中的两个目标:Web Store A 和 Web Store B。

参数编辑器示例



此处，这两个目标在左侧的 Targets 列表中已选定，因此右侧的 Parameters 表中显示了这两个目标的参数值。只有为此部署选择的目标才在 Targets 列表中可用；相关环境中的其他目标将灰显。

Type 列指示每个组件参数值的源 (请参见参数和特殊变量 (第 474 页)):

组件参数类型

类型	参数值源
DEPLOY	常量或特殊变量
VERSION	发行版参数
GLOBAL	全局参数 如果组件参数已加密，则参数编辑器不会显示由该组件参数引用的全局参数。

此部署的大部分参数都在指定 Web Storefront 应用程序的此发行版时确定了默认值。

两个参数 (Database Conf 组件中的 `DBHOST` 和 DB Credentials 组件中的 `db-config/db-host`) 使用“特殊变量”，并将在部署此版本时获取其值。在本例中，其值是各自目标中 Oracle 11.2.0.1 层中的服务器名称。

Schema Update 组件中的 port 参数从 QA 环境获取其值，在该环境中，此参数由部署应用程序的用户配置。

对于每个目标，DB Credentials 组件中的 db-config/db-password 参数值各不相同。在此示例中，该值是常量，但也可以是特殊变量。

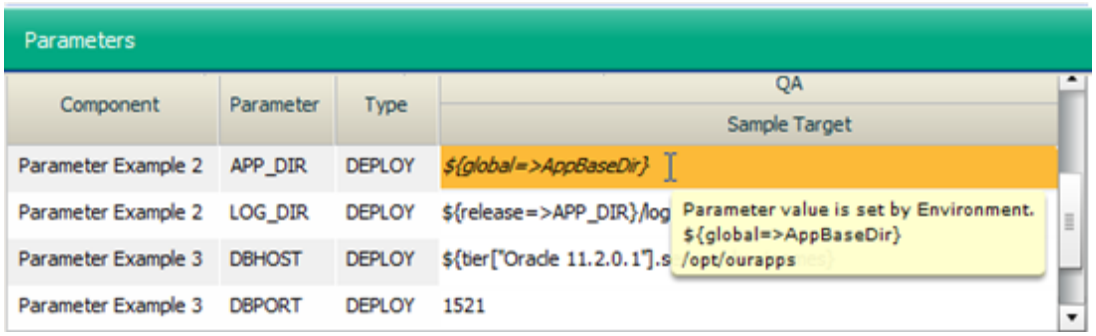
参数的优先顺序如下所示:

1. 目标特定的值 (表中的黄色背景)
2. 环境通用值 (橙色背景)
3. 应用程序默认值 (白色背景)

这意味着环境通用值将覆盖应用程序默认值。目标特定的值将覆盖环境通用值或应用程序默认值。

将鼠标悬停在参数值上方时，该参数的解析值将出现在工具提示中，此外，工具提示还指示任何覆盖是否生效。

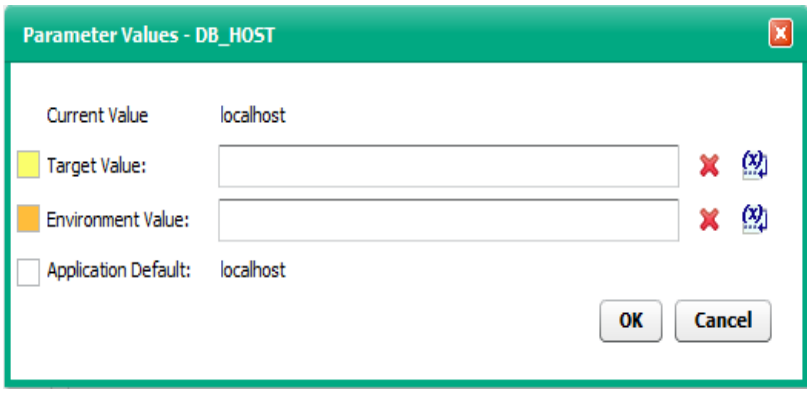
参数值工具提示示例



在下图中，光标悬停在 APP_DIR 参数的值上方。此参数引用 AppBaseDir 全局参数，后者解析为 /opt/ourapps。如工具提示中所示，这是环境覆盖，表中此单元格的橙色背景也表明了这一点。

要修改特定目标的参数值，请执行以下操作:

1. 在 Deployment 屏幕中，指定版本和目标。
2. 在右侧面板中，单击 图标展开 Parameters 部分。
3. 单击 **Edit** 或双击表中的任何行。将打开参数编辑器。
4. 在 Parameters 表中，双击参数的值。将打开 Parameter Values 对话框，如下所示:



请注意，加密值将显示为一串星号 (*)。

要指定常数值，请在 Target Value 中键入该值。

要引用特殊变量、发行版参数或全局参数 (请参见参数和特殊变量 (第 474 页))，请执行以下步骤：

a. 单击 图标。

b. 从列表中选择要使用的项目。

c. 单击 **OK**。

要擦除 Target Value 框中的任何字符并将该值重置为优先顺序中的下一个值，请单击“Delete”按钮: . 例如，如果删除目标值且存在环境值，则将使用环境值。如果没有环境值，则使用应用程序默认值。

5. 单击 **OK**，关闭 Parameter Values 对话框。

如果仅更改了此目标的参数值，则它现在将在 Parameters 表中具有黄色背景。如果将该值应用于环境中的所有目标，则它将具有橙色背景。

为给定发行版参数创建的环境或目标值覆盖还将用于从原始发行版克隆的所有发行版。

查看版本的属性

创建版本后，您可以使用版本查看器检查其属性。可以从应用程序部署用户界面中的多个位置打开版本查看器。

版本查看器启动点

位置	如何打开
Deployment 屏幕	单击版本选择器右侧的 **View version**。

版本查看器启动点(续)

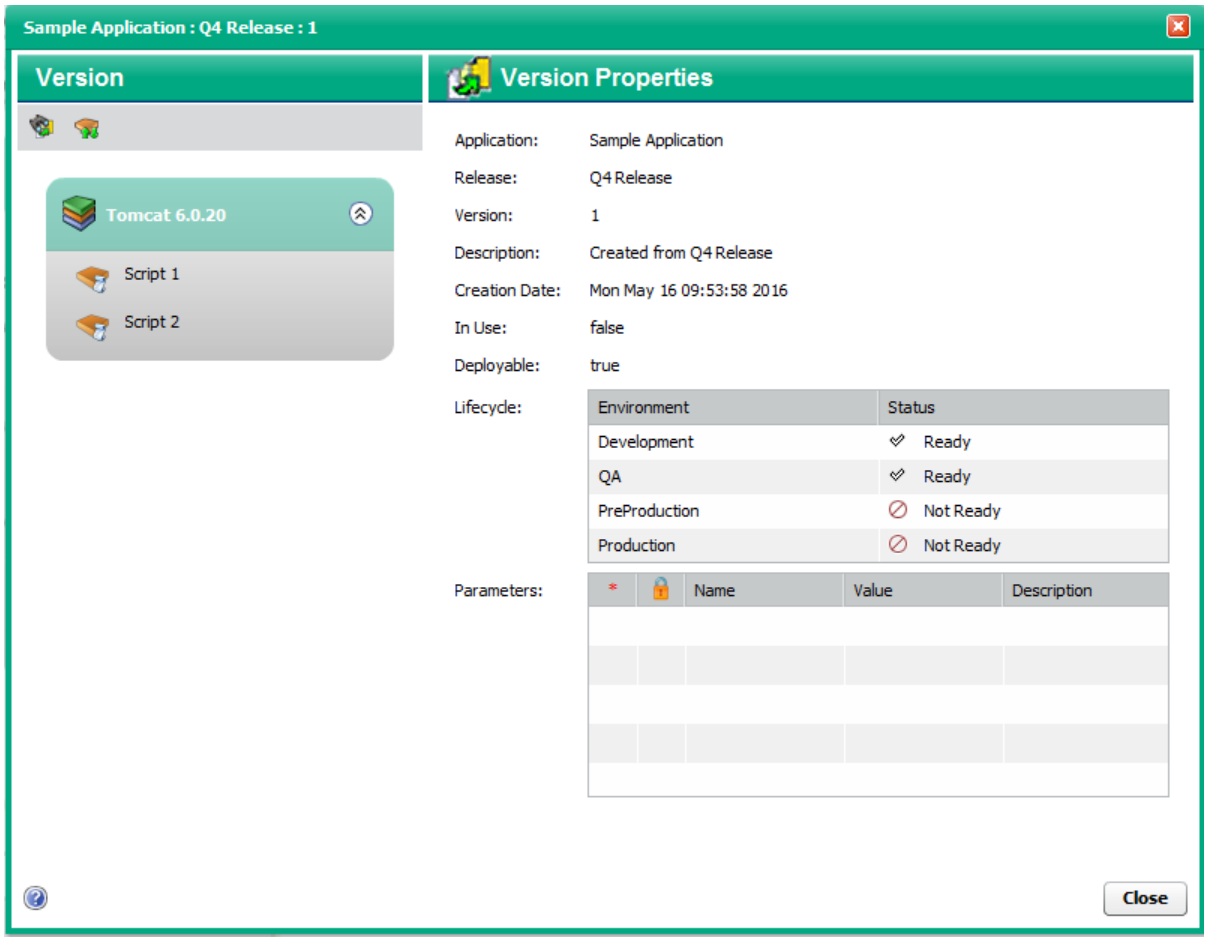
位置	如何打开
Application 屏幕	打开 Manage Applications 工具，在表中选择一个版本，然后单击工具栏上的 **View version**。
Jobs 屏幕	在 Job Logs 表中选择一个部署作业，然后单击工具栏上的 **View version**。 您也可以通过单击 Job details 窗口中的 Application 链接打开版本查看器。

版本查看器显示了在当前上下文中选择的版本。它在左侧面板中显示层和组件信息，而在右侧面板中显示版本、层或组件属性。在版本查看器中显示的信息为只读。

创建版本时，发布信息是不变的，但层信息可以更改。层配置指定要用于该层的中间件。由于中间件能够更新，因此旧应用程序可以使用新的中间件。在版本查看器中显示的层信息表示当前层配置。

要查看版本的属性，您必须具有关联应用程序的“查看”权限 (请参见应用程序权限 (第 557 页))。

版本查看器 - 结构视图

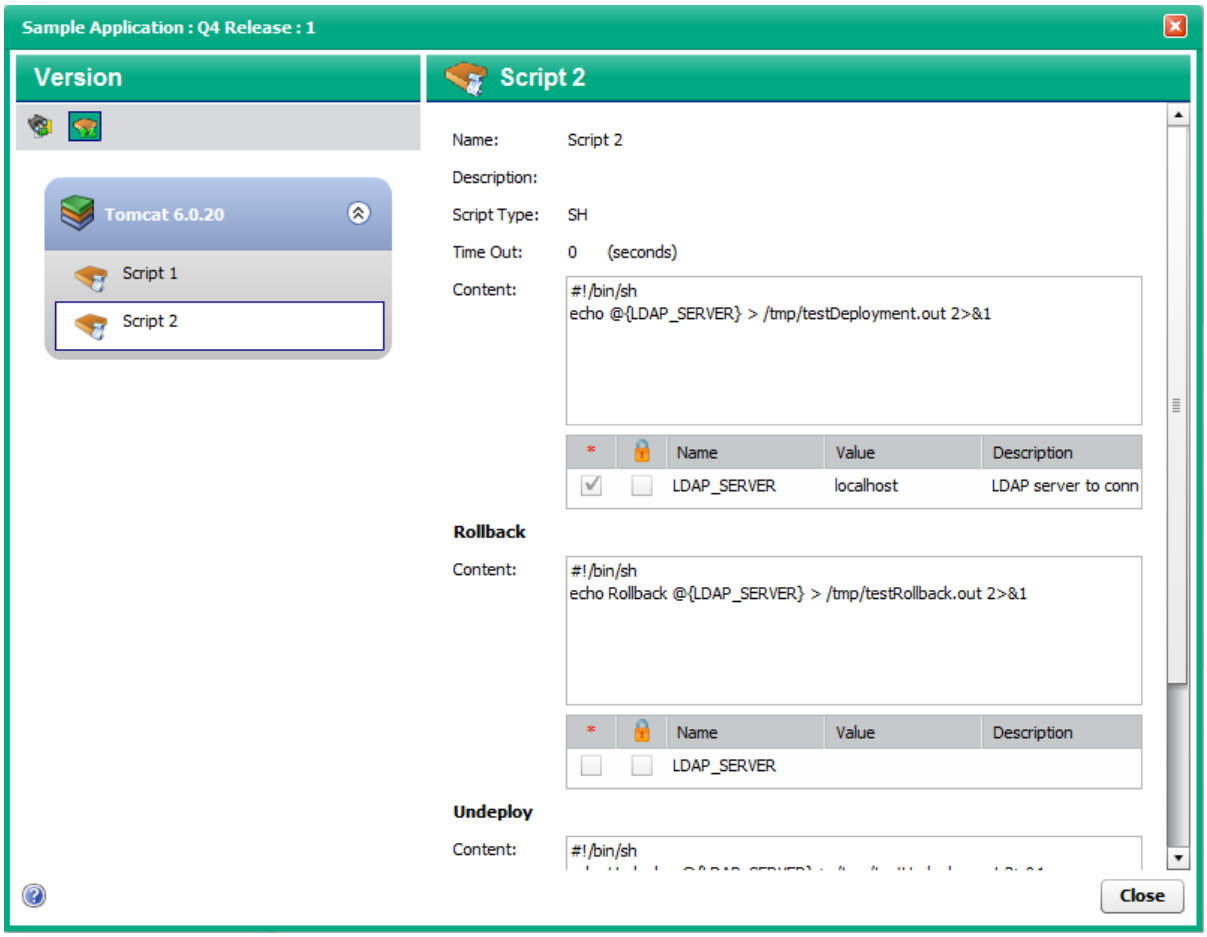


可通过以下两种方式之一查看左侧面板中的信息:

- **结构**视图显示应用程序中包含的层以及每层中包含的组件。
- **顺序**视图显示将按其部署组件的顺序。

在任一视图中，选择左侧面板中的一个层或组件，可在右侧面板中显示其属性。

版本查看器 - 顺序视图



您可以使用工具栏按钮进一步控制显示的信息。

版本查看器工具栏按钮

按钮	用途
	在左侧面板中，按过去或将来的部署顺序列出组件。
	在左侧面板中，将组件顺序 (顺序) 视图更改为层 (结构) 视图。
	在右侧面板中显示版本属性。

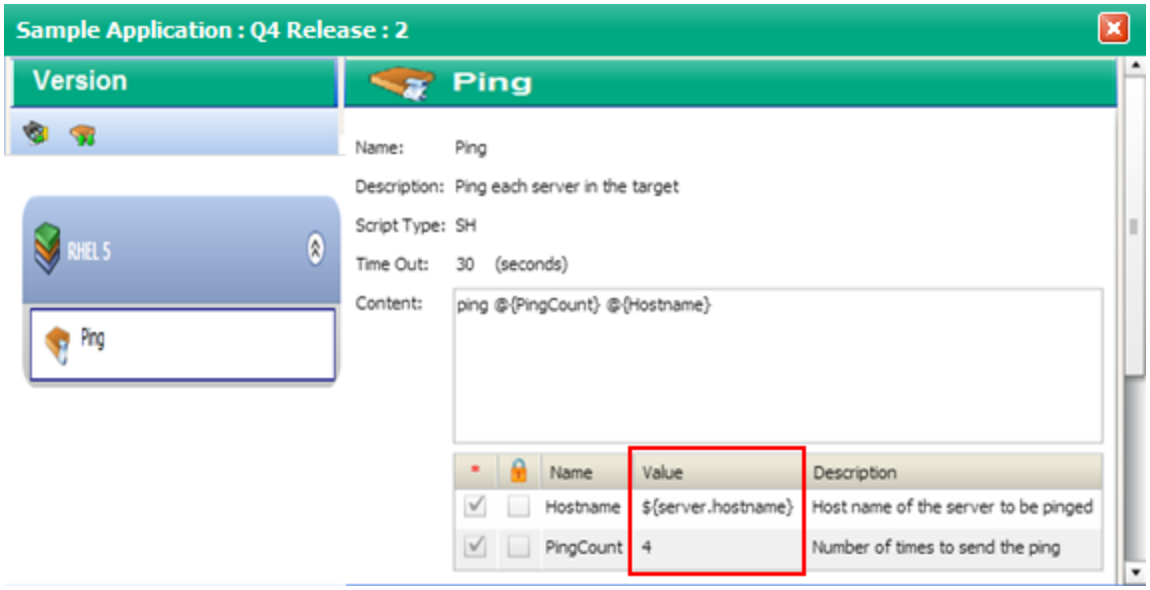
请注意以下几点：

- 在 Application 屏幕上查看应用程序时，您会看到备份、回滚和取消部署脚本的名称 (如果这些脚本已启用)。但在版本查看器中，您会看到在创建版本时启用的每个脚本的内容。将这些脚本的内容复制到版本，这样后续的脚本更改就不会影响现有版本。

- 版本查看器中显示的参数值是该组件的默认值。如果已部署此版本，版本查看器中显示的值可能与该部署作业的 Job detail 或 Step detail 窗口中显示的部署特定值不匹配 (请参见 Jobs 屏幕 (第 525 页))。

在下面的简化示例中，应用程序包括一个名为 Ping 的脚本组件，仅用于对目标中的每台服务器执行 ping 操作。在版本查看器中，您将看到参数的默认值:

版本查看器参数列表

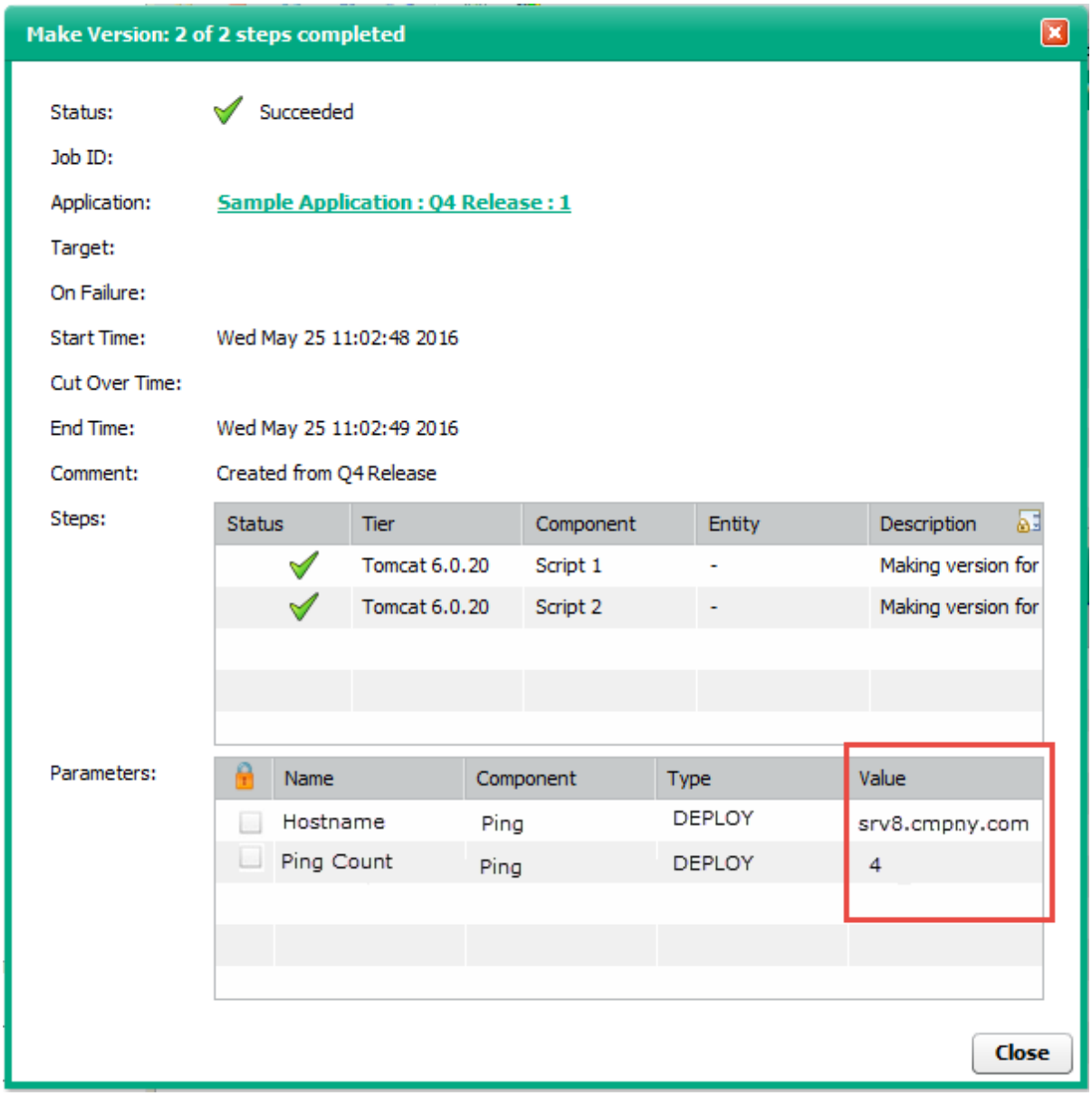


但是，将此应用程序的版本部署到服务器时，Hostname 参数包含该服务器的实际主机名。

您可以在此部署作业的 Job details 窗口中看到这种替换。在此示例中，在部署时为特殊变量 `${server.hostname}` 分配了值 `srv8.compny.com`。

请注意，PingCount 参数是一个常量 (而非特殊变量)。在本例中，目标和环境在部署时都不覆盖默认值。因此，该参数在两个视图中具有相同的值 (4)。

Job details 窗口参数列表



回滚部署

如果在版本部署期间发生故障，则该部署将自动从故障点回滚。从 Jobs 屏幕中，您可以手动回滚已部署的任何版本。

无论是自动还是手动启动回滚操作，过程和结果都是相同的。

回滚操作按照与部署相反的顺序删除组件，然后将相关文件、目录或策略还原到其先前状态。回滚指令在定义组件时指定。如果对组件禁用回滚，则不会回滚该组件。

回滚行为 (按组件类型)

组件类型	回滚行为
代码	执行特定于 OS 的回滚并还原由应用程序部署管理员维护的脚本 (请参见管理应用程序部署 (第 560 页))。
脚本	执行回滚脚本 (如果在组件中指定)。
配置文件	执行由应用程序部署管理员维护的特定于 OS 的还原脚本。
应用程序配置	从服务器中删除应用程序配置并还原原始文件 (如果已保存)。
软件策略	从服务器分离策略、还原相关的现有策略 (如有) 并进行修正。
程序包	如果部署方法是 Policy，则分离和修正包含程序包的策略，并附加和修正相关策略 (如有)。 如果部署方法是 Package，则删除程序包并安装任何相关的程序包 (如有)。
OO 流	启动指定的 OO 回滚流 (如果已指定)。
DMA 流	执行指定的 DMA 流 (如果已指定)。
Windows 注册表	执行由应用程序部署管理员维护的特定于 Windows 的回滚脚本。

回滚操作会将服务器恢复到在该部署之前所处的状态。

标准回滚脚本仅为每个发行版维护一个备份版本，为了节省目标服务器上的磁盘空间，它们将删除任何现有的备份文件。这意味着，如果您回滚的版本低于最近部署的版本，应用程序部署将无法还原代码文件。如果您尝试回滚以前的版本，应用程序部署将显示一条警告消息。然后，您可以确认或取消回滚。

要手动回滚部署，请执行以下操作:

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 在 Jobs Log 表中，选择与要回滚的部署对应的行。
3. 单击 **Rollback**。
   将打开 Rollback Job 对话框。
4. 单击“是”。将创建和启动回滚作业。

由于执行的唯一组件没有启用回滚，因此可能无需回滚。如果是这种情况，您将看到一条警告消息。

取消部署某项部署

部署版本时，您可以请求取消部署以前版本 (如果确实部署了以前版本)。从 Jobs 屏幕中，您可以手动取消部署已部署的最新版本。应用程序部署管理员可以取消部署任何版本。

无论是自动还是手动启动取消部署操作，过程和结果都是相同的。

取消部署操作按照与部署相反的顺序删除组件。取消部署指令在定义组件时指定。如果对组件禁用取消部署，则不会删除该组件。

取消部署行为 (按组件类型)

组件类型	取消部署行为
代码	从默认安装位置删除文件。
脚本	执行取消部署脚本 (如果在组件中指定)。
配置文件	执行由应用程序部署管理员维护的特定于 OS 的取消部署脚本。
应用程序配置	删除应用程序配置。
软件策略	从服务器分离策略并进行修正。
程序包	如果部署方法是 Policy，则分离和修正包含程序包的策略。 如果部署方法是 Package，则删除程序包。
OO 流	启动指定的 OO 取消部署流 (如果已指定)。
DMA 流	执行指定的 DMA 流 (如果已指定)。
Windows 注册表	执行由应用程序部署管理员维护的特定于 Windows 的取消部署脚本。

取消部署操作将删除版本的任何痕迹。它不会尝试将服务器恢复到特定状态。

要手动取消部署版本，请执行以下操作:

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 在 Jobs Log 表中，选择与要取消部署的部署对应的行。
3. 单击 **Undeploy**。将打开 Undeploy Job 对话框。
4. 单击 **Yes**。将创建和启动取消部署作业。

管理应用程序部署作业

要查看简单示例，请参见快速入门 (第 414 页)。

概述

应用程序部署将创建并运行以下类型的作业:

- 每次创建新版本时，都会创建 Make Version 作业。
- 每次部署版本时，都会创建 Deployment 作业 (每个目标一个作业)。
- 每次因部署失败而自动或手动回滚部署时，都会创建 Rollback 作业 (每个目标一个作业)。
- 每次在部署更新版本之前自动或手动取消部署版本时，都会创建 Undeployment 作业 (每个目标一个作业)。

作业由一系列步骤组成。部署、回滚或取消部署每个组件需要执行多个步骤。步骤数取决于使用的组件类型、使用的组件数量和服务器的数量。

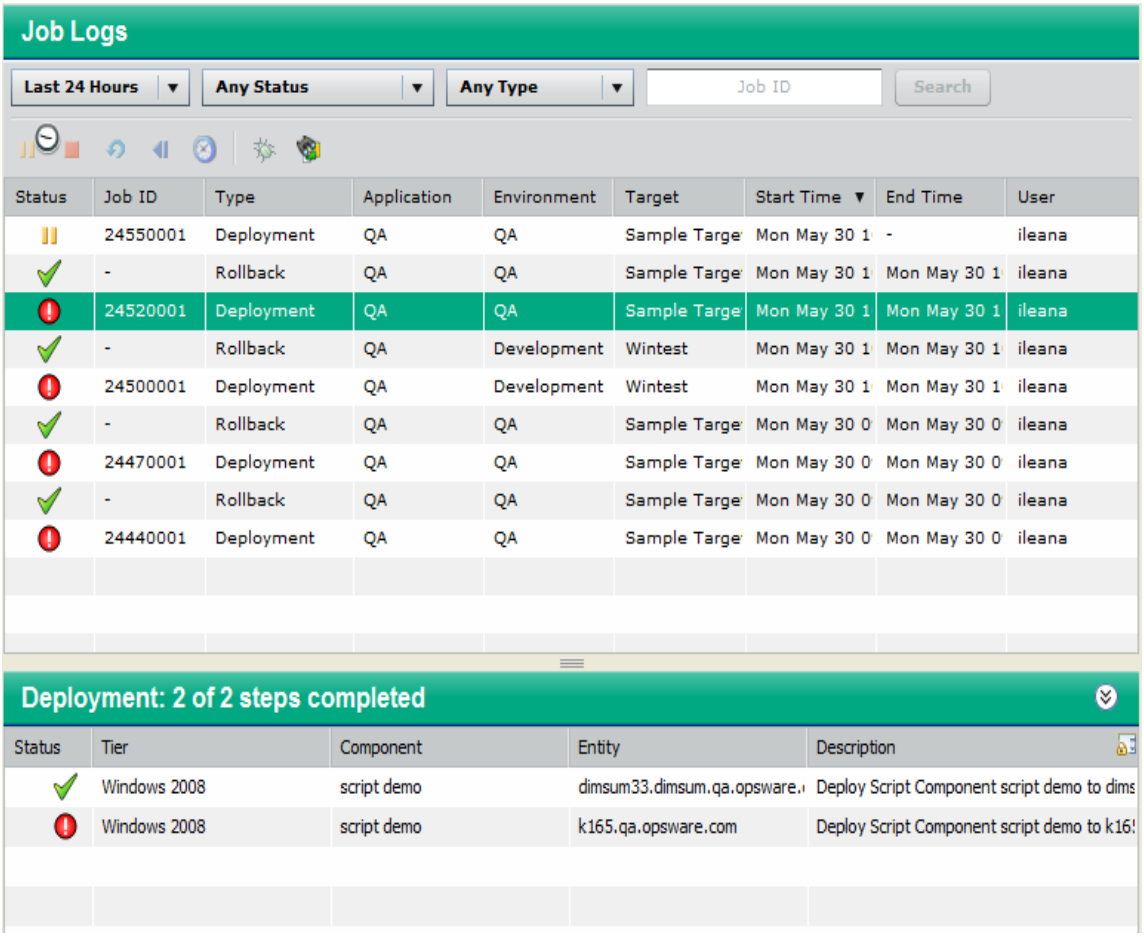
Jobs 屏幕

本节将讨论以下主题:

Jobs 屏幕

通过 Jobs 屏幕可以查看应用程序部署作业的累积日志。下图显示了 Jobs 屏幕的右侧面板。

Jobs 屏幕



Job Logs 部分包含与您指定的筛选条件匹配的所有应用程序部署作业列表。您可以根据特定的时间范围、作业状态、作业类型或特定的作业 ID 进行筛选。

如果在启动部署作业时选择了“Show job status when deployment starts”，则 Job Logs 将自动筛选为仅显示您的作业。以下框将显示在 Job Logs 区域的顶部：

您可以通过选择位于顶部的“Clear Search”按钮 清除此筛选器。

在 Job Logs 部分中选择一行时，该作业的步骤将显示在下半部分中。在此处的示例中，请注意以下几点。

- 所有步骤均无复选标记 指示它们已成功完成。
- 步骤 1 和 2 失败：

- 步骤 3 和 8-13 仍在计划: 
- 步骤 4 到 7 将跳过: 

如果部署作业中的单个步骤失败，则整个作业将自动回滚，并且目标服务器将恢复到在部署之前所处的状态。

系统会始终创建自动回滚作业。如果无需回滚，则该作业没有任何步骤，且其描述将说明原因。

作业状态指示器

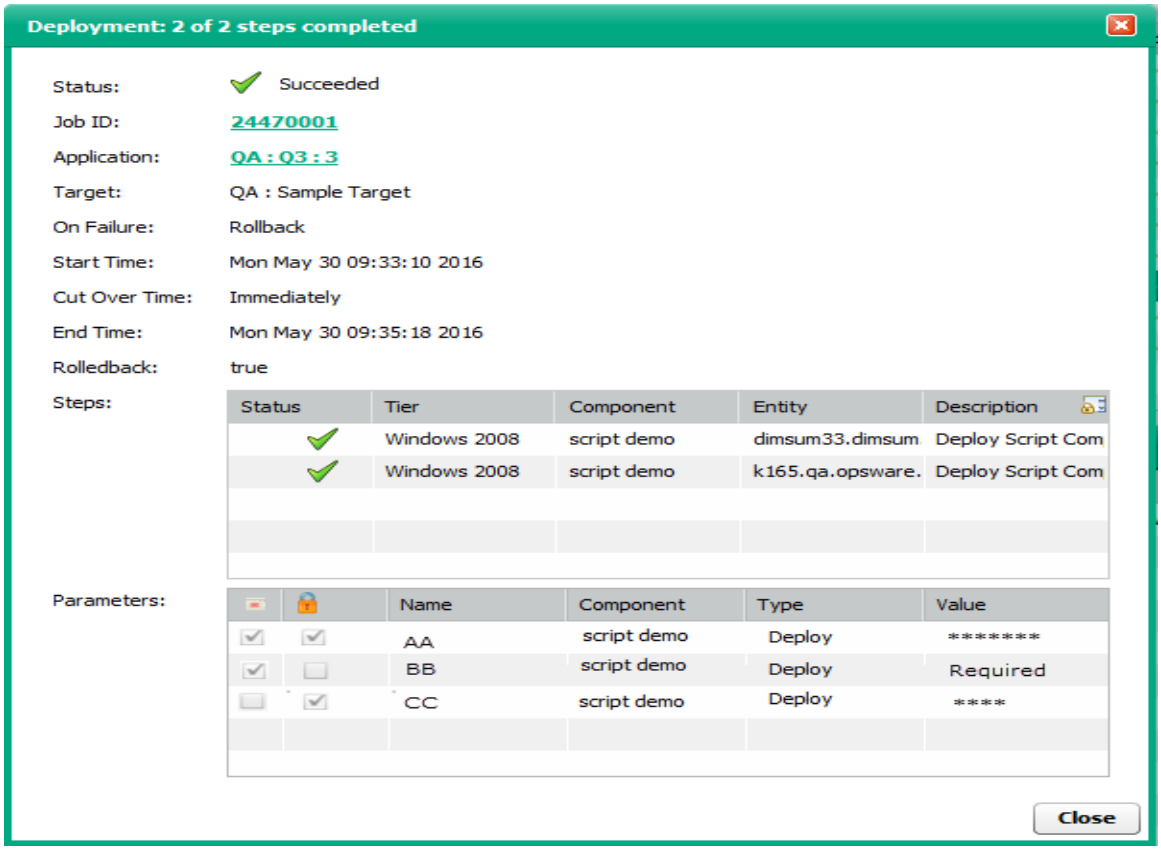
图标	状态	含义
	已成功	步骤或作业已成功完成。
	已失败	步骤或作业失败。
	已计划	计划运行步骤或作业。
	跳过	将跳过此步骤，因为与之相关的组件或服务器在调试器中未选定。请参见 调试部署作业 (第 534 页)。
	已跳过	此步骤已跳过，因为与之相关的组件或服务器在调试器中未选定。请参见 调试部署作业 (第 534 页)。
	运行中	步骤或作业当前正在运行。对于涉及许多步骤的作业，在执行这些步骤时，Status 列中将显示一个绿色进度条。
	注意	作业失败，为便于调试，现在处于已挂起状态。请参见 调试部署作业 (第 534 页)。
	已暂停	作业已暂停。
Paused (pending)	已按下“Pause”按钮，但是作业尚未暂停。必须先完成当前正在执行的步骤，然后才能暂停作业。	
	已取消	作业已取消。
Cancelled (pending)	已按下“Cancel”按钮，但是作业尚未取消。必须先完成当前正在执行的步骤，然后才能取消作业。	

当作业处于 In Progress 状态时，如果应用程序部署 (SA 的 "da" 组件) 停止并重新启动，则会仔细检查每个作业步骤的状态以确定作业状态。如果所有作业步骤已成功完成，则作业状态将设置为 Succeeded。如果任何作业步骤处于 In Progress 状态，则该步骤将标记为 Failed，一个错误将被记录到该步骤的输出，并且作业也会标记为 Failed。

默认情况下，将突出显示在 Job Logs 表中选择的作业的第一个 In Progress 步骤。完成该步骤并启动新步骤后，突出显示将移至新步骤。如果要阻止移动突出显示，则可使用“Scroll Lock”按钮。

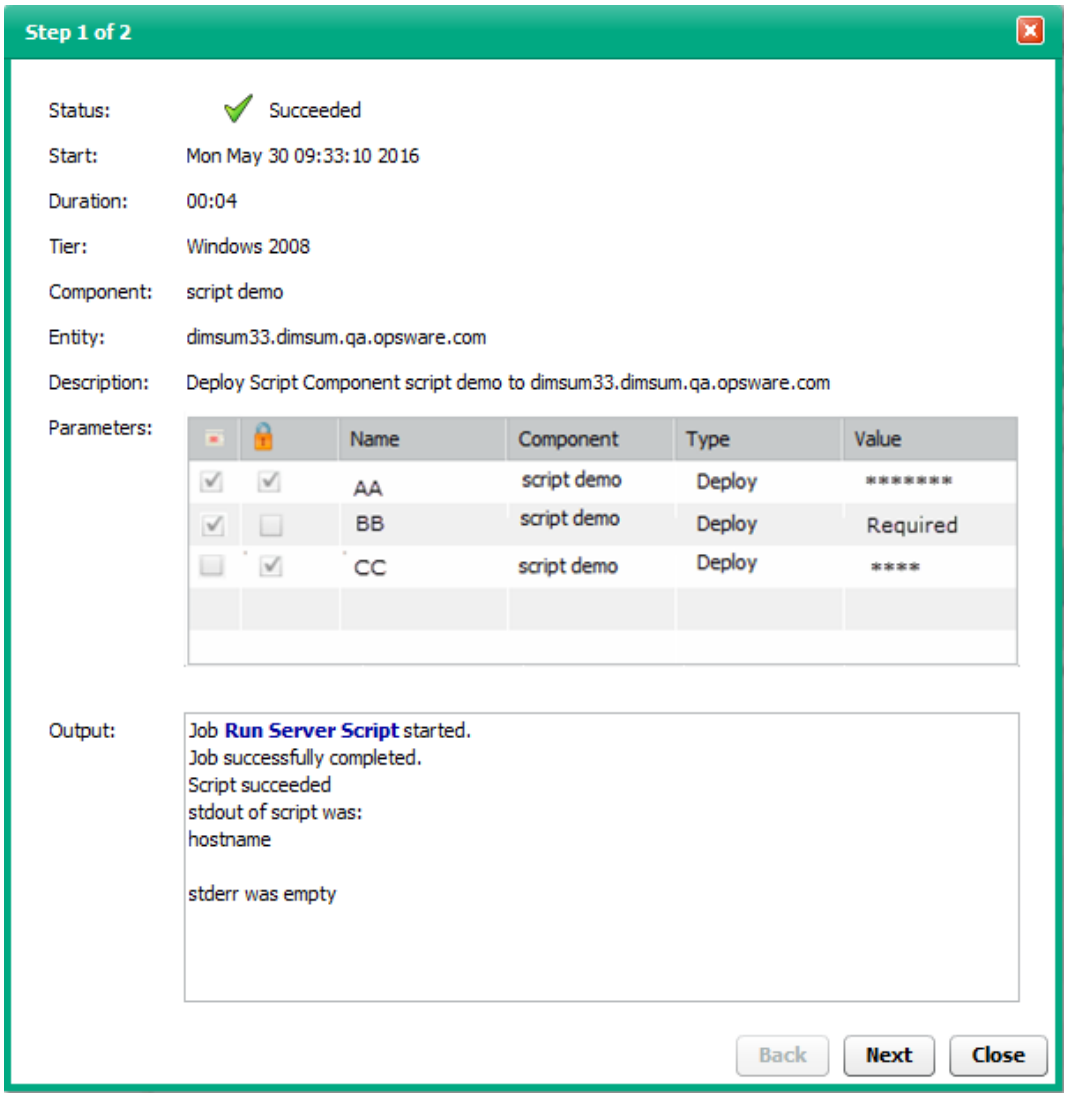
要查看有关作业的详细信息，请在 Job Logs 表中双击与该作业对应的行。将打开一个新窗口，如下所示：

Job Detail 窗口



如果双击 Steps 表中的一行，您可以查看有关如何执行该步骤的更多详细信息。在以下示例中，显示了步骤 4 的详细信息。

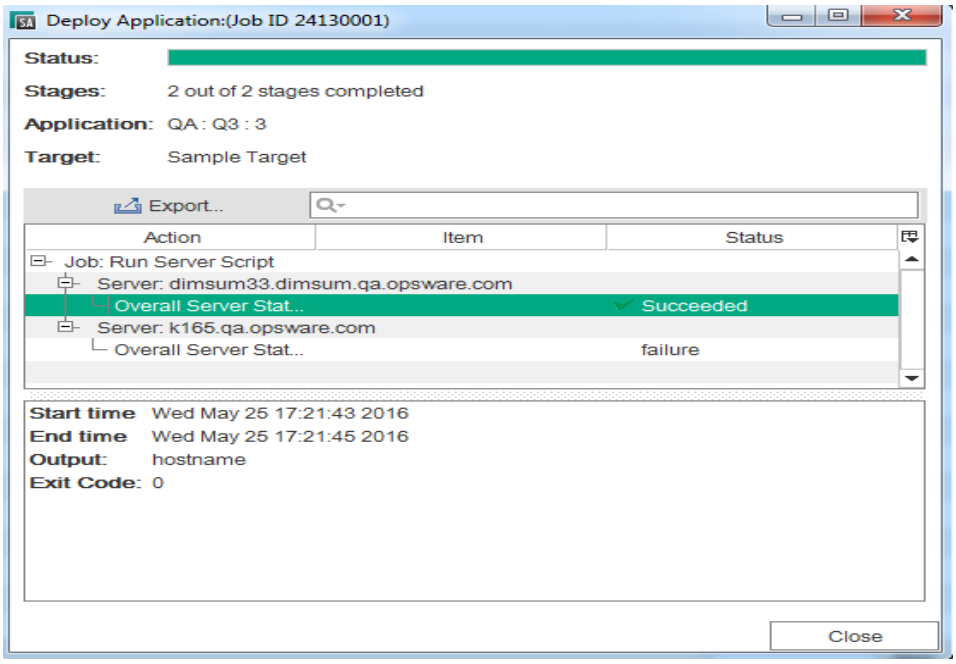
Step Detail 窗口



在这个简单的示例中，名为 "script demo" 的脚本组件将其所有三个参数的值回显到 stdout。如 Output 框中所示，四个参数的值被回显到 stdout，且脚本已成功完成。

在 Step detail 窗口中，通过显示在 Output 框中的任何蓝色粗体文本，可以链接到 SA“Deploy Application”作业窗口中提供的更多详细信息。在以下示例中，单击了 Run Server Script 链接。

Deploy Application Job 窗口



此窗口中的信息按服务器组织，并列出了目标中的每台服务器。每个“Overall Server Status”行对应于在原始应用程序部署作业中对特定服务器执行的单个步骤。单击其中一行，可查看有关对特定服务器执行该步骤的详细信息。

您也可以通过单击应用程序部署 Job Detail 窗口中的 Job ID 打开此窗口。

已阻止作业

如果为特定类型的作业选择了 Require Approval (在 SA 客户端 Administration 选项卡上的 Job Blocking 页面中)，应用程序部署将显示一条消息，指出该作业已阻止，待批准。

必须在 SA 客户端中的 Jobs and Sessions 选项卡上批准或取消待批准的已阻止作业。它不能从 应用程序部署 Jobs 屏幕取消。

如果在 SA 客户端中取消已阻止作业，应用程序部署 Jobs 屏幕将指示此作业已失败。

有关详细信息，请参见开发人员指南概述 (第 23 页)中的“作业批准集成”一节。

使用作业

您可以使用 Jobs 窗口执行下列任一操作:

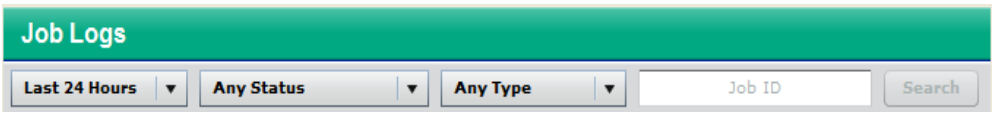
还可以使用应用程序部署调试器解决作业。有关详细信息，请参见调试部署作业 (第 534 页)。

有关从 Jobs 屏幕向下钻取详细信息的信息，请参见 Jobs 屏幕 (第 525 页)。

查找特定作业

您可以使用搜索工具筛选 Job Logs 面板中显示的作业或查找特定作业 (如果知道作业 ID)。

Jobs 屏幕的搜索工具



在此示例中，只会列出上周失败的部署作业。

要筛选作业日志，请执行以下操作:

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 执行以下一项或多项操作:
   - 指定时间范围
   - 指定作业状态
   - 指定作业类型
   - 指定作业 ID
3. 单击“Search”。

按作业 ID 进行搜索时，您必须指定整个作业 ID (而非部分 ID 或通配符)。您可以使用逗号分隔指定多个作业 ID。

要扩大搜索范围并显示更多作业，请在一个或多个下拉列表中选择“Any”选项。

暂停作业

您可以暂停已部署、且当前正在进行的任何作业。应用程序部署管理员可以暂停当前正在进行的任何作业。

暂停作业时，应用程序部署将完成当前正在处理的步骤，然后暂停。直到当前步骤完成，作业的状态才会变为“Paused (pending)”。当前步骤完成后，除非取消或得到指示要恢复，否则作业的状态将是“Paused”。

要暂停作业，请执行以下操作:

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 在 Job Logs 中查找并选择要暂停的作业 (请参见查找特定作业 (第 531 页))。
3. 在工具栏上，单击 **Pause**。
4. 单击 **Yes** 确认。

除非您 (或应用程序部署管理员) 恢复作业，否则作业将保持暂停状态 (请参见恢复暂停的作业 (第 532 页))。

恢复暂停的作业

您可以恢复暂停的任何作业。应用程序部署管理员可以恢复暂停的任何作业。

恢复作业后，应用程序部署将会运行作业中下一计划步骤。

要恢复暂停的作业，请执行以下操作:

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 在 Job Logs 中查找并选择要恢复的暂停作业 (请参见查找特定作业 (第 531 页))。
3. 在工具栏上，单击 **Resume**。

作业将从以前的暂停处继续运行 (请参见暂停作业 (第 532 页))。

取消作业

您可以取消任何当前正在进行或已暂停的已部署作业。应用程序部署管理员可以取消正在进行或已暂停的任何作业。

取消作业时，应用程序部署将在停止前完成正在进行的步骤。完成当前步骤前，作业的状态为“Cancelled (pending)”。完成当前步骤后，作业的状态将转为“Cancelled”。其余所有步骤均处于“Scheduled”状态。

如果要取消的作业是部署作业，应用程序部署将尝试回滚部署。

要取消作业，请执行以下操作：

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 在 Job Logs 中查找并选择要取消的作业 (请参见查找特定作业 (第 531 页))。
3. 在工具栏上，单击 **Cancel**。
4. 单击 **Yes** 确认。

不能恢复已取消的作业。如果日后要将此版本部署到同一目标，只需重新部署相同版本即可。

重新计划作业

您可以重新计划部署的“Scheduled”作业。应用程序部署管理员可以重新计划任何“已计划”的作业。

要重新计划作业，请执行以下操作：

1. 转到 Jobs 屏幕 (单击左下角的 Jobs)。
2. 在 Job Logs 中查找并选择要重新计划的作业 (请参见查找特定作业 (第 531 页))。它必须处于“Scheduled”状态。
3. 在工具栏上，单击 **Reschedule**。
4. 选择一个新的分段时间和/或切换时间。
5. 单击 **OK**。

请注意以下几点：

如果作业已计划但尚未运行，并启用了“Auto-undeploy”，在重新计划作业时，将显示信息说明您需要重新计划取消部署和部署作业。

如果已暂存但尚未切换，则您只能重新计划切换分段。

在运行作业时，**Reschedule** 按钮处于禁用状态。

回滚部署

完成已部署的部署作业后，您可以手动进行回滚。这会将服务器恢复到在该部署之前所处的状态。应用程序部署管理员可以回滚任何部署作业。

有关说明，请参见回滚部署 (第 522 页)。

如果作业失败或取消并启用了回滚，则应用程序部署将自动进行回滚。

标准回滚脚本仅为每个发行版维护一个备份版本，为了节省目标服务器上的磁盘空间，它们将删除任何现有的备份文件。这意味着，如果您回滚的版本低于最近部署的版本，应用程序部署将无法还原代码文件。如果您尝试回滚以前版本，将显示一条警告消息，您可以确认或取消回滚。

取消部署某项部署

完成已部署的部署作业后，您可以手动取消部署。这将删除部署的任何痕迹，但不会尝试将服务器恢复到在该部署之前所处的状态。如果尚未部署干预版本，则取消部署与回滚等效。应用程序部署管理员可以取消部署任何部署作业。

取消部署是一种性能比回滚更高的“撤消”操作。

调试部署作业

您可以使用应用程序部署调试器解决部署作业、回滚作业或取消部署工作 (不能使用它调试 Make Version 作业)。

请考虑以下场景:

1. 部署作业因特定组件存在问题而失败。
2. 您进行更改以修复该组件。
3. 然后，创建一个可调试作业并启动调试器。
4. 在调试器中，尽量禁用该作业。这允许您快速测试所做的更改，看看它是否生效。
5. 如果所做的更改生效，则可启用更多作业，直到您确定整个作业生效为止。

根据故障排除目标，您可以从 Deployment 屏幕或 Jobs 屏幕启动调试器。您可以使用调试器执行以下操作:

- 使用断点
- 部署或跳过任何组件
- 部署到或跳过任何服务器
- 单步执行部署
- 启用或禁用层配置和备份步骤

跳过组件或服务器时，将跳过有关该组件或服务器的所有作业步骤，包括回滚和取消部署步骤。

要运行调试器，必须具有相关应用程序的“编辑”权限 (请参见设置权限 (第 546 页))或“管理应用程序部署”权限 (请参见设置权限 (第 546 页))。

Debugger 窗口

调试器在一个单独的窗口中运行，该窗口包含七个功能区域:

- 工具栏 (第 535 页)
- 状态栏 (第 536 页)
- Job Options (第 536 页)
- Job Selections (第 537 页)
- 详细信息 (第 538 页)
- Steps (第 538 页)
- Parameters (第 539 页)
- 输出 (第 539 页)

要展开折叠区域，请单击蓝色条上的“Expand”  按钮。

有关使用调试器执行常见故障排除任务的信息，请参见排除作业故障 (第 540 页)。

工具栏

调试器工具栏位于 Job Debugger 窗口的左上角，可控制正在进行故障排除的作业如何运行。

调试器工具栏按钮

按钮	名称	用途
	Single Step	单击此按钮可运行调试作业中的下一步。 不执行标记为“Skipped”的步骤。
	Resume	单击此按钮可继续运行已暂停的作业。 如果设置了断点，则作业将一直运行到该断点为止。
	Pause	单击此按钮可暂停调试作业。 在运行作业时，Pause 按钮将替换 Resume 按钮。
	Cancel Job	单击此按钮可取消整个调试作业。
	Set Breakpoint	在 Steps (第 538 页) 区域中选择“Scheduled”步骤，然后单击此按钮可在该步骤设置断点。作业将在执行具有断点的步骤前停止。 不能在已执行或将跳过的步骤上设置断点。
	Remove Breakpoint	在 Steps (第 538 页) 区域中选择一个具有断点的步骤，然后单击此按钮可删除该断点。
	Refresh	强制刷新整个 Job Debugger 窗口。

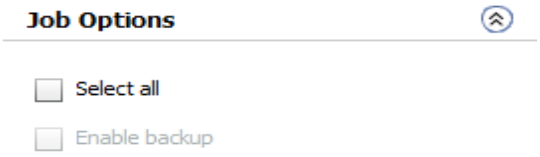
状态栏

状态栏位于 Job Debugger 窗口的左下角。它显示调试作业的当前状态:

有关所有可能的作业状态的描述，请参见作业状态指示器 (第 527 页)。

Job Options

Job Options 区域位于 Job Debugger 窗口左上角的工具栏正下方。在此处，您可以设置以下选项:



“Select all”选项用于选择此作业中包含的所有服务器和所有组件。如果清除“Select all”框，则将跳过所有作业步骤。有关其他信息，请参见选择或跳过特定组件或目标项 (第 541 页)。

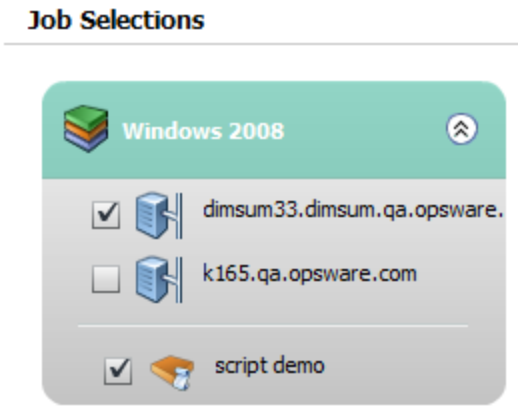
“Enable backup”选项用于运行此作业中包含的任何自动备份步骤。要禁用所有备份步骤，请清除此框。这将使调试作业运行得更快，但无法正常回滚。

如果清除“Select all”框，则会自动清除“Enable backup”框，因为清除“Select all”框将跳过所有作业步骤，其中包括备份步骤。有关详细信息，请参见跳过备份步骤 (第 544 页)。

这些选项仅在执行调试作业中的任何步骤前可用。

Job Selections

Job Selections 区域位于 Debugger 窗口的左侧：



在此处，您可以选择要包含在调试作业中的单台服务器和组件。清除对应于特定组件或服务器的框时，将跳过有关该组件或服务器的任何步骤，包括任何回滚和取消部署步骤。

在上述示例中，将跳过有关 k165.opsware.com 服务器的所有步骤。

您还可以启用或禁用部署与任何层关联的软件策略或 OS 序列。请参见《SA 10.50 管理指南》。

已执行或跳过其步骤的组件或服务器在 Job Selections 区域中处于禁用状态，无法选择。

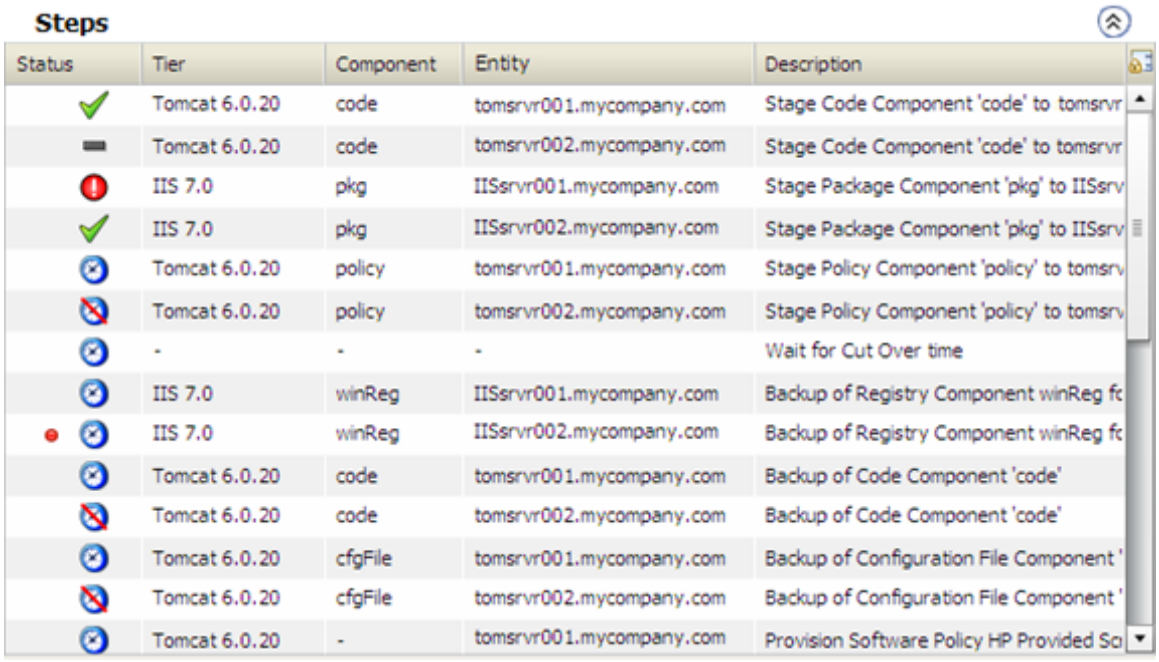
详细信息

Details 区域位于 Job Debugger 窗口的右上角。它包含有关正在进行故障排除的作业的信息。在作业运行之前，不会填充 Job ID、Target 和 Start Time 字段。如果从 Deployment 页面启动调试器，除非作业至少运行一个可执行步骤，否则这些字段将为空。

您可以在 Details 区域中添加或修改注释。在 Jobs 屏幕的 Job Logs 区域中双击此作业时，此注释将显示在 Details 窗口中。

Steps

Steps 区域位于 Details 区域下的 Job Debugger 窗口右侧：



Steps 区域显示应用程序部署在运行此作业时执行的各个步骤。在 Job Selections (第 537 页) 区域中清除服务器或组件对应的复选框时，将跳过有关该服务器或组件的所有步骤。

在上述示例中，将跳过有关 `tomsrvr002.mycompany.com` 服务器的所有步骤。

在执行步骤时，将更新状态图标。

调试器步骤状态指示器

图标	状态	描述
	已计划	计划运行此步骤。
	跳过	将跳过此步骤 (因为与之相关的组件或服务器在 Job Selections (第 537 页) 区域中未选定)。
	进行中	此步骤正在进行。
	已成功	此步骤成功。
	已失败	此步骤失败。
	已跳过	已跳过此步骤。

Parameters

首次打开特定作业的调试器时，如果尚未选择任何步骤，则 Parameters 表将列出该作业的所有参数。如果该作业没有参数，则将显示消息“Job has no parameters”。

选择特定步骤时，Parameters 表仅列出与在该步骤中显示的组件有关的参数:

如果该步骤没有参数，则将显示消息“Step has no parameters”。

Parameters 表为只读。不能覆盖调试器中的参数值。

输出

Output 区域显示应用程序部署运行在 Steps (第 538 页) 区域中选择的步骤时所执行的操作。它指示该步骤是否成功完成，并显示 `stdout` 和 `stderr` 的内容 (如果相关)。

通过在 Output 区域中以蓝色粗体显示的任何信息，可以链接到更多详细信息。

排除作业故障

本节中的主题说明如何使用调试器执行以下故障排除任务:

有关导航 Job Debugger 窗口的信息，请参见Debugger 窗口 (第 535 页)。

启动调试器

可使用两种方法启动调试器:

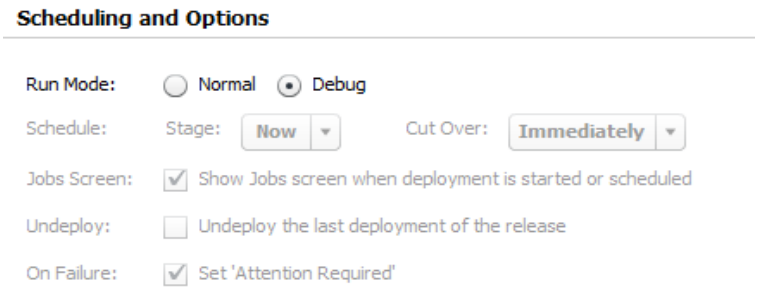
- 如果要从一开始解决部署作业，请从 Deployment 屏幕启动调试器。
- 如果要解决的作业已处于 In Progress 或 Paused 状态，请从 Jobs 屏幕启动调试器。

  从 Jobs 屏幕中，您可以启动部署作业、回滚作业和取消部署作业的调试器 (不能调试 Make Version 作业)。

您只能为处于 Paused、In Progress 或 Attention Required 状态的作业启动调试器。您不能为 Scheduled、Completed 或 Failed 作业启动调试器。

要从 Deployment 屏幕启动调试器，请执行以下操作:

1. 在 Scheduling and Options 下，选择 **Debug** 模式。



当您选择 **Debug** 模式时，将自动设置部署计划和选项。

有关这些选项的详细信息，请参见Scheduling and Options (第 510 页)。

2. 单击 **Start**。

要从 Jobs 屏幕启动调试器，请执行以下操作：

1. 在右窗格中，选择要调试的作业。
2. 单击工具栏上的 **Debug**。

选择或跳过特定组件或目标项

当调试作业处于 Paused 状态时，您可以使用 Job Selections 设置指定恢复作业后哪些组件将部署到哪些目标服务器。您可以将某层中的任何或所有组件部署到该层中的任何或所有目标项，或者不将任何组件部署到任何目标项。

您只能在启动涉及特定组件的任何步骤之前更改该组件的设置。同样，您只能在启动涉及特定服务器的任何步骤之前更改该目标项的设置。

如果通过选择 Deployment 页面上的“Debug”模式启动调试器 (请参见启动调试器 (第 540 页))，作业将在执行任何步骤之前自动处于 Paused 状态。在这种情况下，您可以完全控制包括或排除哪些组件和目标项。

如果从 Jobs 屏幕启动调试器，仅当满足以下某个条件时，您才可以访问 Job Selections 设置：

- 在单击 Jobs 屏幕上的 **Debug** 之前，作业处于 Paused 状态。
- 您已在调试器中单击 **Pause**。

在任一种情况下，仅当未启动涉及组件或目标项的任何步骤时，您才能更改该组件或目标项的设置。

以下说明展示如何实现一些常见的调试场景。所有这些示例假定您通过选择 Deployment 页面上的“Debug”模式启动调试器。

要将所有组件部署到所有层中的所有目标项，请执行以下操作：

在 Job Options 区域中，选中 **Select all** 框。

要跳过有关特定组件或目标项的所有步骤，请执行以下操作：

在 Job Selections 区域中，清除要跳过的组件或目标服务器对应的框。

涉及该组件或目标项的所有步骤将在 Steps 区域中显示“Skip”  状态。

要将单个组件部署到层中的所有目标项，请执行以下操作：

1. 在 Job Options 区域中，清除 **Select all** 框。
2. 在 Job Selections 区域中，选择要部署的组件。将自动选择层中的所有服务器。

要将所有相关组件部署到单台服务器，请执行以下操作：

1. 在 Job Options 区域中，清除 **Select all** 框。
2. 在 Job Selections 区域中，选择要部署组件的服务器。将自动选择层中的所有组件。

要将单个组件部署到单台服务器，请执行以下操作：

1. 在 Job Options 区域中，清除 **Select all** 框。
2. 在 Job Selections 中：
   a. 选择要部署的组件。将自动选择层中的所有服务器。
   b. 清除层中所有服务器对应的框，但要部署此组件的服务器除外。

暂停调试作业

如果调试作业处于 In Progress 状态，并且您要将 Job Selections (第 537 页) 设置更改为包括或跳过单个组件或服务器，则必须先暂停作业。

当调试作业处于 Paused 状态时，排除作业故障 (第 540 页) 设置已启用，并且您可以更改未执行或跳过其步骤的任何组件或服务器的设置。

要暂停处于 In Progress 状态的调试作业，请单击工具栏上行的“Pause”  按钮。

作业首先将过渡到 Pause-Pending 状态 (以便执行完当前正在进行的任何步骤)，然后将过渡到 Paused 状态。

恢复调试作业

要恢复处于 Paused 状态的调试工作，请单击工具栏上的 **Resume**。

如果对处于 Attention Required 状态的调试作业单击“Resume”按钮，则会显示一条消息，警告您如果恢复，作业将过渡到 Failed 状态。要恢复作业，必须显式单击 OK。然后，应用程序部署将原始部署作业的状态设置为 Failed 并启动回滚作业 (如果适用)。

单步执行调试作业

要单步执行处于 Paused 状态的调试作业，请单击工具栏上的 **Single Step**。将执行计划的下一步 (不跳过)。

如果对处于 Attention Required 状态的调试作业单击 **Single Step** 按钮，则会显示一条消息，警告您如果继续单步执行，作业将过渡到 Failed 状态。要单步执行作业，必须显式单击 OK。然后，应用程序部署将原始部署作业的状态设置为 Failed 并启动回滚作业 (如果适用)。

使用断点

当调试作业处于 Paused 状态时，您可以在计划的任何步骤 (不跳过、尚未执行) 设置断点。恢复作业后，调试器将在该断点之前运行计划的所有步骤 (不跳过)，然后暂停。

断点会持续存留在应用程序部署用户界面 (UI) 会话中。如果您退出应用程序部署 UI，然后重新打开，将丢失在上一会话中设置的任何断点。

作业完成并处于 Succeeded 或 Failed 状态后 (此时不能再调试)，将自动删除任何断点。

要设置断点，请执行以下操作:

1. 在 Steps 区域中，选择要设置断点的步骤。
2. 单击工具栏上的 **Set Breakpoint**。

不能在已启动的步骤中设置断点。

要删除断点，请执行以下操作:

1. 在 Steps 区域中，选择具有断点的步骤。
2. 单击工具栏上的 **Remove Breakpoint**。

不能从已启动的步骤中删除断点。

查看步骤详细信息

选择 (单击) Steps (第 538 页) 区域中的一个步骤时， Parameters (第 539 页) 和 输出 (第 539 页) 区域将显示有关该步骤的信息。

双击 Steps (第 538 页) 区域中的一个步骤时，会在弹出窗口中显示有关该步骤的详细信息。有关所示信息的摘要，请参见 Step Detail 窗口 (第 528 页)。

向部署添加注释

您可以添加或修改与调试作业关联的注释。如果您希望能够使用 Comment 字段的内容区分 Jobs 屏幕中的调试作业，这将非常有用。无论调试作业的状态如何，您均可添加注释。

要添加注释，请执行以下操作:

1. 在调试器中，展开 Details 区域。
2. 在 **Comment** 框中键入注释。
3. 单击 **Save**。
4. 如果要撤消对注释所做的任何更改，请单击 **Reset**。

跳过备份步骤

如果通过选择“Debug”模式从 Deployment 屏幕启动调试器 (请参见启动调试器 (第 540 页))，则可启用或禁用调试作业中的任何备份步骤。默认情况下，备份步骤已启用。在调试过程中，禁用这些步骤可以节省时间。

并非所有组件类型都有备份步骤。例如：脚本、应用程序配置和 OO 流组件没有备份步骤。如果调试作业没有涉及备份步骤的组件，则“Enable backup”处于禁用状态。

您只能在执行任何步骤之前更改 Job Options 设置。作业中的任何步骤开始运行之后，您不能更改这些设置。

如果通过单击 Jobs 页面上的 **Debug** 启动调试器，则 Job Options (第 536 页) 区域中的设置处于禁用状态。这是因为在您进入调试器时已执行一些步骤。

要禁用所有备份步骤，请执行以下操作:

在 Job Options 区域中，清除“Enable backup”框。

跳过层策略或配置

如果通过选择“Debug”模式从 Deployment 屏幕启动调试器 (请参见启动调试器 (第 540 页))，则可启用或禁用与任何层关联的软件策略或 OS 序列。这些步骤通常耗时冗长，并且可能没有必要，具体取决于所调试部署的故障点。

仅当调试作业处于 Paused 状态且未执行有关该层的任何层配置步骤时，您才能更改层设置。

要禁用层策略或 OS 配置，请执行以下操作:

1. 在 Job Selections 区域中，清除每个相关层的 Tier Policy/OS Provision 框。

   只有包含软件策略和/或OS 序列的层才有此设置 (请参见管理层 (第 560 页))。
2. 使用工具栏按钮恢复或单步执行部署作业。

取消调试作业

您可以取消正在运行或已暂停的调试作业。

要取消调试作业，请执行以下操作:

1. 在调试器中，单击工具栏上的 **Cancel**。
2. 在 Cancel Job 对话框中，单击 **Yes**。

   此时，将出现以下情况:
   - 作业过渡到 Cancel-Pending 状态 (以便执行完当前正在进行的任何步骤)。
   - 终止调试作业。
   - 原始部署作业处于 Cancelled 状态。
   - 回滚原始部署作业 (如果适用)。

调试回滚或取消部署作业

您可以使用调试器解决回滚或取消部署作业。该过程类似于调试部署作业。

要调试回滚或取消部署作业，请执行以下操作:

1. 在 Jobs 屏幕上，选择处于 In Progress 状态的回滚或取消部署作业。
2. 单击 **Pause** 暂停作业。
3. 在作业过渡到 Paused 状态之后，单击 **Debug** 启动调试器。

相同的限制适用于调试回滚和取消部署作业。由于您在启动作业步骤之后启动调试器，因此 Job Options (第 536 页) 设置不可用。对于已执行或跳过其步骤的组件或服务器，您不能更改 Job Selections (第 537 页) 设置。

设置权限

概述

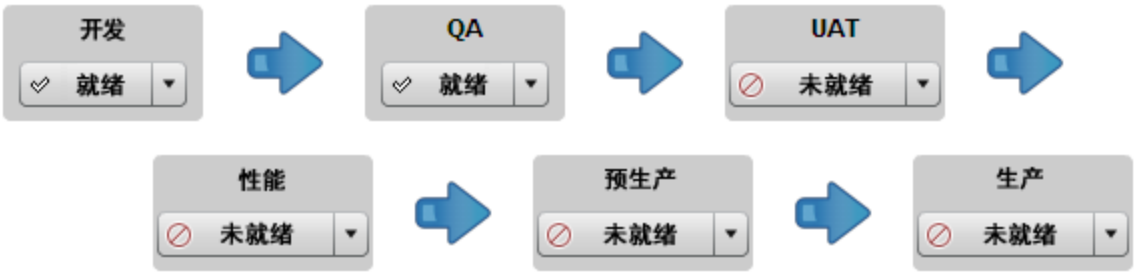
通过应用程序部署可以非常精细地控制下列三点:

- 可以创建、查看、编辑或部署应用程序的用户
- 可以在应用程序组件中使用各种库项目的用户
- 上述用户可以部署应用程序的位置

权限是用来建立这种控制的机制。权限可以帮助您有条不紊地促进应用程序完成软件开发生命周期。管理权限的最简便方法是创建具有一组特定权限的用户组。

由于应用程序部署提供了非常精细的权限，因此只能授权用户将特定应用程序部署到特定环境。总的说来，与 SA 功能相比，它们的功能似乎有所升级，但实际上颇受约束。

请考虑以下这个简单的修补程序生命周期:



例如，为了开发和发布简单的应用程序或修补程序，您可以创建下列用户组:

- 应用程序所有者 (第 547 页)
- QA 环境所有者 (第 548 页)
- QA 工程师 (第 548 页)
- 生产环境所有者 (第 549 页)
- 生产工程师 (第 550 页)

根据所授权限的组合，每个组将具有特定的功能。

应用程序所有者

应用程序所有者组的成员将执行以下类型的任务:

- 创建应用程序
- 将应用程序配置附加到应用程序
- 将软件策略附加到应用程序
- 将程序包附加到应用程序
- 将应用程序部署到开发环境

要执行这些任务，应用程序所有者将需要以下权限:

权限类型	所需权限
操作权限	系统管理：托管的服务器和组 应用程序部署:访问应用程序部署 应用程序部署:创建应用程序 应用程序配置:管理应用程序配置 (读取)

(续)

权限类型	所需权限
	程序包管理:管理程序包 (读取) 策略管理:管理软件策略 (读取)
资源权限	对包含开发环境服务器的设备组的“读取”权限
文件夹权限	应用程序配置、程序包和软件策略的“列表”和“读取”权限
应用程序权限	应用程序所有者可以向其他用户和组授予该应用程序的“查看”、“编辑”和“部署”权限
环境权限	开发环境的“部署”权限

QA 环境所有者

QA 环境所有者组的成员将执行以下类型的任务:

- 创建目标
- 将服务器添加到 QA 环境中的目标

要执行这些任务，QA 环境所有者将需要以下权限:

权限类型	所需权限
操作权限	系统管理：托管的服务器和组 应用程序部署:访问应用程序部署
资源权限	对包含 QA 环境服务器的设备组的“读取”权限
文件夹权限	无
应用程序权限	无
环境权限	QA 环境的“编辑”权限

QA 工程师

QA 工程师组的成员将执行以下类型的任务:

- 将应用程序部署到 QA 环境
- 解决部署作业

要执行这些任务，QA 工程师将需要以下权限:

权限类型	所需权限
操作权限	系统管理：托管的服务器和组 应用程序部署:访问应用程序部署
资源权限	对包含 QA 环境服务器的设备组的“读取”权限
文件夹权限	无
应用程序权限	相关应用程序的“部署”权限
环境权限	QA 环境的“部署”权限

生产环境所有者

生产环境所有者组的成员将执行以下类型的任务:

- 创建目标
- 将服务器添加到生产环境中的目标

要执行这些任务，生产环境所有者将需要以下权限:

权限类型	所需权限
操作权限	系统管理：托管的服务器和组 应用程序部署:访问应用程序部署
资源权限	对包含生产环境服务器的设备组的“读取”权限
文件夹权限	无
应用程序权限	无
环境权限	生产环境的“编辑”权限

生产工程师

生产工程师组的成员将执行以下类型的任务:

- 将应用程序部署到生产环境
- 解决部署作业

要执行这些任务，生产工程师将需要以下权限:

权限类型	所需权限
操作权限	系统管理：托管的服务器和组 应用程序部署:访问应用程序部署
资源权限	对包含生产环境服务器的设备组的“读取”权限
文件夹权限	无
应用程序权限	相关应用程序的“部署”权限
环境权限	生产环境的“部署”权限

权限类型

有各种类型的权限影响着您在应用程序部署用户界面 (UI) 中可以查看和访问的内容:

操作权限 (第 551 页)

三种类型的操作权限影响对应用程序部署的访问:

- “托管服务器和组”权限将确定您能否查看托管服务器和设备组。您需要使用此功能将服务器添加到应用程序部署目标。
- 应用程序部署权限将确定您能否访问应用程序部署 UI、能否创建应用程序以及是否具有应用程序部署管理员权限。
- 三个附加权限将确定您能否在应用程序组件中引用软件策略、程序包、应用程序配置或 OS 序列或者将其附加到层。

所有操作权限由 SA 系统管理员设置。

设备组权限 (第 555 页)

这些权限将确定您可以访问哪些设备组。您需要访问设备组，才能将该设备组中的服务器添加到应用程序部署目标。

设备组权限由 SA 系统管理员设置。

库文件夹权限 (第 555 页)

文件夹权限将进一步确定您可以在应用程序中引用哪些特定的库项目 (软件策略、程序包、应用程序配置或 OS 序列)或者将哪些特定的库项目附加到层。

文件夹权限由 SA 系统管理员设置。

应用程序部署权限 (第 557 页)

有两种类型的应用程序部署权限:

- 应用程序权限确定可以查看、编辑或部署特定应用程序的用户。

  应用程序权限由创建应用程序的用户或已被授予该应用程序的“编辑”权限的任何用户设置。应用程序部署管理员还可以设置应用程序权限。
- 环境权限确定可以将应用程序部署到特定环境的用户。它们也决定了可以在环境中创建或修改应用程序部署目标的用户。环境权限由应用程序部署管理员设置。

本节描述完成常见的应用程序部署任务所需的最低权限。有关权限更全面的讨论，请参见《SA 10.50 管理指南》。

操作权限

以下 SA 操作权限与应用程序部署相关。

操作权限由 SA 系统管理员使用 SA 客户端中的 Administration 屏幕授予给用户组。有关详细信息，请参见《SA 10.50 管理指南》。

托管服务器和组权限

如果您需要创建或修改应用程序部署目标，则必须能够查看和管理服务器与服务器组:

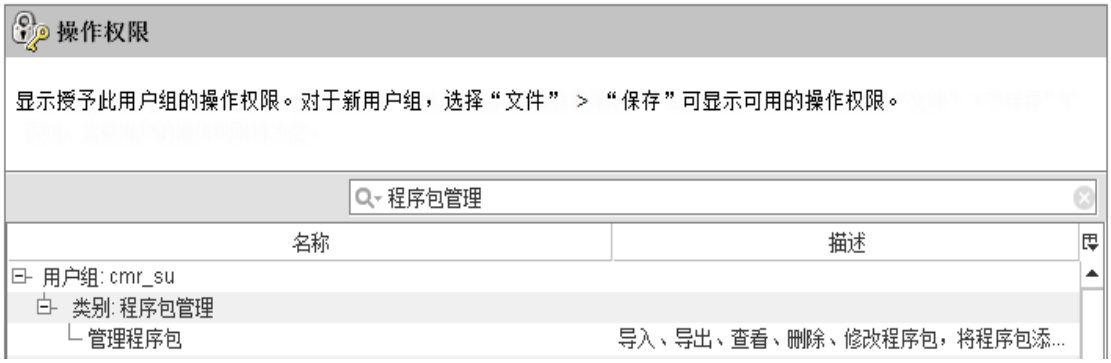
应用程序部署访问权限

以下权限决定了您能否访问应用程序部署 UI 且可在其中执行哪些操作:

- Access Application Deployment 使您能够启动应用程序部署。如果您没有此权限，则应用程序部署菜单项不会显示在 Tools 菜单上，并且您无法使用 Web 浏览器登录到应用程序部署。
- Create Applications 使您能够在应用程序部署用户界面中创建应用程序。每个应用程序也有各自的查看/编辑/部署权限 (请参见应用程序权限 (第 557 页))。
- 将应用程序部署到特定环境时需要更多权限 (请参见环境权限 (第 559 页)和设备组权限 (第 555 页))。
- Manage Application Deployment 使您能够访问应用程序部署 UI 中的 Administration 屏幕。它还支持创建和修改任何应用程序、目标或环境。

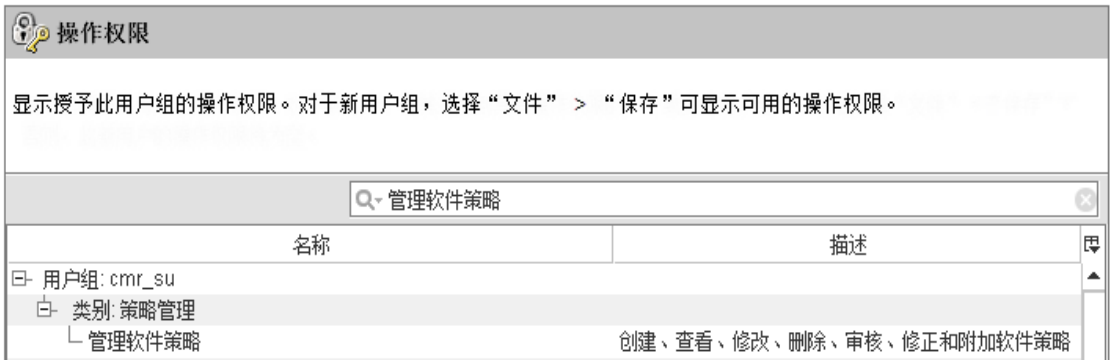
程序包权限

您必须具有 Manage Package (Read) 权限，才能在应用程序组件中引用软件包或将其附加到层。



策略权限

您必须具有 Manage Software Policy (Read) 权限，才能在应用程序组件中引用软件包或将其附加到层。



管理应用程序配置权限

您必须具有 Manage Application Configurations (Read) 权限，才能在组件中引用应用程序配置。

管理 OS 序列权限

您必须具有 Manage OS Sequence (Read) 权限，才能在组件中引用应用程序配置。

服务器池权限

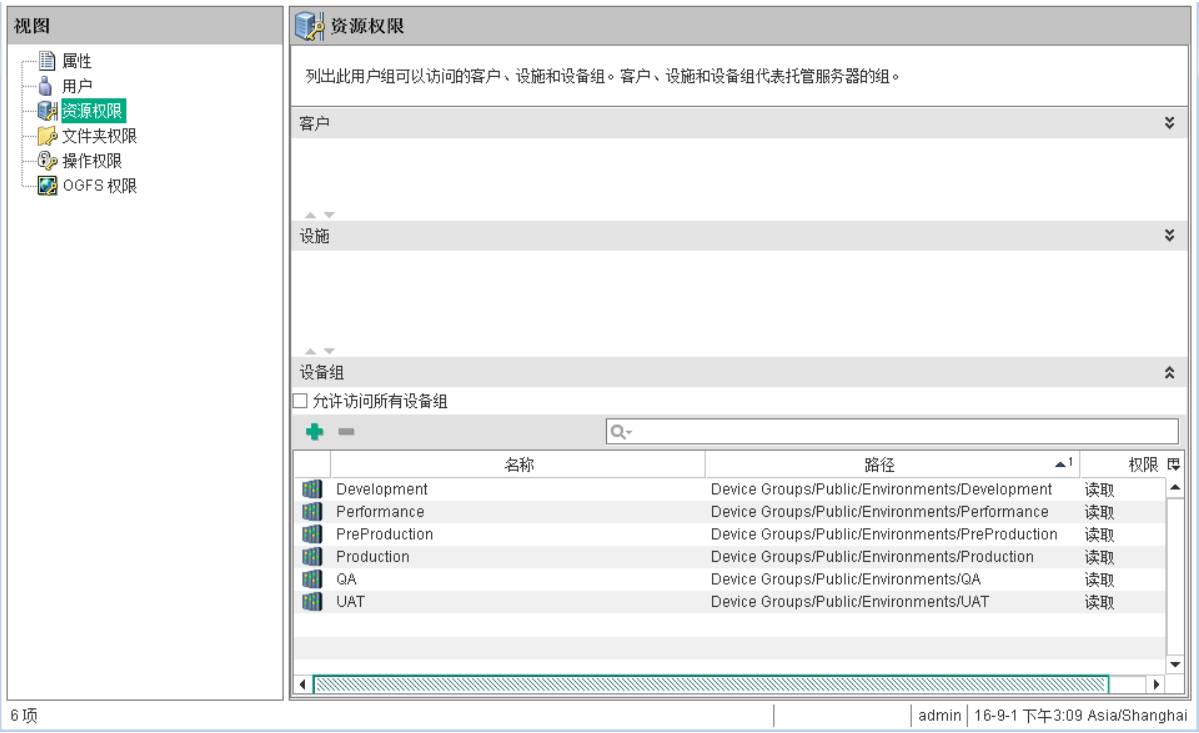
如果要在部署时使用 OS 序列配置服务器，则必须具有 Server Pool 权限。

有关详细信息，请参见 在部署时配置服务器 (第 499 页)。

设备组权限

如果需要创建或修改应用程序部署目标，您必须对包含将在这些目标中使用的服务器的设备组具有读取访问权限。例如，QA 所有者将需要对 QA 服务器设备组具有读取访问权限。

设备组权限由 SA 系统管理员使用 SA 客户端中的 Administration 屏幕授予给用户组。有关详细信息，请参见《SA 10.50 管理指南》。



在此示例中，上述环境中的所有服务器将对应用程序部署 UI 的 Add Servers to Target 对话框中的设备组可用。

库文件夹权限

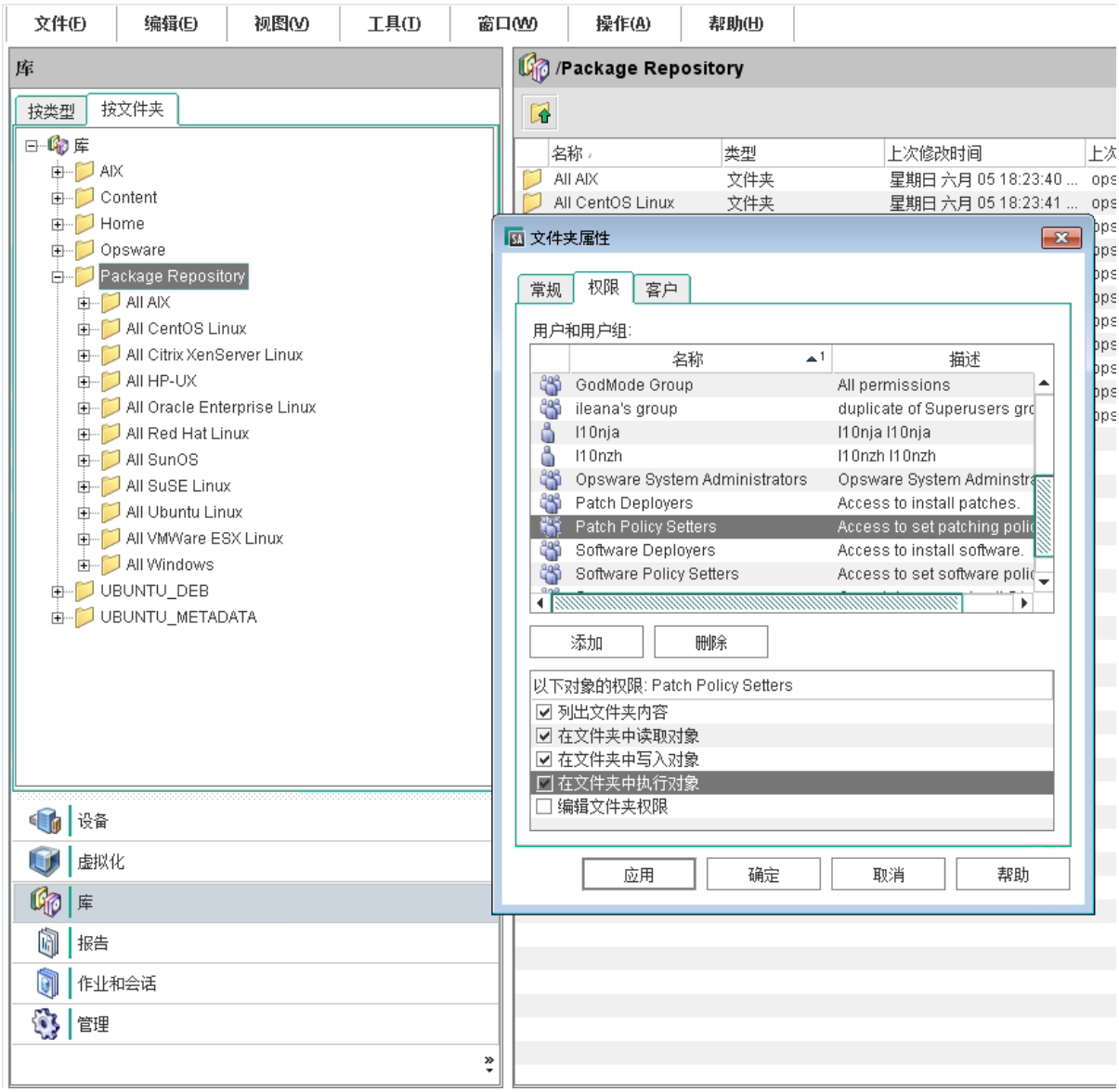
本主题中讨论的权限由 SA 系统管理员在 Library 屏幕上的 Folder 级别设置。

如果需要创建或修改的应用程序需要以下项，您必须在 Library 中对其具有 Read 和 List 权限：

- 软件策略
- 程序包
- 应用程序配置
- OS 序列

可以在 Folder 级别控制对 Library 中的项目的访问权限。在何处设置程序包的库文件夹权限(第 556 页)显示了如何设置程序包的这些权限的示例。

在何处设置程序包的库文件夹权限



您还必须对各类库项目的相关 Manage 设置具有 Read 能力，例如 Manage Package (请参见操作权限 (第 551 页))。

应用程序部署权限

有两种类型的权限决定了您在应用程序部署 UI 中可以执行的操作:

- 应用程序权限 (第 557 页)
- 环境权限 (第 559 页)

应用程序权限由应用程序所有者或已被授予该应用程序的 Write 权限的任何用户进行设置。应用程序部署管理员还可以设置应用程序权限。

环境权限由应用程序部署管理员设置。

应用程序权限

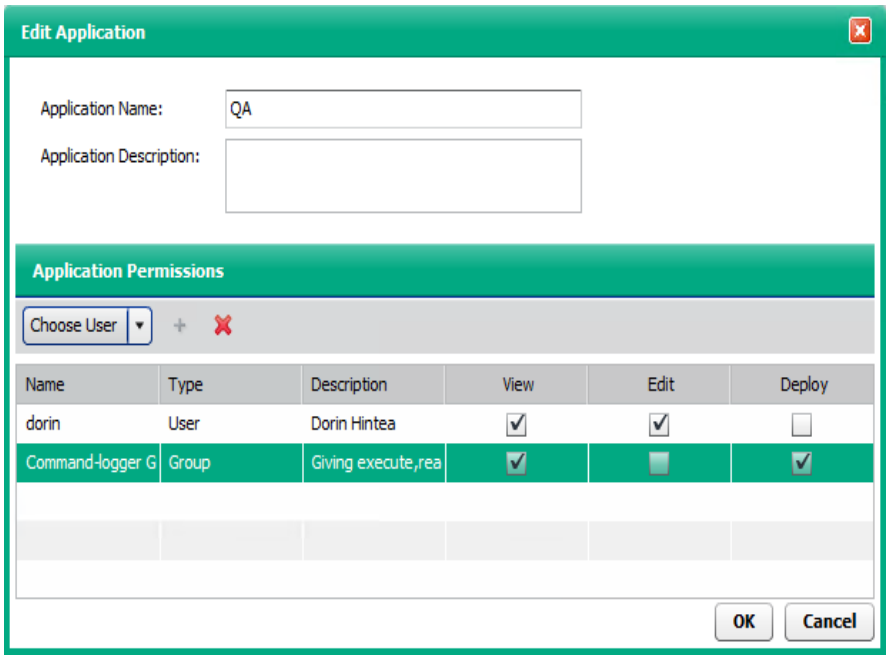
有三种类型的应用程序权限:

- View - 确定应用程序在 Manage Applications 工具和 Select Release 下拉列表中是否可见。
- Edit - 确定可以编辑应用程序属性的用户。
- Deploy - 确定可以部署应用程序的用户。还需要部署到相关环境的权限 (请参见环境权限 (第 559 页))。

默认情况下，只有应用程序的创建者 (所有者) 和应用程序部署管理员才有权查看或编辑应用程序。

如果希望其他用户查看、编辑或部署您的应用程序，则必须向其显式授予相应权限。

在何处设置应用程序权限



Choose User 下拉列表中显示的用户和组与 SA 客户端中保留的用户和组相同。

只有应用程序所有者 (即，已被显式授予该应用程序的 Edit 权限的用户) 和应用程序部署管理员才能修改应用程序权限。

要授予访问应用程序的权限，请执行以下操作:

1. 转到 Application 屏幕 (单击左下角的 Applications)。
2. 单击  图标打开 Manage Applications 工具。
3. 在 Manage Applications 工具中，找到并选择要使用的应用程序。
4. 单击 **Edit Properties** (或双击要编辑的应用程序)。将打开 Edit Application 对话框。
5. 在 Application Permissions 部分中，从 Choose User 列表中选择单个用户或用户组。
6. 单击 **Add User**。
7. 选择要授予的 View、Edi 或 Deploy 权限。
8. 对于要授予权限的每个用户或组，重复步骤 5 到步骤 7。
9. 可选：如果要从列表中删除一个用户或组，请在列表中选择该用户或组，然后单击 。
10. 单击 **OK**，关闭 Edit Application 对话框并保存所做的更改。

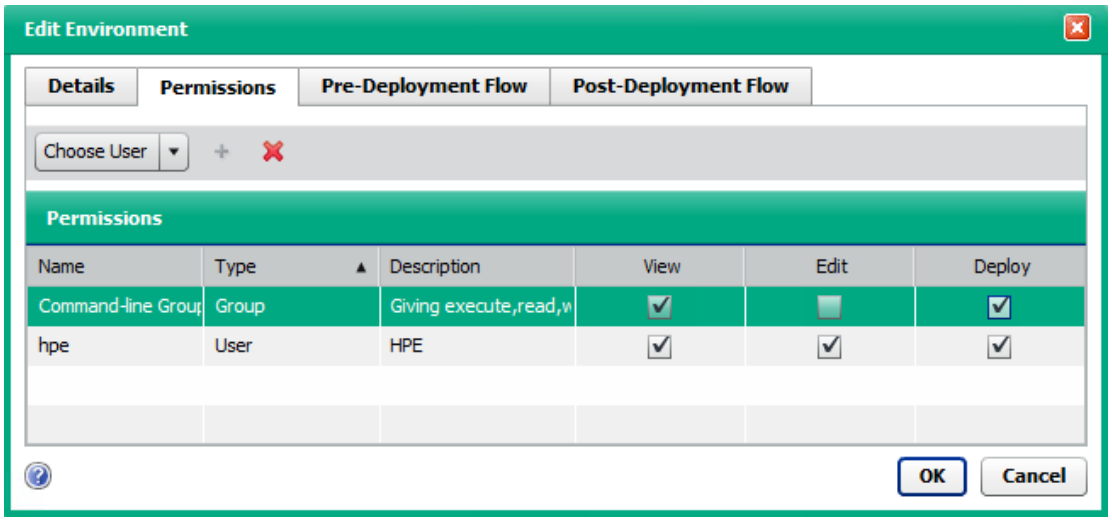
环境权限

有三种类型的环境权限:

- View - 确定可以查看目标结构的用户。仅当您具有相关环境的 View 权限时，才会在 Manage Targets 工具和 Select Target 下拉列表中看到目标。
- Edit - 确定可以在环境中修改目标的用户。此配置包括：
  - 创建目标、重命名或删除目标
  - 将服务器添加到目标
  - 从目标中删除服务器
- Deploy - 确定可以将应用程序部署到此环境中的目标的用户。还需要部署到相关应用程序的权限 (请参见应用程序权限 (第 557 页))。

默认情况下，仅应用程序部署管理员有权查看、编辑或部署到任何环境。如果其他用户需要查看、编辑或部署到特定环境，应用程序部署管理员必须显式授予他们执行此操作的权限。

在何处设置环境权限



Choose User 下拉列表中显示的用户和组与 SA 客户端中保留的用户和组相同。

只有应用程序部署管理员才能修改环境权限。

要授予访问环境的权限，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 Environments。
3. 在右侧面板中，选择要使用的环境。
4. 单击 **Edit Properties** (或双击要编辑的环境)。将打开 Edit Environment 对话框。
5. 在 Environment Permissions 下，从 Choose User 列表中选择单个用户或用户组。
6. 单击 **Add User**。.
7. 选择要授予的 View、Edi 或 Deploy 权限。
8. 对于要授予权限的每个用户或组，重复步骤 5 到步骤 7。
9. 可选：如果要从列表中删除一个用户或组，请在列表中选择该用户或组，然后单击  。
10. 单击 **OK**，关闭 Edit Environment 对话框并保存所做的更改。

管理应用程序部署

本节说明如何执行 Administration 屏幕上提供的功能。它包含以下主题:

您必须具有 Manage Application Deployment 权限，才能查看应用程序部署用户界面中的 Administration 屏幕并修改本节中讨论的设置 (请参见概述 (第 546 页))。

管理层

作为应用程序部署管理员，您可以管理可用于创建和部署应用程序的层。您可执行以下操作:

- 创建新层 (第 561 页)
- 创建新层组 (第 562 页)
- 修改层的层次结构 (第 562 页)
- 删除层或层组 (第 562 页)

下面简要描述其中每个过程，所有主题均假设您具有应用程序部署管理员权限。

创建新层

您必须具有 Manage Application Deployment 权限，才能管理层。

您可以创建一个新层，并将它放在现有层的层次结构中的任意位置。创建层之后，具有应用程序和环境的 Edit 权限的任何用户均可使用该层。

要创建新层，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Tiers**。
3. 在右侧面板中，单击 **Create Tier**。将打开 New Tier 对话框。
4. 指定以下属性:
   - Name - 输入唯一名称 (请参见命名规则)。
   - Group - 在要创建新层的层的层次结构中找到并选择相应组。
   - Platform - 选择 Windows 或 UNIX。
   - Backup Directory - (可选) 指定需要回滚时此层中的组件将其备份文件存储在目标服务器上的位置。
   - 可选： Associated Policy - 单击 **Select Policy**，然后从 Library 中选择软件策略。这将确保支持该层的功能所需的中间件在目标服务器上均已就绪。
   - Deploy Behavior - 指定如果关联策略在部署时尚未附加到服务器，应用程序部署应该执行哪些操作。
   - 可选： OS Provisioning - 单击 **Select OS Sequence**，然后从 Library 中选择 OS 序列。使用此选项，可将相应的操作系统安装和配置到目标中包含的任何未配置服务器上。有关详细信息，请参见在部署时配置服务器 (第 499 页)。

   您还可以修改任何现有层的这些属性。
5. 单击 **OK**。

创建新层组

您必须具有 Manage Application Deployment 权限，才能管理层组 (请参见设置权限 (第 546 页))。

您可以在层的层次结构中创建新层组。然后，可以将现有层移动到新层组中，或者为其创建新层。

要创建新层组，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Tiers**。
3. 在右侧面板中，单击 **Create Tier Group**。将打开 New Tier Group 对话框。
4. 指定以下属性:
   - Name - 输入唯一名称 (请参见命名规则)。
   - Group - 在其下方要创建新层组的层的层次结构中找到并选择“父”组。
5. 单击 **OK**。

修改层的层次结构

您必须具有 Manage Application Deployment 权限，才能管理层 (请参见设置权限 (第 546 页))。

您可以将现有层或层组移动到任何其他现有层组中。

要将层移动到其他层组，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Tiers**。
3. 在右侧面板中，双击任何层或层组。
4. 在要移动选定项的层的层次结构中找到并选择“父”组。
5. 单击 **OK**。

删除层或层组

您必须具有 Manage Application Deployment 权限，才能管理层 (请参见设置权限 (第 546 页))。

您可以删除任何未使用的现有层或层组 (此类层或层组原本在当前部署到目标服务器或者属于发行版或目标一部分的版本中使用)。如果层组中的任何层正在使用，则不能删除该层组。

要删除层或层组，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Tiers**。
3. 在右侧面板中，选择要删除的层或层组。
4. 单击“Delete”。
5. 单击 **Yes** 确认。

如果要删除正在使用的层，则必须先执行以下操作:

- 取消部署并删除使用该层的所有版本。
- 删除使用该层的任何发行版。
- 删除使用该层的任何目标。
- 只能删除空层组。

层一旦使用，就很难删除。

管理环境

作为应用程序部署管理员，您可以管理应用程序部署的可用环境。您可执行以下操作:

- 创建新环境 (第 563 页)
- 更改环境权限 (第 565 页)
- 指定环境的 OO 流 (第 565 页)
- 删除环境 (第 565 页)

下面简要描述其中每个过程，所有主题均假设您具有应用程序部署管理员权限。

创建新环境

您必须具有 Manage Application Deployment 权限，才能管理环境 (请参见设置权限 (第 546 页))。

您可以创建新环境并使其可用于特定用户或用户组。您可以指定某一 OO 流在每个部署之前启动，另一 OO 流在每个部署之后启动。

应用程序部署上下文中的环境将镜像为 SA 设备组。创建新环境时，对应的设备组将在下次发生同步后可见。

- 要强制立即同步，请右键单击环境并选择 **Export**。
- 要查看镜像设备组，请右键单击环境并选择 **View Device Group**。

要创建新环境，请执行以下操作：

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Environments**。
3. 在右侧面板中，单击“Create Environment”按钮：. 将打开 New Environment 对话框。
4. 在 Details 选项卡中，指定以下属性：
   - Name - 输入唯一名称 (请参见命名规则)。
   - Initial Status - 指定环境的默认状态。这将确定环境在默认情况下能否接受应用程序，或者是否需要每个版本的显式权限。有权部署到此环境的用户将能够在 Deployment 屏幕上更改此状态。
   - Undeploy - 选中此框可建立默认的自动取消部署设置。用户可以覆盖此设置。
5. 在 Permissions 选项卡中，指定有权查看、编辑和部署到此环境的用户。
   - 具有 View 权限的用户可以在 Select Targets 下拉列表 (在 Targets 和 Deploy 屏幕上) 和 Manage Targets 工具 (可从 Targets 屏幕访问) 中查看此环境。
   - 具有 Edit 权限的用户可以打开 Edit Environment 对话框并更改环境的属性。他们还可以创建和修改目标。
   - 具有 Deploy 权限的用户可以将应用程序部署 (前提是有权部署) 到此环境。

     要指定用户权限，请执行以下步骤：

     a. 从 Choose User 列表中选择单个用户或用户组。
     b. 单击“Add User”按钮：。
     c. 选择要授予的 View、Edi 或 Deploy 权限。
     d. 对于要授予权限的每个用户或组，重复步骤 a 到步骤 c。
     e. 可选：如果要从列表中删除一个用户或组，请在列表中选择该用户或组，然后单击 。
6. 可选：在 Pre-Deployment Flow 选项卡中，选择要在此环境中的目标服务器上进行任何部署之前启动的 OO 流。例如，您可能要在部署之前在生产环境中禁用监控。

单击 Change 按钮，然后从 OO 库中选择一个流。双击要更改其值的任何参数对应的 Value 单元格 (请参见参数和特殊变量 (第 474 页))。

在指定部署前或部署后 OO 流的参数值时，您可以引用全局参数或发行版参数 (请参见参数和特殊变量 (第 474 页))。

7. 可选：在 Post-Deployment Flow 选项卡中，选择要在此环境中的目标服务器上进行所有部署之后立即启动的 OO 流。例如，您可能要在部署之后重新启用监控。
8. 单击 **OK**，关闭 New Environment 对话框并保存所做的更改。

更改环境权限

您必须具有 Manage Application Deployment 权限，才能管理环境 (请参见设置权限 (第 546 页))。

您可以更改任何环境的“View”、“Edit”和“Deploy”权限。有关说明，请参见创建新环境 (第 563 页)下的步骤 5。

指定环境的 OO 流

您必须具有 Manage Application Deployment 权限，才能管理环境 (请参见设置权限 (第 546 页))。

您可以更改任何环境的部署前和部署后流。有关说明，请参见创建新环境 (第 563 页)下的步骤 6 和步骤 7。

删除环境

您必须具有 Manage Application Deployment 权限，才能管理环境 (请参见设置权限 (第 546 页))。

您可以删除任何未使用的现有环境 (此类环境原本在当前部署到目标服务器或属于生命周期一部分的版本中使用)。

要删除环境，请执行以下操作：

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Environments**。

3. 在右侧面板中，选择要删除的环境。
4. 单击“Delete”。
5. 单击 **Yes** 确认。

如果要删除正在使用的环境，则必须先取消部署使用该环境的所有版本。此外，您还必须从该环境所属的任何生命周期中将其删除。

管理生命周期

作为应用程序部署管理员，您可以管理应用程序部署的可用生命周期。您可执行以下操作：

- 创建生命周期 (第 566 页)
- 修改生命周期 (第 567 页)
- 删除生命周期 (第 567 页)

下面简要描述其中每个过程，所有主题均假设您具有应用程序部署管理员权限。

创建生命周期

您必须具有 Manage Application Deployment 权限，才能管理生命周期 (请参见设置权限 (第 546 页))。

生命周期用于表示软件应用程序经历的各个阶段，从开发环境到不同级别的测试环境，最后到生产环境。借助生命周期，可以显式允许或防止部署到各种环境，从而控制应用程序从一个环境提升到下一个环境。

您可以创建新生命周期、更改现有生命周期和删除未使用的生命周期。

在任何情况下，应用程序部署生命周期均不由 SA 强制执行。它们仅在应用程序部署上下文中有意义。

要创建新生命周期，请执行以下操作：

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Lifecycles**。
3. 在右侧面板中，单击“Create Lifecycle”按钮：  . 将打开 New Lifecycle 对话框。

4. 指定以下属性:
   - Name - 输入唯一名称 (请参见命名规则)。
   - Lifecycle - 使用水平箭头将一个或多个可用环境移动到 Lifecycle 框中；使用垂直箭头修改这些环境在 Lifecycle 中的顺序。
5. 单击 **OK**，关闭 New Lifecycle 对话框并保存所做的更改。

修改生命周期

您必须具有 Manage Application Deployment 权限，才能管理生命周期 (请参见设置权限 (第 546 页))。

您可以重命名或重新排列生命周期中的环境。

要编辑生命周期，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Lifecycles**。
3. 在右侧面板中，选择要编辑的生命周期。
4. 单击 **Edit Properties**。将打开 Edit Lifecycle 对话框。
5. 可选：输入生命周期的新名称。
6. 可选：使用水平箭头将一个或多个可用环境移动到 Lifecycle 框中；使用垂直箭头修改这些环境在 Lifecycle 中的顺序。
7. 单击 **OK**，关闭 Edit Lifecycle 对话框并保存所做的更改。

删除生命周期

您必须具有 Manage Application Deployment 权限，才能管理生命周期 (请参见设置权限 (第 546 页))。

您可以删除任何未使用的现有生命周期 (此类生命周期原本在当前部署到目标服务器的版本或发行版中使用)。

要删除生命周期，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Lifecycles**。

3. 在右侧面板中，选择要删除的生命周期。
4. 单击 **Delete**。
5. 单击 **Yes** 确认。

要删除某个生命周期，则该生命周期不能与任何发行版或版本关联。

管理脚本

作为应用程序部署管理员，您可以管理用于部署、备份、回滚和取消部署操作的脚本。这些脚本由代码、配置文件和 Windows 注册表组件引用。您可执行以下操作:

- 创建新脚本 (第 568 页)
- 修改现有脚本 (第 569 页)
- 删除脚本 (第 569 页)

下面简要描述其中每个过程，所有主题均假设您具有应用程序部署管理员权限。

在创建版本时将会复制脚本的内容。对脚本所做的更改不影响以前创建的版本。

创建新脚本

您必须具有 Manage Application Deployment 权限，才能管理脚本。

脚本在应用程序组件中引用。它们用于简化部署、备份、回滚和取消部署操作。系统提供了一组“预置”的基本脚本。您可以修改这些脚本，也可以创建新脚本。有权创建应用程序的用户可以在创建组件时引用这些脚本。

要创建新脚本，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Scripts**。
3. 在右侧面板中，单击 **Create Scrip**。将打开 New Script 对话框。
4. 指定以下属性:
   - Name - 输入唯一名称 (请参见命名规则 (第 412 页))。
   - Component - 将引用此脚本的组件类型。

- Purpose - 选择 Deploy、Undeploy、Backup 或 Rollback。不同的组件在不同的部署阶段中使用脚本。例如，只有 Windows 注册表组件使用 Deploy 脚本。
- Platform - 选择 Windows 或 UNIX。
- Primary - 如果有多个脚本对此平台上的这类组件具有此用途，请选中此框，这是供新组件使用的默认脚本。在开发或测试新脚本时，这将非常有用。
- Type - 选择脚本类型。可用选项取决于指定的平台。
- Content - 输入构成脚本的实际指令。

5. 您还可以修改任何现有脚本的上述属性。
6. 单击 **OK**，保存所做的更改并关闭 New Script 对话框。

修改现有脚本

您必须具有 Manage Application Deployment 权限，才能管理脚本 (请参见设置权限 (第 546 页))。

您可以使用 Edit Script 对话框修改任何现有脚本。

要修改脚本，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Scripts**。
3. 在右侧面板中，选择要修改的脚本。
4. 单击 **Edit Properties**。将打开 Edit Script 对话框。
5. 更改脚本的属性 (请参见创建新脚本 (第 568 页))。
6. 单击 **OK**，保存所做的更改并关闭 Edit Script 对话框。

删除脚本

您必须具有 Manage Application Deployment 权限，才能管理脚本 (请参见设置权限 (第 546 页))。

您可以删除任何未使用的现有脚本 (此类脚本原本在发行版中使用)。

要删除脚本，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 Administration)。
2. 在左侧面板中，单击 **Scripts**。
3. 在右侧面板中，选择要删除的脚本。
4. 单击 **Delete**。
5. 单击 **Yes** 确认。

要删除的脚本不能由任何发行版引用。

管理代码组件源类型

应用程序开发人员使用代码组件指定应用程序所需的文件。代码组件源类型可指示应用程序部署如何访问这些文件。

可以启用或禁用源类型。无论何时，必须至少启用一种源类型。不能禁用引用的源类型。在可以禁用某源类型之前，必须删除引用该源类型的任何版本并修改引用该源类型的任何发行版。

应用程序部署使用开源构建系统 CruiseControl，以此管理源类型是文件系统或特定源代码控制系统 (如 CVS 或 Subversion) 的代码组件。对于这些源类型，您必须提供包含必需的 CruiseControl 配置设置的 XML 代码片段。此处提供的示例适用于文件系统 (第 571 页)和源代码控制系统 (第 573 页)。

有关配置 CruiseControl 的详细信息，请参见以下资源:

http://cruisecontrol.sourceforge.net/main/configxml.html

编辑代码组件源类型

您必须有权管理应用程序部署，才能管理代码组件源类型。

通过 Edit Code Component Source Type 对话框，可以配置任意代码组件源类型的设置。无论编辑何种代码组件源，您均必须先打开此对话框。

要编辑代码组件源，请执行以下操作:

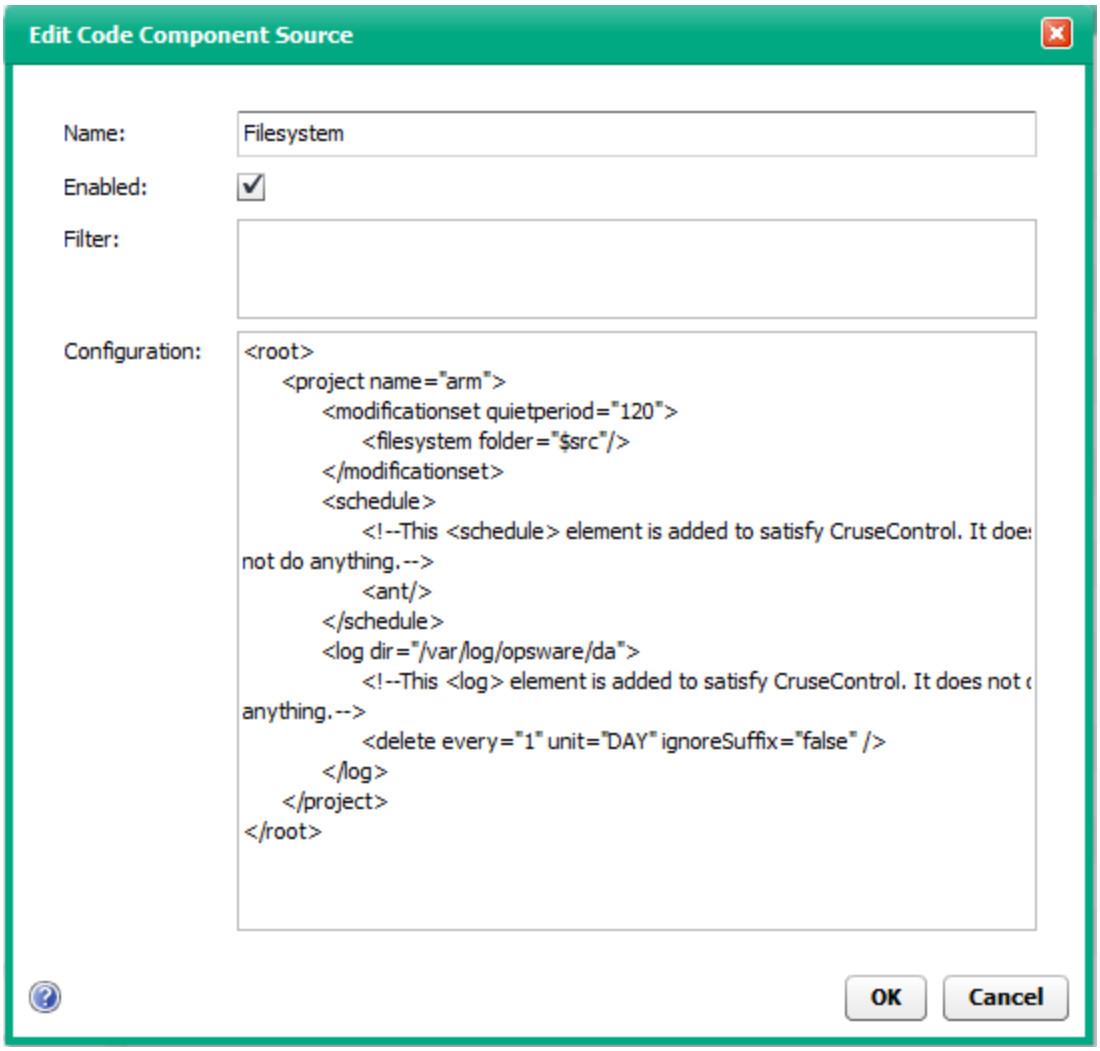
1. 转到 Administration 屏幕 (单击左下角的 **Administration**)。
2. 在左窗格中，选择 **Code Component Source Types**。

3. 在右窗格中，执行以下操作之一：
   - 要创建新的源类型，请单击 **Create Source Type**。
   - 要修改现有源类型，请选择该源类型，然后单击 **Edit Properties**。将打开 Edit Lifecycle 对话框。
4. 要使此代码组件源类型可用于应用程序开发人员，请选择 **Enabled**。
5. 请按照以下详细说明，编辑特定组件源类型：
   - 文件系统
   - URL
   - 源代码控制系统

文件系统

要配置代码组件的文件系统源，必须指定此处所示的 CruiseControl 设置。

文件系统源类型的设置



`quietperiod` 是指从最后一次修改到可以创建新版本之间必须达到的秒数。

`$src` 变量表示在代码组件中指定的源目录。

URL

URL 代码组件源的可配置设置仅限其 Name 和 Enable 字段。应用程序开发人员必须指定 URL。请参见 使用 URL (第 447 页)。

源代码控制系统

为了与源代码控制系统 (如 CVS 或 Subversion) 集成，应用程序部署嵌入了开源构建系统 CruiseControl。CruiseControl 配置作为源类型的一部分存储为 XML 代码片段。仅需添加新的 XML 代码片段，即可添加其他源代码控制系统。

Server Automation 核心上的时间必须与源代码控制系统服务器同步。否则，可能会因时间差而无法准确判断代码组件中包含的文件。

请确保先将源代码控制系统客户端软件安装在 SA 核心服务器上，然后再尝试将其与应用程序部署集成。

要集成源代码控制系统，请执行以下操作:

将源代码控制系统客户端软件安装在 SA 核心服务器上。

在 SA 核心服务器上，为要包含在代码组件中的源创建客户端副本。在 CVS 和 Subversion 中，这称为“签出”存储库。

打开 Edit Code Component Source 对话框 (请参见上文的编辑代码组件源类型 (第 570 页))。

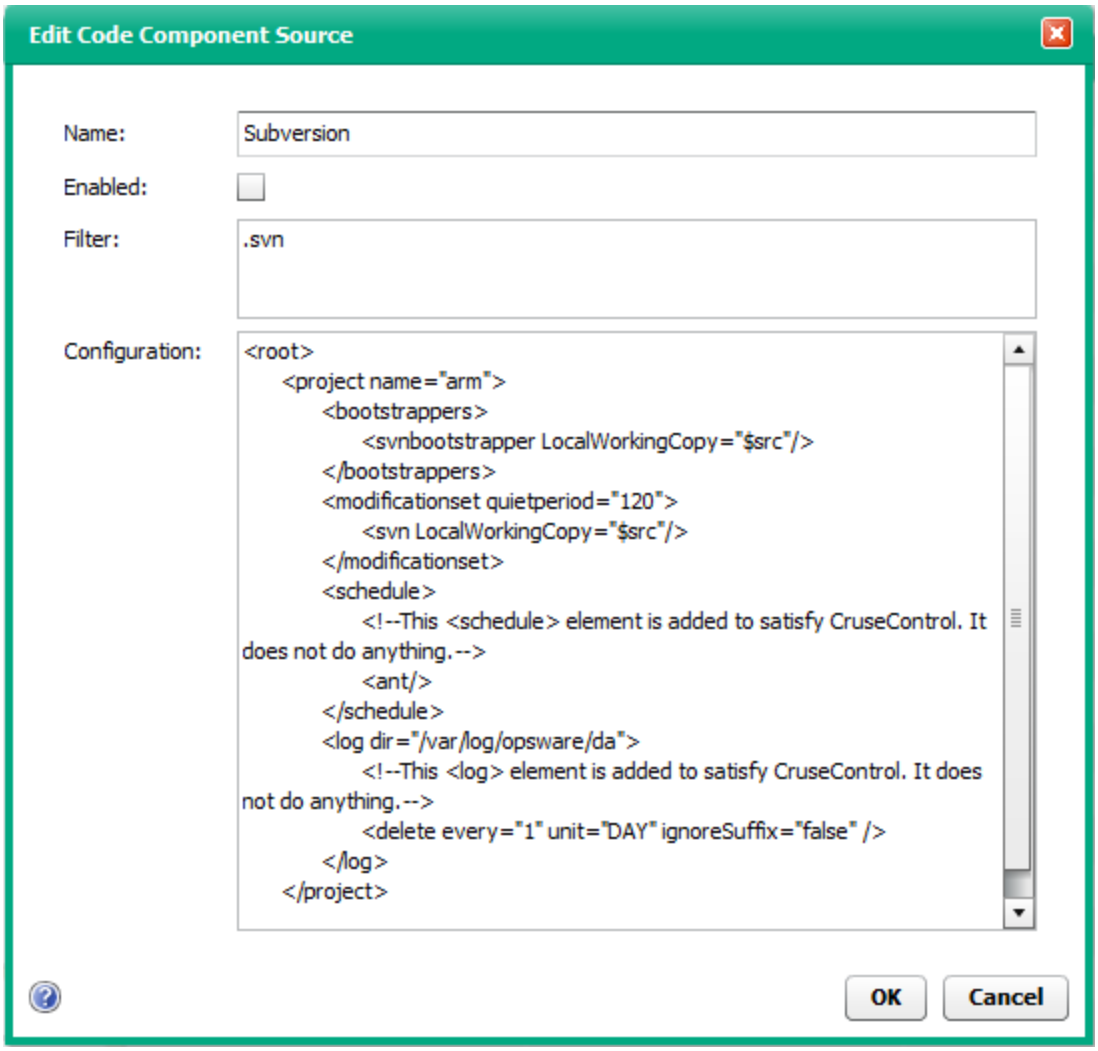
添加 CruiseControl 配置以及源代码控制系统所需的任何筛选条件。

完成后，应用程序开发人员可以在代码组件中引用指向此客户端副本的路径。请参见使用源代码管理系统 (第 446 页)。

示例： Subversion

以下是 Subversion 代码组件源的示例配置。

Subversion 源类型设置



要在 SA 核心服务器上签出此 Subversion 模块，可使用以下命令：

```
svn checkout URL
```

例如：

```
svn checkout https://mySVNserver.mycompany.com:444/svn/proj1/trunk
```

在 SA 多核心或多切分环境中设置 SVN 客户端

要避免在多核心或多切分环境中的所有服务器上安装 Subversion (SVN) 客户端，您可以使用网络文件系统 (NFS) 安装。以下过程已经 Linux 服务器测试：

在主控核心服务器上：

1. 将 SVN 客户端 (例如，CollabNet Subversion 客户端) 安装在以下目录中:
   `/opt/CollabNet_Subversion`
2. 将以下行添加到 `/etc/exports` 文件:
   `/opt/CollabNet_Subversion *(ro)`
3. 使用以下命令重新启动 NFS 服务:
   `/etc/init.d/nfs restart`

在第二台服务器 (和后续服务器) 上:

1. 安装到主控核心的导出驱动器。例如，执行以下步骤:

   ```
   cd /opt
   mkdir CollabNet_Subversion
   mount <MasterCore>:/opt/CollabNet_Subversion /opt/CollabNet_Subversion
   ```

   此处，MasterCore可以是主控核心服务器的 DNS 名称或 IP 地址。
2. 从 `/usr/bin/svn` 创建指向 SVN 的链接:

   ```
   ln -s /opt/CollabNet_Subversion/bin/svn /usr/bin/svn
   ```

现在，您可以像 SVN 安装在第二台服务器上一样使用它。

可以使用相同的 NFS 安装策略设置 CVS 客户端。

管理应用程序设置

您可以管理确定应用程序部署如何运行的应用程序设置。

要修改应用程序设置，请执行以下操作:

1. 转到 Administration 屏幕 (单击左下角的 **Administration**)。
2. 在左窗格中，选择 **Application Settings**。
3. 修改以下一个或多个设置:
   - 版本编号 (第 576 页)
   - 实时目标流 (第 577 页)

- 代码和配置文件组件基目录 (第 577 页)
- 全局参数 (第 580 页)

4. 单击 **Save**，保存所做的更改。

您必须具有 Manage Application Deployment 权限，才能修改这些设置。

版本编号

此组包含以下三个设置:

初始版本

指定应该如何为发行版的初始版本命名或编号。默认值为 1。如果您指定一个数字，则应用程序部署将尝试在每次创建新版本时增加该数字。

只要用户创建发行版的第一个版本，您在此处指定的值就会显示在 Create New Version 对话框的 Version 框中。用户可以覆盖初始值。

版本上下文

指定应该如何构造版本标签。

如果要将发行版名称与初始版本号连在一起形成版本标签，请选择 Use Release Name。

软件部署方法

选择部署代码和程序包组件的默认方法。

只要您创建新版本，应用程序部署就会创建一个程序包，其中包含指定的文件或程序包和所需的任何脚本。此外，您可以指定以下选项之一:

- Attach Policy as Default - 在创建新版本时还会创建策略；部署此版本时，在每个目标服务器上附加此策略并进行修正。
- Install Package as Default - 在每个目标服务器上“临时”安装程序包。
- Always Attach Policy - 始终创建策略；附加并修正。
- Always Install Package - 始终“临时”安装程序包。

如果选择 “as Default” 选项之一，则用户可以在创建或编辑组件时覆盖部署方法 (请参见组件 (第 442 页))。

如果选择 “Always” 选项之一，则将显示 “Locked” 指示器，并且用户无法覆盖部署方法。

实时目标流

此设置指定应用程序部署将在部署时用于创建实时目标的 Operations Orchestration (OO) 流 (请参见实时目标 (第 496 页))。

OO 库中必须存在此流。有关详细信息，请参见在库中查找并选择项目 (第 472 页)。

此设置仅当 SA 配置为与 OO 集成时才可用。

代码和配置文件组件基目录

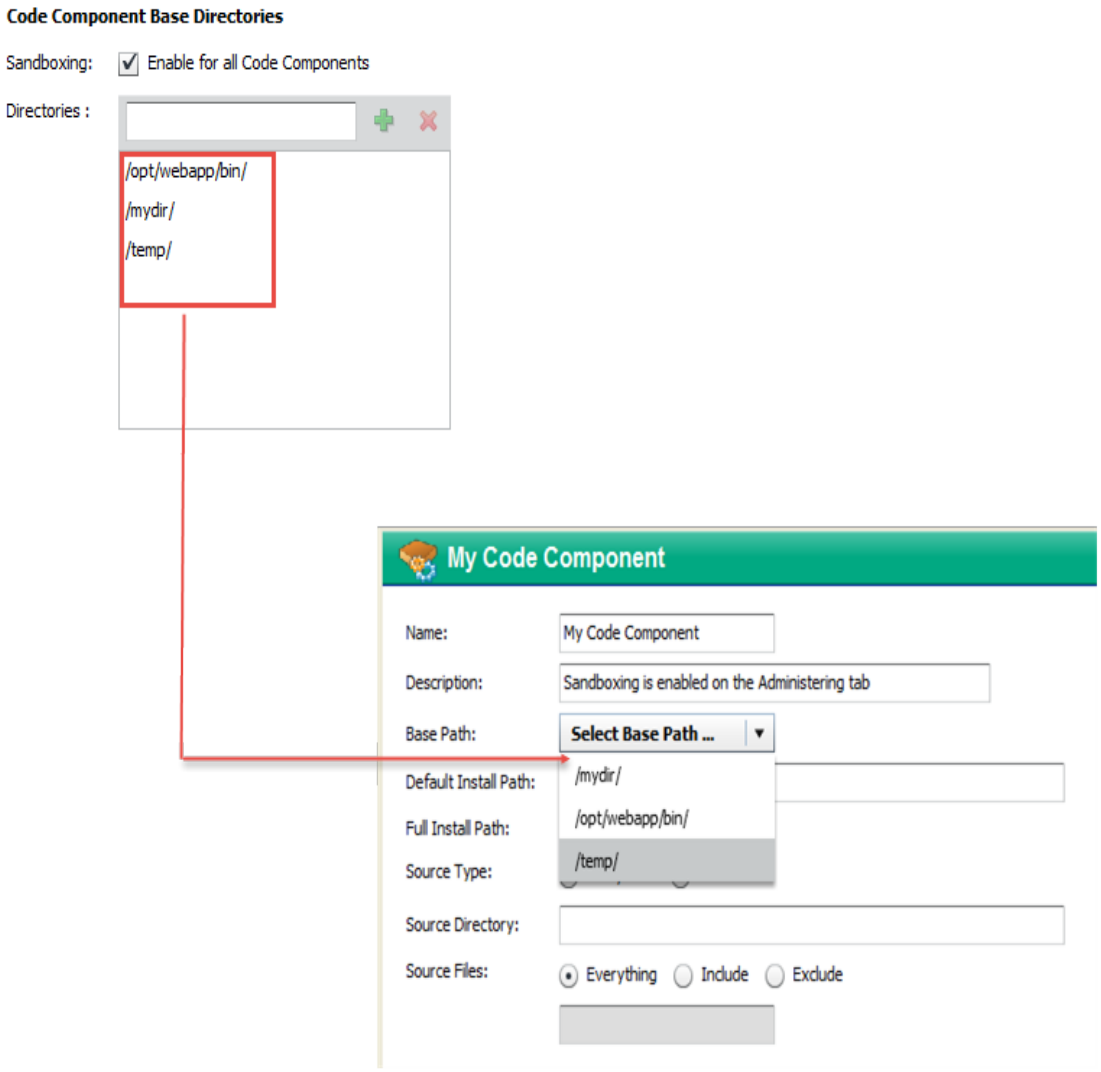
通过这些设置，您可以为代码和配置文件组件部署创建单独的“沙盒”。如果您希望这些组件创建的文件仅安装在受控的文件系统位置中，这将非常有用。

以下示例描述沙盒机制对代码组件的作用。其工作方式与针对配置文件组件时相同。

工作方式

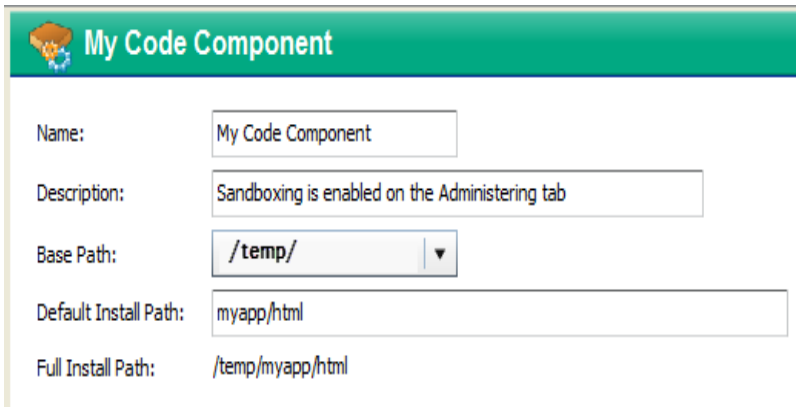
启用沙盒后，您可以指定代码组件和配置文件组件的列表基目录 (沙盒)。然后，这些目录会出现在 Applications 选项卡上组件属性编辑器的 Base Path 下拉列表中。

基目录示例



在代码组件中，所选的基路径与指定的默认安装路径相结合形成完整安装路径，即:文件在目标上放置的位置。

含沙盒的代码组件

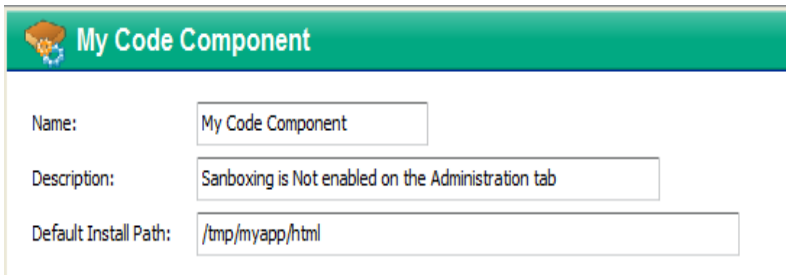


启用沙盒后，应用程序设计师必须在创建代码或配置文件组件时指定基路径。对于包含尚未指定基路径的代码或配置文件组件的发行版 (启用沙盒后)，不能创建版本。

如果删除现有发行版中的组件所使用的基目录，除非在组件中指定新的基路径，否则无法创建该发行版的新版本。但是，仍可部署现有版本。

如果未启用沙盒，则 Base Path 和 Full Install Path 字段不会显示在代码组件属性编辑器中。

不含沙盒的代码组件



如何配置

您可以为代码组件和/或配置文件组件配置沙盒。

要配置沙盒，请执行以下操作:

1. 在 Administration 屏幕的左窗格中，选择 **Application Settings**。
2. 在右窗格中，选择以下一个或两个选项:
   - **Enable for all Code Components**
   - **Enable for all Configuration File Components**
3. 对于要沙盒化的每个组件类型，执行以下步骤:
   a. 在“Directories”文本框中，键入基路径目录。仅允许绝对路径，而不允许相对路径 (例如，`../webapps`)。

b. 单击 **Add directory choice**。

c. 对于要添加的每个基目录，重复步骤 (a) 和 (b)。

4. 单击 **Save**，保存所做的更改。

要删除现有基路径目录，请执行以下操作:

1. 在 Directories 框中，选择现有目录。
2. 单击 **Remove directory choice**。
3. 单击 **Save**，保存所做的更改。

仅当启用沙盒时才能修改 Directories 列表。

全局参数

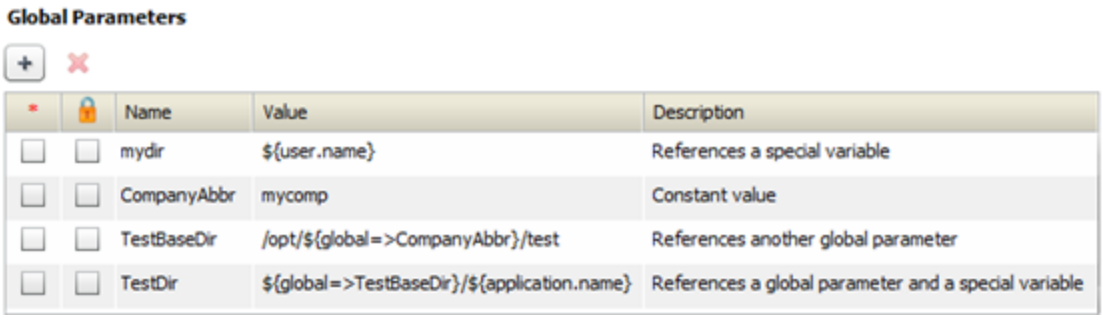
全局参数适用于所有应用程序。指定组件参数的默认值或发行版参数的值时，有权创建应用程序或编辑特定应用程序的 SA 用户可以引用全局参数 (请参见参数和特殊变量 (第 474 页))。

部署时，通过在参数编辑器中执行类似的步骤，具有应用程序的“部署”权限和相关环境的“写入”权限的 SA 用户可以使用全局参数覆盖默认参数值 (请参见自定义部署参数 (第 514 页))。

要定义或修改全局参数，请执行以下操作:

1. 在 Application Settings 页面中，向下滚动到 Global Parameters 表。
2. 要添加新的全局参数，请执行以下步骤:
   a. 单击 **Add New Global Parameter**。
   b. 输入新全局参数的名称。该名称在全局参数中必须唯一。该名称不能包含以下任何字符:@、\、{或 }。
   c. 单击 **OK**。
3. 单击此参数的 Value 列中的任意位置。
4. 指定参数的值 (有关详细说明，请参见指定参数的值 (第 482 页))。

   您可以在值指定中引用特殊变量和其他全局参数。例如：



5. 可选：要对参数值进行加密，请选中 (已加密)列中的框。

   请注意，在此上下文中，* (必需)列不相关。
6. 单击工具栏上的 **Save**，保存所做的更改。

您也可以删除目前未由任何其他参数引用的任意全局参数。

要删除全局参数，请执行以下操作:

1. 在 Global Parameters 表中，选择要删除的参数。
2. 单击 **Delete Global Parameter**。此时将显示确认对话框。
3. 单击 **Yes** 确认删除。
4. 单击工具栏上的 **Save**，保存所做的更改。

导入和导出数据

本主题包含有关如何在 Server Automation (SA) 中导入和导出应用程序部署数据的说明。它包含以下主题:

- 工作 (第 581 页)
- 特别注意事项 (第 582 页)

工作

您可以将应用程序部署数据导出到 XML 文件，或从 XML 文件导入数据。当您需要将应用程序部署数据从一个 SA 核心复制或移动到另一个核心、备份数据以供存档或使用脚本创

建应用程序部署场景时，这一点非常有用。

您可以导入和导出以下项:

- 生命周期
- 环境
- 应用程序
- 应用程序组
- 层和层组
- 发行版
- 组件
- 脚本
- 源类型

您可以预览潜在导入的结果，看看将会创建哪些内容，并确定是否会与现有的应用程序部署数据发生任何冲突。

您可以导入或导出整个应用程序部署数据库，也可以导入或导出特定的应用程序。导入或导出特定的应用程序时，与该应用程序关联的所有项 - 发行版、生命周期、环境、源类型、组件 (如果适用，包括回滚和取消部署脚本) 和层包括在内。

特别注意事项

不会导出以下项:

- 发行版本
- 部署作业
- 目标或目标组
- 环境部署前或部署后流
- 应用程序设置 (在 Administration 屏幕上)
- 应用程序权限
- 环境权限
- SA 库对象 (软件策略、应用程序配置和程序包)

- HPE OO 流内容或定义
- 也不能导入这些项。

由于不能导入或导出发行版本，因此无法导入或导出增量发布信息。

对于导入操作，如果执行导入的 SA 核心中不存在 SA 库项目，则该项目将在其所属的对象中不可见。例如，如果层有一个策略且该 SA 核心中不存在该策略，则该策略将为空。

您可以指定在导入操作期间遇到冲突时的操作:跳过或覆盖该项目。默认行为是跳过该项目并继续导入。但是，如果指定 `--importConflict overwrite`，则会根据 XML 导入文件的内容更新冲突项。

命令行工具

应用程序部署提供了一个命令行工具，帮助您导入和导出应用程序部署数据。其语法如下:

`/opt/opsware/da/bin/admtool.sh -opType [--appListTypeappList|appListFile] --fileTypefilePath [--importConflict overwrite|skip]`

您可以使用此工具导入或导出所有或部分应用程序。

admtool.sh 选项

选项	描述
opType	指定操作:`import`、`export` 或 `preview`。 您可以通过指定 `-i`、`-e` 或 `-p` 缩写此选项。
appListType	可选：指示将导入或导出特定应用程序以及如何指定应用程序名称: `importAppList` - 在命令行上指定的应用程序名称 `exportAppList` - 在命令行上指定的应用程序名称 `importAppListFile` - 在 CSV 文件中指定的应用程序名称 `exportAppListFile` - 在 CSV 文件中指定的应用程序名称 如果不指定此选项，则将导入或导出所有应用程序。
appList	将导入或导出的应用程序的列表。应用程序名称必须以分号分隔: `app1;app2;app3;...appN` 仅与 `importAppList` 和 `exportAppList` 选项一起使用。

admtool.sh 选项(续)

选项	描述
appListFile	包含要导入或导出的应用程序列表的 CSV 文件。应用程序名称必须以分号分隔: app1;app2;app3;...appN 仅与 importAppListFile 和 exportAppListFile 选项一起使用。
fileType	为导入或预览操作指定 importFile。 为导出操作指定 exportFile。
filePath	要导入到、从中导出或预览的 XML 文件的完整路径和名称。 如果不指定导出操作的文件名，将使用默认导出文件。可以在 /etc/opt/opsware/da/da.conf 文件中指定默认导出文件，如下所示: exportFile=/var/tmp/my_export_default_file 如果在 da.conf 文件中未指定 exportFile，且在命令行上未指定导出文件，则导出的数据将写入以下文件中:/opt/opsware/da/bin/ADMExport.xml
importConflict	可选：指示在导入操作期间发生冲突时采取的操作:skip 或 overwrite。如果应用程序部署数据库中已经存在 XML 导入文件中的任何项，则会发生冲突。 skip 将使发生冲突的任何项在该数据库中保持不变。这是默认行为。 overwrite 将根据导入文件的内容更新发生冲突的任何项 (请参见覆盖冲突解决 (第 586 页))。 您可以通过指定 -c overwrite 或 -c skip 缩写此选项。

语法示例

下表显示了应用程序部署管理器》支持的各种导入和导出场景的正确语法。

预览语法示例

预览场景	示例
完全导入预览 (指定的 XML 文件中的所有应用程序)	/opt/opsware/da/bin/admtool.sh -p --importFile myImportData.xml
完全导入预览并在冲突时覆盖	/opt/opsware/da/bin/admtool.sh -p --importFile myImportData.xml --importConflict overwrite
部分导入预览特定应用程序	/opt/opsware/da/bin/admtool.sh -p --importAppList

预览语法示例(续)

预览场景	示例
(在命令行上列出)	"app1;app2;app3" --importFile myImportData.xml
部分导入预览特定应用程序(在命令行上列出)并在冲突时覆盖	/opt/opsware/da/bin/admtool.sh -p --importAppList "app1;app2;app3" --importFile myImportData.xml --importConflict overwrite
部分导入预览特定应用程序(在 CSV 文件中列出)	/opt/opsware/da/bin/admtool.sh -p --importAppListFile app_list.csv --importFile myImportData.xml
部分导入预览特定应用程序(在 CSV 文件中列出)并在冲突时覆盖	/opt/opsware/da/bin/admtool.sh -p --importAppListFile app_list.csv --importFile myImportData.xml --importConflict overwrite

有关其他信息，请参见预览示例 (第 588 页)。

导入语法示例

导入场景	示例
完全导入 (指定的 XML 文件中的所有应用程序)	/opt/opsware/da/bin/admtool.sh -i --importFile myImportData.xml
完全导入并在冲突时覆盖	/opt/opsware/da/bin/admtool.sh -i --importFile myImportData.xml --importConflict overwrite
部分导入特定应用程序 (在命令行上列出)	/opt/opsware/da/bin/admtool.sh -i --importAppList "app1;app2;app3" --importFile myImportData.xml
部分导入特定应用程序 (在命令行上列出)并在冲突时覆盖	/opt/opsware/da/bin/admtool.sh -i --importAppList "app1;app2;app3" --importFile myImportData.xml --importConflict overwrite
部分导入特定应用程序 (在 CSV 文件中列出)	/opt/opsware/da/bin/admtool.sh -i --importAppListFile app_list.csv --importFile myImportData.xml
部分导入特定应用程序 (在 CSV 文件中列出)并在冲突时覆盖	/opt/opsware/da/bin/admtool.sh -i --importAppListFile app_list.csv --importFile myImportData.xml --importConflict overwrite

有关其他信息，请参见导入示例 (第 590 页)。

导出语法示例

导出场景	示例
完全导出所有应用程序部署数据	/opt/opsware/da/bin/admtool.sh -e --exportFile myExportData.xml
部分导出特定应用程序	/opt/opsware/da/bin/admtool.sh -e --exportAppList

导出语法示例(续)

导出场景	示例
(在命令行上列出)	`"app1;app2;app3" --exportFile myExportData.xml`
部分导出特定应用程序(在 CSV 文件中列出)	`/opt/opsware/da/bin/admtool.sh -e --exportAppListFile app_list.csv --exportFile myExportData.xml`

有关其他信息，请参见导出示例 (第 593 页)。

在这些示例中，`app_list.csv` 文件包含以分号 (;) 分隔的应用程序名称列表。例如：`app1;app2;app3;app4`

覆盖冲突解决

如果此 SA 核心上的应用程序部署数据库中已经存在 XML 导入文件中的项，则会发生冲突。默认情况下，导入期间将跳过冲突项。

但如果指定 `--importConflict overwrite`，则会修改冲突项。也可以删除与特定发行版关联的某些项。

下表汇总了指定 `--importConflict overwrite` 时导入操作的结果。

冲突解决场景

项目类型	在导入文件中 不在数据库中	在导入文件中 在数据库中	不在导入文件中 在数据库中
应用程序 应用程序组 与特定应用程序关联的发行版 环境 Lifecycle 层 层组 脚本 源类型	创建项目	根据导入文件的内容更新项目	项目保留在数据库中，不进行修改
与特定发行版关联的层 与特定发行版关联的层相关的组件	创建项目	根据导入文件的内容更新项目	删除项目

有关涉及导入冲突的消息示例，请参见重要消息 (第 587 页)下的有关数据冲突的消息 (第 587 页)。

日志文件

admtool.sh 工具将生成详细的日志文件。文件名取决于执行的操作:

/var/log/opsware/da/importData.log

/var/log/opsware/da/exportData.log

/var/log/opsware/da/previewData.log

重要消息

对于预览操作，以下类型的消息 (在 stdout 中)指示如果执行以下导入，则将创建新项:

Import:'CodeComponentSourceType' 'QA' will be created.

以下类型的消息指示将跳过应用程序，因为它不包含在要导入或导出的应用程序列表中:

跳过：Application 'AppD' due to Application Filter.

对于导入操作，以下类型的消息 (在 stdout 中)指示目标 SA 核心上不存在软件策略、程序包或应用程序配置等库项目:

Not Found:Package'OPSWpytwist2-40.0.0.7.325.zip(SunOS 5.8)'does not exist

以下类型的消息指示目标 SA 核心上的现有应用程序部署数据与 XML 导入文件中的数据之间是否存在冲突:

有关数据冲突的消息

模式	冲突解决	检测到冲突?	示例消息
预览	覆盖	是	Conflict:Release 'App1:Release1' will be updated.
预览	跳过	是	Conflict:Release 'App1:Release1' already exists. will skip.
预览	覆盖或跳过	否	Import:Release 'App1:Release1' will be created.
导入	覆盖	是	Updated Script 'Windows Code Rollback' based on importConflict

有关数据冲突的消息(续)

模式	冲突解决	检测到冲突?	示例消息
			'overwrite' option.
导入	覆盖	是	Deleted Component 'Script 1 (Sample Application:Q3 Release:Tomcat 6.0.20)' based on importConflict 'overwrite' option.
导入	跳过	是	Lifecycle 'Standard' already exists.Will not import based on importConflict 'skip' option.
导入	覆盖或跳过	否	Created Lifecycle 'Complete'.

预览示例

以下简化示例旨在说明完全导入、部分导入、完全导出和部分导出操作的工作方式。XML 文件仅包含应用程序定义。一个典型的文件还将包含有关应用程序组、发行版、生命周期、环境、组件、脚本、层、层组和源类型的信息。

预览 (-p) 操作显示能否成功导入 XML 文件中的信息以及是否存在任何冲突。

在此示例中，导入文件包含四个应用程序:MyAppA、MyAppB、MyAppC 和 MyAppD。MyAppA 应用程序已存在于此 SA 核心上。

以下是此 XML 导入文件 (myImportData.xml) 的内容:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DAData xmlns="http://www.hp.com/DA">
       <ModelVersion>1.3</ModelVersion>
       <ReleaseVersion>09.003.001</ReleaseVersion>
       <SCM_VERSION>0</SCM_VERSION>
       <Applications>
              <Application>
                     <Name>MyAppA</Name>
                     <Description>This application exists</Description>
              </Application>
       <Application>
              <Name>MyAppB</Name>
              <Description>This is a new application - no conflict</Description>
       </Application>
       <Application>
              <Name>MyAppC</Name>
```

```
            <Description>This is a new application - no conflict</Description>
        </Application>
        <Application>
            <Name>MyAppD</Name>
            <Description>This is a new application - no conflict</Description>
        </Application>
        </Applications>
</DAData>
```

示例 1:完全导入预览

以下命令将预览在 myImportData.xml 文件中导入的所有应用程序部署信息:

```
/opt/opsware/da/bin/admtool.sh -p --importFile /tmp/myImportData.xml
```

在执行预览时，stdout 中将显示以下消息:

```
Importer - ========== Begin Previewing with importConflict option 'skip' ...
AbstractBuilder - Conflict: Application 'MyAppA' already exists. Will skip.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Import: Application 'MyAppC' will be created.
AbstractBuilder - Import: Application 'MyAppD' will be created.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Previewing ==========
```

在这种情况下，将不导入 MyAppA，因为它已存在于此 SA 核心上，并且默认冲突解决策略 (跳过) 适用。

要使用覆盖冲突解决策略，请改为指定以下命令:

```
/opt/opsware/da/bin/admtool.sh -p --importFile /tmp/myImportData.xml --
importConflict overwrite
```

在这种情况下，stdout 中将显示以下消息:

```
Importer - ========== Begin Previewing with importConflict option 'overwrite'
AbstractBuilder - Conflict: Application 'MyAppA' will be updated.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Import: Application 'MyAppC' will be created.
AbstractBuilder - Import: Application 'MyAppD' will be created.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Previewing ==========
```

示例 2:部分导入预览

以下命令将预览 MyAppA 和 MyAppB 导入:

```
/opt/opsware/da/bin/admtool.sh -p --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml
```

```
Importer - ============ Begin Previewing with importConflict option 'skip' ...
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Conflict: Application 'MyAppA' already exists. Will skip.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Skip: Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skip: Application 'MyAppD' due to Application Filter.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ============ End Previewing ============
```

在这种情况下，将不导入 MyAppA，因为它已存在于此 SA 核心上，并且默认冲突解决策略 (跳过) 适用。不会导入 MyAppC 和 MyAppD，因为它们不是 `--importAppList` 参数的一部分。

要预览使用覆盖冲突解决策略执行的此导入，请使用以下命令:

```
/opt/opsware/da/bin/admtool.sh -p --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml --importConflict overwrite
```

在这种情况下，`stdout` 中将显示以下消息:

```
Importer - ============ Begin Previewing with importConflict option 'overwrite'
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Conflict: Application 'MyAppA' will be updated.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Skip: Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skip: Application 'MyAppD' due to Application Filter.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ============ End Previewing ============
```

导入示例

我们将继续前面的示例并导入同一 XML 文件 (`myImportData.xml`) 的内容。

示例 1:完全导入

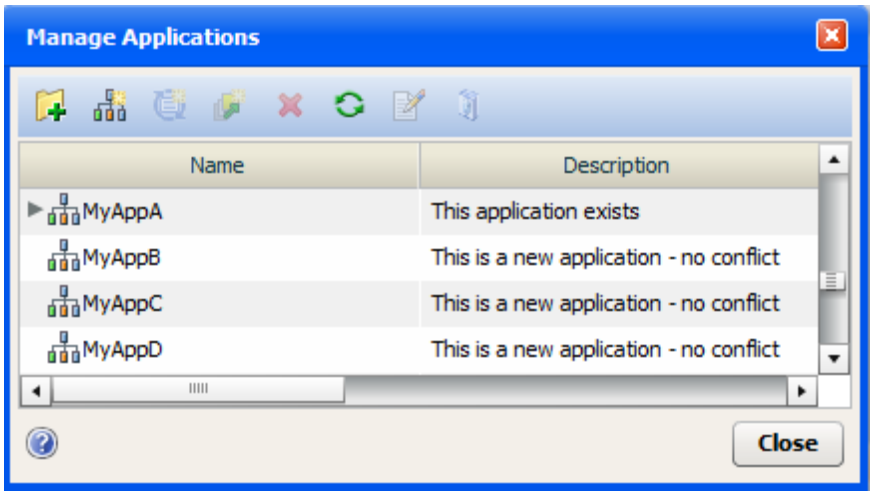
以下命令将尝试使用默认冲突解决策略 (跳过) 导入 `myImportData.xml` 文件中的所有应用程序部署信息:

```
/opt/opsware/da/bin/admtool.sh -i --importFile /tmp/myImportData.xml
```

我们预计将跳过 MyAppA，因为它已存在，并将创建其他应用程序 (MyAppB、MyAppC 和 MyAppD)。以下消息确认了此结果:

```
Importer - ========== Begin Importing with importConflict option 'skip' ...
AbstractBuilder - Application 'MyAppA' already exists. Will not import based
on importConflict 'skip' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Created Application '<top>.MyAppC'.
AbstractBuilder - Created Application '<top>.MyAppD'.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

在导入操作后，Manage Applications 对话框中将列出所有这四个应用程序：



请注意以下几点：

如果您尝试从包含无效 XML 的文件导入数据，则不会执行导入。

如果在导入软件策略、程序包或应用程序配置组件时缺少相关的库项目，将显示以下消息：

```
Not Found:Package'OPSWpytwist2-40.0.0.7.325.zip(SunOS 5.8)'does not exist
```

如果发生这种情况，除非选择其他库项目或将缺失项添加到此核心上的库中，然后选择该项，否则您将无法创建版本。

要使用覆盖冲突解决策略执行此导入，请使用以下命令：

```
/opt/opsware/da/bin/admtool.sh -i --importFile /tmp/myImportData.xml --
importConflict overwrite
```

在这种情况下，stdout 中将显示以下消息：

```
Importer - ========== Begin Importing with importConflict option 'overwrite'
AbstractBuilder - Updated Application '<top>.MyAppA' based on importConflict
'overwrite'
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Created Application '<top>.MyAppC'.
AbstractBuilder - Created Application '<top>.MyAppD'.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

示例 2:部分导入

假设只存在 MyAppA，以下命令仅从 myImportData.xml 文件导入 MyAppB:

```
/opt/opsware/da/bin/admtool.sh -i --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml
```

我们预计将出现以下结果:

系统将跳过 MyAppA，因为它已存在于此 SA 核心上。

将创建 MyAppB。

不会创建 MyAppC 和 MyAppD，因为它们不是 --importAppList 参数的一部分。

以下消息确认了此结果:

```
Importer - ========== Begin Importing with importConflict option 'skip' ... ======
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Application 'MyAppA' already exists. Will not import based on
importConflict 'skip' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Skipped import of Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skipped import of Application 'MyAppD' due to Application Filter.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

要使用覆盖冲突解决策略执行此导入，请使用以下命令:

```
/opt/opsware/da/bin/admtool.sh -i --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml --importConflict overwrite
```

在这种情况下，stdout 中将显示以下消息:

```
Importer - ========== Begin Importing with importConflict option 'overwrite' ...
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Updated Application '<top>.MyAppA' based on importConflict
'overwrite' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Skipped import of Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skipped import of Application 'MyAppD' due to Application Filter.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

导出示例

有两种类型的导出操作。完全导出会将此 SA 核心上存在的所有应用程序部署数据写入指定的 XML 文件。部分导出仅写入与指定应用程序有关的信息。

有关可以导出的项目的列表，请参见工作 (第 581 页)。

有关不能导出的项目的列表，请参见特别注意事项 (第 582 页)。

示例 1:完全导出

以下命令将导出此 SA 核心上的所有应用程序部署数据:

```
/opt/opsware/da/bin/admtool.sh -e --exportFile /tmp/myFullExportData.xml
```

示例 2:部分导出

以下命令将仅导出 MyAppA 和 MyAppB:

```
/opt/opsware/da/bin/admtool.sh -e --exportAppList "MyAppA;MyAppB" --exportFile
/tmp/myPartialExportData.xml
```

假设自上次导入以来未更改 MyAppA 或 MyAppB，myPartialExportData.xml 文件由此产生的 <Applications> 部分将会如下所示:

```
<Applications> portion of the myPartialExportData.xml file would look like this:
       <Applications>
              <Application>
                     <Name>MyAppB</Name>
                     <Description>This is a new application - no conflict</Description>
       </Application>
       <Application>
                     <Name>MyAppA</Name>
                     <Description>This application exists</Description>
              </Application>
</Applications>
```

发送文档反馈

如果您对本文档有任何意见，可以通过电子邮件与文档团队联系。如果在此系统上配置了电子邮件客户端，请单击以上链接，此时将打开一个电子邮件窗口，主题行中为以下信息：

开发人员指南 (Server Automation 10.50) 反馈

只需在电子邮件中添加反馈并单击“发送”即可。

如果没有可用的电子邮件客户端，请将以上信息复制到 Web 邮件客户端的新邮件中，然后将您的反馈发送至 hpe_sa_docs@hpe.com 。

我们感谢您提出宝贵的意见！