思科身份服务引擎 API 参考指南，版本 2.0

2015 年 10 月 15 日

思科系统公司
www.cisco.com

思科在全球设有 200 多个办事处。
有关地址、电话号码和传真号码信息，
可查阅思科网站：
www.cisco.com/go/offices

文本部件号： OL-26134-01

目 录

前言

本前言说明目标、目标受众和组织。*思科身份服务引擎 API 参考指南，版本 2.0* 还描述提供说明的约定并提供信息的其他类型在以下部分的：

思科身份服务引擎概述

思科身份服务引擎 (ISE) 是下一代身份和访问控制策略平台，可帮助企业执行策略规定、加强基础设施安全以及简化服务操作。凭借 Cisco ISE 的独特架构，企业可以通过把身份绑定到各种网络元素（包括访问交换机、无线局域网控制器 (WLC)、虚拟专用网络 (VPN) 网关和数据中心交换机），从网络、用户和设备收集实时背景信息，从而做出前瞻性的管理决策。

Cisco ISE 是思科安全组访问解决方案的关键组件。Cisco ISE 是一个统一的基于策略的访问控制解决方案具有以下

- 将身份验证、授权、会计 (AAA)、状态、分析和访客管理服务到设备
- 通过检查访问网络，包括 802.1X 环境的所有终端设备状态执行终端合规性
- 提供发现、分析、基于策略的布局和监控网络上的终端设备支持
- 在集中式和分布式部署中启用一致的策略，以实现根据实际需要交付服务
- 通过使用安全组标记 (SGT) 和安全组 (SG) 访问控制列表 (ACL) 使用高级实施功能，包括安全组访问 (SGA)
- 支持将多种部署方案从小型办公室扩展到大型企业环境的可扩展性

Cisco ISE 架构支持独立和分布式部署，允许您配置和管理您的从一个集中的门户的网络。有关 Cisco ISE 的功能的详细信息，请参阅《*思科身份服务引擎管理指南*》。

目的

此应用编程接口 (API) 参考指南提供支持的 API 提供的功能的仅短暂高级概述。此 API 参考指南的目的是为开发人员、系统或网络管理员或系统集成了基本的指导原则使用在 Cisco ISE 配置中概述的 API。

REST API 呼叫使用查询确定数据的以下类型的

- 活动会话的数量
- 活动会话的类型
- 活动会话的身份验证状态
- MAC 地址在使用中
- 在使用 NAS 的 IP 地址
- 节点版本和类型
- 节点会话失败的原因

外部 RESTful 服务 API 和相关 API 调用可用于对 Cisco ISE 资源执行 CRUD（创建、读取、更新、删除）操作。外部宁静的服务根据 HTTP 协议和其他方法。

备注 有关 Cisco ISE 网络及其节点和角色、操作概念或用法以及 Cisco ISE 用户界面使用方法的详细信息，请参阅《*思科身份服务引擎观礼指南*》。

受众

此 API 参考指南为管理网络环境中的 Cisco ISE 设备，系统集成可能要利用 API，或第三方合作伙伴与管理或排除 Cisco ISE 配置的责任的有经验的系统管理员使用。为使用此 API 参考指南的一个前提条件，您应该有故障排除和诊断工作有基本的了解和如何发出和解释 API 调用。

文档结构

本指南的结构如下：

- 第 1 部分 - Cisco ISE 监控 REST API
  - 第 1 章“监控的 REST API 简介”
  - 第 2 章“Session Management 查询 API”
  - 第 3 章“故障排除的查询 API”
  - 第 4 章“权限 REST API 更改”
- 第 2 部分 - Cisco ISE 外部宁静的服务 API
  - 第 5 章“ERS API 简介”
  - 第 7 章“外部 RESTful 服务 API 操作”
- 附录 A“Cisco ISE 故障原因报告”

文档约定

本节概述约定使用在本文中。

注意 表示*读者应当小心*。您的某些操作可能会导致设备损坏或数据丢失。

备注 表示*读者需要注意的地方*。注释中包含有用的建议或包含对本手册中所没有的材料的引用。

此 API 参考指南使用以下约定表示指令和信息。

项目	约定
命令、关键字、特殊应选择在过程中的术语和选项	**粗体**
由您提供值和新或重要术语的变量	*斜体*
显示的会话和系统信息、路径和文件名	屏幕字体
您输入的信息	**屏幕粗体**
您输入的变量	*屏幕斜体*
菜单项和按钮名称	**粗**体
表示菜单项按您选择其顺序选择。	**Option > Network Preferences**

产品文档

备注 思科有时会在原始版本之后更新印刷文档和电子文档。因此，您还应在 http://www.cisco.com 上查阅文档以获取任何更新。

表 1 列出适用于 www.cisco.com 上的 Cisco ISE 版本 2.0 的相关产品文档。要在 www.cisco.com 上查找所有产品的最终用户文档，请转至：

http://www.cisco.com/go/techdocs

相关文档

本节提供有关版本特定文档以及平台特定文档的信息。

版本特定文档

表 1 列出适用于 Cisco ISE 版本的产品文档。Cisco ISE 的一般产品信息位于 http://www.cisco.com/go/ise。最终用户文档位于 Cisco.com 上的 http://www.cisco.com/en/US/products/ps11640/tsd_products_support_series_home.html。

表 1　　思科身份服务引擎的产品文档

文档标题	位置
思科身份服务引擎版本说明，版本 2.0	http://www.cisco.com/en/US/products/ps11640/prod_release_notes_list.html
思科身份服务引擎网络组件兼容性，版本 2.0	http://www.cisco.com/en/US/products/ps11640/products_device_support_tables_list.html
思科身份服务引擎管理员指南，版本 2.0	http://www.cisco.com/en/US/products/ps11640/products_user_guide_list.html
思科身份服务引擎硬件安装指南，版本 2.0	http://www.cisco.com/en/US/products/ps11640/prod_installation_guides_list.html
思科身份服务引擎发起人门户用户指南	http://www.cisco.com/en/US/products/ps11640/products_user_guide_list.html
思科身份服务引擎 CLI 参考指南，版本 2.0	http://www.cisco.com/en/US/products/ps11640/prod_command_reference_list.html
思科身份服务引擎 API 参考指南，版本 2.0	http://www.cisco.com/en/US/products/ps11640/prod_command_reference_list.html
思科身份服务引擎故障排除指南，版本 2.0	http://www.cisco.com/en/US/products/ps11640/prod_troubleshooting_guides_list.html
思科身份服务引擎、Cisco 1121 安全访问控制系统、Cisco NAC Appliance、Cisco NAC Guest Server 与 Cisco NAC Profiler 的合规性和安全信息	http://www.cisco.com/en/US/products/ps11640/prod_installation_guides_list.html
思科身份服务引擎设备内文档和中国 RoHS 指针卡	http://www.cisco.com/en/US/products/ps11640/products_documentation_roadmaps_list.html

Platform - Specific 文档

对策略管理业务部门文档的链接位于 http://www.cisco.com 在以下位置

- Cisco ISE
  http://www.cisco.com/en/US/products/ps11640/prod_installation_guides_list.html
- Cisco Secure ACS
  http://www.cisco.com/en/US/products/ps9911/tsd_products_support_series_home.html
- Cisco NAC 设备
  http://www.cisco.com/en/US/products/ps6128/tsd_products_support_series_home.html
- Cisco NAC 分析器
  http://www.cisco.com/en/US/products/ps8464/tsd_products_support_series_home.html
- Cisco NAC 访客服务器
  http://www.cisco.com/en/US/products/ps10160/tsd_products_support_series_home.html

获取文档和提交服务请求

关于如何获取文档、提交服务请求和收集其他信息的信息，请参阅每月的*思科产品文档更新*，其中还含有所有最新及修订的思科技术文档，此文档位于：

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

通过 Really Simple Syndication (RSS) 源的方式订阅*思科产品文档更新*，相关内容将通过阅读器应用直接发送到您的桌面。 RSS 源是一项免费服务，思科目前支持 RSS 2.0 版本。

第 1 部分

Cisco ISE 监控 REST API

监控的 REST API 简介

通过在您的网络，以监控节点监控的 REST API 允许您收集会话和节点特定信息。当您访问所需的节点并完成操作需要收集信息时，会话定义为持续时间在之间。

Cisco ISE，版本 1.4 支持以下监控的 REST API 类别：

- 会话管理
- 故障排除
- 授权更改 (CoA)

备注 只使用支持的类别收集有关监控角色被监控的终端的信息。监控是节点类型在 Cisco ISE，版本 1.4，配置可以执行哪三个支持的人之一。对于本指南其他，“监控节点”将用于描述 Cisco ISE 节点的监控作用。

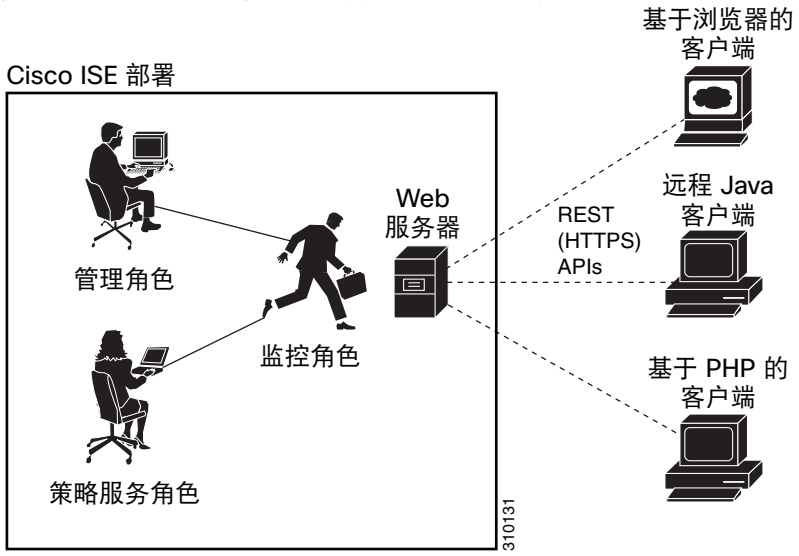
所有尝试使用这些类别收集有关 Cisco ISE 设备的策略服务角色的信息产生错误。有关 Cisco ISE 节点和人员的详细信息，请参阅*思科身份服务引擎管理员指南，版本 2.0*。

监控 REST API 呼叫在网络允许您隔离，监控和累计在单个终端上存储的重要实时，基于会话的信息。您可以通过监控节点的此信息。

实时，您收集可帮助了解 Cisco ISE 操作并帮助诊断情况或问题的基于会话的信息。它还可用于故障排除可影响监控操作的错误状态或活动或行为。如图 1-1 所示，监控的 REST API 呼叫用于访问监控节点和检索在 Cisco ISE 配置终端已存储的重要基于会话的信息。

备注 Cisco ISE 的，版本 1.2 监控的 REST API 已弃用，并更改了所有服务呼叫的 URL 路径。请为 Cisco ISE，版本 1.4 使用监控的 REST API。

图 1-1　　监控在分布式部署的 REST API 呼叫



验证监控节点

准备工作

才能成功传输之前 API 在监控节点，您不需要验证要监控的节点是有效的。

备注　使用有效的凭证，希望能够使用公共监控的 REST API，您必须首先是否与 Cisco ISE。

步骤 1　输入有效的登录凭证（用户名和密码）在 Cisco ISE 登录，然后单击**登录**。

Cisco ISE 仪表板和用户界面显示。

步骤 2　选择 **Authorization > System > Deployment**。

系统将显示 Deployment Nodes 页面，其中列出所部署的所有已配置的节点。

步骤 3　在部署节点的列呼叫的角色，验证您要监控的目标节点的角色列为监控节点。

支持的 API 呼叫

下表介绍 API 呼叫不同类型并提供 API 呼叫格式的示例

- 表 1-1，第 1-3 页 - 定义 API 呼叫会话管理。
- 表 1-2，第 1-5 页 - 定义 API 呼叫请求进行故障排除。
- 表 1-3，第 1-7 页 - 定义 CoA API 呼叫。

如果您要使用通用编程接口是否与 Cisco ISE 支持的显示器的 REST API，您需要先创建桥接 Cisco ISE 和特定工具您使用的基于 REST 的客户端。然后使用此 REST 客户端是否与 Cisco ISE 监控 REST API，安排和提交 API 请求到监控节点，然后 unmarshal API 响应和转发到指定的工具。

表 1-1　Cisco ISE Session Management API 呼叫

API 呼叫类别	说明和示例
会话计数器	
ActiveCount	列出了“活动会话的数量”。 https:// <ISEhost>/admin/API/mnt/Session/ActiveCount
PostureCount	列出 Postured 终端的数量。 https:// <ISEhost>/admin/API/mnt/Session/PostureCount **备注**　状态处于控制中帮助该状态的服务（或状态）所有终端连接到 Cisco ISE 网络。Cisco ISE 为检查设备的状态合规性使用 NAC 代理。
ProfilerCount	列出了活动的分析器服务会话数量。 https:// <ISEhost>/admin/API/mnt/Session/ProfilerCount **备注**　分析器是在确定，找到和定位为所有相连的终端功能在 Cisco ISE 网络的服务。
会话列表 **备注**　会话列表包括 MAC 地址、网络接入设备 (NAD) IP 地址，用户名和会话 ID 信息与会话相关联。	
ActiveList	列出所有活动会话。 https:// <ISEhost>/admin/API/mnt/Session/ActiveList **备注**　在此 Cisco ISE 版本中，可以显示且经过身份验证的最大活动终端会话数为 250000。

表 1-1　Cisco ISE Session Management API 呼叫（续）

API 呼叫类别	说明和示例
AuthList	列出所有当前活动的已验证的会话。 https:// <ISEhost>/admin/API/mnt/Session/AuthList/<parameteroptions> 您可以指定将返回不同值的以下参数选项卡 • 空 / 空列出所有激活已验证会话。 • 空 /Endtime 列出所有激活已验证会话在指定的结束时间之后。 • 开始 / 空列出所有活动在指定的开始时间之前的已验证的会话。 • 开始 /Endtime 列出所有活动在指定的开始时间和结束时间之间的已验证的会话。 输入日期和时间的开始时间和结束时间使用以下格式 YYYY - MM - DD hh：mm 格式 mm ： ss.s 其中： • YYYY 四年数字 • MM 两个数字个月（01=January，等等） • DD 两 Num 天（ 01 - 31 ） • HH 两位小时（ 00 - 23 ）（客户经理和 p.m、不允许） • 毫米两位分钟（ 00 - 59 ） • 接下来 hh：mm：ss 两位（ 00 - 59 ） • 表示十进制一转眼工夫的 s - one 或更多数字 **备注**　每个 Cisco ISE 节点配置时区。建议使用的时区为 UTC。 用于采样从与空 / 空选项的 AuthList API 调用返回的数据，第 2-8 页显示所有四个参数选项的示例，请参阅。
会话属性 **备注**　这是包含指定的搜索属性的最新的会话的基于时间戳的搜索。	
MAC 地址	搜索数据库包含指定的 MAC 地址的最新的会话。 https:// <ISEhost>/admin/API/mnt/Session/MACAddress/<macaddress> **备注**　XX：XX：XX：XX：XX：XX 是 MAC 地址格式不区分大小写（例如，答案：0B：020c：85：cf：66：0D：058d：09：0F ）。 **备注**　MAC 地址作为唯一的唯一密钥对找到您要监控的正确的会话。使用 ActiveList API 呼叫列出所有活动会话和其 MAC 地址，您可以根据自己的 MAC 地址搜索。
UserName	搜索数据库包含指定的用户名的最新的会话。 https:// <ISEhost>/admin/API/mnt/Session/UserName/<username> **备注**　用户名必须符合用于网络用户名相同的 Cisco ISE 密码策略。监控 REST API 的唯一无效字符为反斜线（\ ）字符。有关详细信息，请参阅“用户密码策略”在*思科身份服务引擎用户指南，版本 1.1*。

表 1-1　Cisco ISE Session Management API 呼叫（续）

API 呼叫类别	说明和示例
IPAddress（IP 地址）	搜索数据库中包含指定 NAS IP 地址（IPv4 或 IPv6 地址）的最新会话。 https:// <ISEhost>/admin/API/mnt/Session/IPAddress/<nasipaddress> **备注**　xxx.xxx.xxx.xxx 是 NAS IP 地址格式（例如，10.10.10.10） 或 https://<ISEhost>/admin/API/mnt/Session/IPAddress/<nasipv6address> **备注**　xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx 是 NAS IPv6 地址格式（例如，2001:cdba:0:0:0:0:3247:9651）
Audit Session ID	搜索数据库包含指定的审计会话 ID 的最新的会话。 https:// <ISEhost>/admin/API/mnt/Session/Active/SessionID/<audit - session - id>/0 **备注**　使用 ActiveList API 呼叫列出所有活动会话及其审计会话 ID，您可以根据您的会话 ID 搜索。或者，您可以获得实时会话页面的审计会话 ID 在管理员门户。

有关 Cisco ISE API 呼叫的特定详细信息请求会话管理，请参阅第 2 章“Session Management 查询 API”。

表 1-2　排除API呼叫 - 故障排除的 Cisco ISE

API 调用	说明和示例
版本	列出节点版本和类型。 https:// <ISEhost>/admin/API/mnt/Version 节点类型可以是下列值（ 0 - 3 ）中的任何一个。 0 - STAND_ALONE_MNT_NODE 1 - ACTIVE_MNT_NODE 2 - STAND_BY_MNT_NODE 3 - NOT_AN_MNT_NODE **备注**　STAND_ALONE_MNT_NODE 意味着它是在任何已分配的配置无法正常运行的监控节点。 ACTIVE_MNT_NODE 意味着它是一个主要的关系的主节点在分布式部署。 STAND_BY_MNT_NODE 意味着它是一个主要的第二个的辅助节点在分布式部署。 NOT_AN_MNT_NODE 意味着它不是监控节点。有关*支持的 ESS 节点和人员的*详细信息，请参阅思科身份服务引擎用户指南，版本 1.1。

表 1-2　排除API呼叫 - 故障排除的 Cisco ISE（续）

API 调用	说明和示例
FailureReasons	列出故障的原因。 https:// <ISEhost>/admin/API/mnt/FailureReasons 每个故障原因显示错误代码（failureReason ID ），简短说明（代码），故障原因（原因）和一个可能的响应（分辨率），如以下示例所示： <failureReason id= “ 100009 ” > <code> 100009 WEBAUTH_FAIL <cause> 可以或可能不指示违规。 <resolution> 根据您组织的策略请查看并解决此问题。 **备注**　FailureReasons API 呼叫仅一次将调用收集从监控节点的信息。您应存储所有返回的故障原因拖动到文件系统或数据库。这些 API 调用返回的目录供参考使用。如果您在身份验证过程中遇到任何问题，您应当比较故障原因代码列表故障原因在验证响应提供的您在您的文件系统或数据库存储了。 有关 Cisco ISE 故障原因的完整列表，请参阅附录 A “Cisco ISE 故障原因报告”。
AuthStatus	列出所有会话的身份验证状态。 https:// <ISEhost>/admin/API/mnt/AuthStatus/MACAddress/<macaddress>/<numberofseconds>/<numberofrecordspermacaddress>/All **备注**　秒参数 <numberofseconds> 是用户可配置的，范围为 0 秒至 432000 秒（5 天）。
获得 ISN 客户状态	
AcctStatus	列出所有会话的客户状态在给定时间段内。 https:// <ISEhost>/admin/API/mnt/AcctStatusTT/MACAddress/<macaddress>/<numberof seconds> **备注**　秒参数 <numberofseconds> 与该范围是用户可配置的，是从 0 - 432000 秒（5 天）。

有关 Cisco ISE API 呼叫的特定详细信息请求进行故障排除，请参阅第 2 章“Session Management 查询 API”。

表 1-3　　权限API呼叫/ Cisco ISE 更改

API 调用	说明和示例
Reauth	发送一个默认端口命令和类型。 https:// <ISEhost>/admin/API/mnt/CoA/Reauth/<serverhostname>/<macaddress>/<reauthtype>/<nasipaddress>/<destinationipaddress> 其中 <*ISEhost*> 表示 ESS 主机的 IP 地址， <*serverhostname*> 表示 ESS 服务器的名称， <*nasipaddress*> 表示 NAS 的确定的 IP 地址，而且， <*destinationipaddress*> 表示目标的 IP 地址。 Reauth 类型可以是下列值（ 0 - 2 ）中的任何一个。 0 - REAUTH_TYPE_DEFAULT 1 - REAUTH_TYPE_LAST 2 - REAUTH_TYPE_RERUN **备注**　如果您不知道 NAS IP 地址，您可以输入所需的值向上传送到该点，并 API 在其搜索查询将使用这些值。但是，您必须知道 MAC 地址执行此 API 调用，但是，您可以从 NAS IP 地址开始将其它参数为空。如果提供了 NAS IP 地址还提供目标 IP 地址是必要的。 此 API 呼叫在一台显示器上 ESS 节点只能执行，提交请求远程执行 CoA。管理 ESS 节点不是包含或所需的执行这些 CoA API 呼叫。
会话断开连接	
Disconnect	发送一个会话断开命令和端口选项类型。 https:// <ISEhost>/admin/API/mnt/CoA/Disconnect/<serverhostname>/<macaddress>/<disconnecttype>/<nasipaddress>/<destinationipaddress> 端口选项类型可以是下列值（ 0 - 2 ）中的任何一个。 0 - DYNAMIC_AUTHZ_PORT_DEFAULT 1 - DYNAMIC_AUTHZ_PORT_BOUNCE 2 - DYNAMIC_AUTHZ_PORT_SHUTDOWN **备注**　如果您不知道 NAS IP 地址，请输入所需的值向上传送到该点，并 API 在其搜索查询将使用这些值。但是，您必须知道 MAC 地址执行此 API 调用，但是，您可以将其他参数为空。

有关权限 API 呼叫的详情 Cisco ISE 更改，请参阅第 4 章“权限 REST API 更改”。

HTTP 将 API 呼叫

类似于 AuthStatus API 呼叫表 1-2，可让客户端检索客户状态 API 呼叫的 HTTP Put 版本。监控的 REST API 支持 Put HTTP 和 HTTP GET 呼叫，与提供 HTTP GET 呼叫的此指南的示例。Put HTTP 处理要求参数输入的呼叫的需求。以下架构文件示例是一个需要客户状态：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="acctRequest" type="mnTRESTAcctRequest"/>

  <xs:"mnTRESTAcctRequest" complexType 的 name= >
    <xs:complexContent>
      <xs:"mnTRESTRequest" 分机的 base= >
        <xs:sequence>
          <xs: 元素 name=" 持续时间 " type="xs: 字符串 " minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:"true" complexType name="mnTRESTRequest" 的 abstract= >
    <xs:sequence>
      <xs: 元素 name="valueList">
        <xs:complexType>
          <xs:sequence>
            <xs: 元素 name=" 值 " type="xs: 字符串 " maxOccurs=" 无边际的 "/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs: 元素 name="searchCriteria" type=" xs: 字符串 "/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Session Management 查询 API

本章介绍会话管理 API 调用。这些 API 调用在 Cisco ISE 部署的思科监控 ESS 节点中，为检索重要会话相关的信息提供方法。

会话计数器 API 调用

以下会话计数器 API 调用可让您快速收集当前计数有关目标监控您的 Cisco ISE 配置的 Cisco 的会话相关的信息 ESS 节点：

- 活动会话（ActiveCount ）- 活动会话已验证在网络中的一个。
- 状况评估的会话（PostureCount）- 状况评估的状态断言，当状态结束时（兼容 / 不合规）。状态是可选的，例如，IP 电话 / 打印机不会转到状况评估的状态。状况评估的状态是一个短期的临时状态，因为在状况评估后，它将移至开头的状态，在设置时认为的开始时间。
- 已分析的会话（ProfilerCount ）

如果终端陷在任何一个阶段，这些不同状态被视为故障排除。

活动会话计数器

您可以使用 ActiveCount API 调用来检索所有当前活动的会话的计数。本节提供架构文件输出示例，计数的所有活动会话方法通过调用 ActiveCount API 呼叫，并发出在此 API 调用后返回的活动会话数据的示例。

ActiveCount API 输出方案

此示例架构文件是 ActiveCount API 调用的输出请求检索活动会话的计数 ESS 节点的目标监控作用的：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionCount" type="activeCount"/>
  <xs:complexType name="activeCount">
    <xs:sequence>
      <xs:element name="count" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

调用 ActiveCount API

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 ActiveCount API 调用通过替换“/admin/ ”组件使用的 API 调用组件（/admin/API/mnt/ <specific - api - call>）：

```
https://acme123/admin/API/mnt /Session/ActiveCount
```

注　因为这些呼叫调用大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 调用约定表示该目标监视器 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 调用。

相关主题

- 验证监控节点，第 1-2 页

采样从 ActiveCount API 调用返回的数据

以下示例说明返回的数据（活动会话数），当您将在目标监视器 ESS 节点时的 Cisco 的 ActiveCount API 调用：

```
此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。
 -
<sessionCount>
<count>5</count>
</sessionCount>
```

状态会话计数器

您可以调用 PostureCount API 来检索当前所有当前活动的会话状态的计数。

PostureCount API 输出方案

此示例架构文件是 PostureCount API 调用的输出请求检索当前有效的会话状态的计数目标的监控 ESS 节点的 Cisco：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionCount" type="postureCount"/>

  <xs:complexType name="postureCount">
    <xs:sequence>
      <xs:element name="count" type="xs:int"/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

调用 PostureCount API

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 PostureCount API 调用通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/Session/ <specific - api - call> ）：

```
https://acme123/admin/API/mnt/Session/PostureCount
```

注　因为这些调用区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“mnt”在 API 调用约定表示该目标监视器 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 调用。

相关主题

- 验证监控节点，第 1-2 页

采样从 PostureCount API 调用返回的数据

以下示例说明返回的数据（当前有效的状态会话数），当您将在目标监视器 ESS 节点时的 Cisco 的一 PostureCount API 调用：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
-
<sessionCount>
<count>3</count>
</sessionCount>
```

分析器的会话计数器

您可以调用 ProfilerCount API 来检索所有当前活动的分析器会话的计数。

ProfilerCount API 输出方案

此示例架构文件是 ProfilerCount API 呼叫的输出请求检索当前有效的分析器会话的计数目标的监控 ESS 节点的 Cisco：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs: 元素 name="sessionCount" type="profilerCount"/>

  <xs:complexType name="profilerCount">
    <xs:sequence>
      <xs: 元素 name=" 计数 " type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

调用 ProfilerCount API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 ProfilerCount API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/Session/ <specific - api - call> ）：

```
https://acme123/admin/API/mnt/Session/ProfilerCount
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 ProfilerCount API 调用返回的数据

以下示例说明返回的数据（有效的分析器会话数），当您将在目标监视器 ESS 节点时的 Cisco 的一 ProfilerCount API 呼叫：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<sessionCount>
<count>1</count>
</sessionCount>
```

简单的会话列表 API 呼叫

以下简单的会话列表 API 呼叫可让您快速收集会话相关的信息（例如 MAC 地址、网络接入设备 (NAD) IP 地址，账号和会话 ID 与目标的监控您的 Cisco ISE 配置的 Cisco 一当前活动会话相关 ESS 节点：

- 活动会话列表（ActiveList ）
- 已验证会话列表（AuthList ）

活动会话列表

您可以使用 ActiveList API 呼叫列出所有当前活动的会话。

备注　激活最大数量验证可以显示是限制为 100,000 的终端会话。

ActiveList API 输出方案

此示例架构文件是 ActiveList API 呼叫的输出请求检索当前活动会话（和会话相关的信息的）列表有关该目标监视器 ESS 节点的 Cisco：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:方案version="1.0 "xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:元素name="activeSessionList" type="simpleActiveSessionList"/>

<xs:complexType name="simpleActiveSessionList">
    <xs:sequence>
      <xs:元素name="activeSession" type="simpleActiveSession" minOccurs="0" maxOccurs="无边
际的"/>
    </xs:sequence>
    <xs:属性name="noOfActiveSession" type="xs:int" use="必需"/>
  </xs:complexType>

  <xs:complexType name="simpleActiveSession">
    <xs:sequence>
      <xs:元素name="user_name" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="calling_station_id" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="nas_ip_address" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="acct_session_id" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="audit_session_id" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="服务器" type="xs:字符串" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

<xs:element name="nas_ipv6_address" type="xs:string"/>
  <xs:complexType name="framed_ipv6_address_list">
    <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
<xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>
</xs:schema>
```

调用 ActiveList API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址>/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 ActiveList API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/Session/ <specific - api - call> ）：

```
https://acme123/admin/API/mnt/Session/ActiveList
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 ActiveList API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的一 ActiveList API 调用以下示例说明从活动会话列表返回的会话相关的数据

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<activeSessionList noOfActiveSession="5">
 -
<activeSession>
<calling_station_id>00:0C:0329:FA:ef:0A</calling_station_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<calling_station_id>70:5A:B6:68:F7:CC</calling_station_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>tom_wolfe</user_name>
<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<acct_session_id>00000032</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>graham_hancock</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3257:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3257:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:9652</ipv6_address>
</framed_ipv6_address>
<acct_session_id>0000002C</acct_session_id>
<audit_session_id>0ACB6BA10000002A165FD0C8</audit_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>ipepvpnuser</user_name>
```

```
<calling_station_id> 172.23.130.89 </calling_station_id>
<nas_ip_address> 10.203.107.45 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<acct_session_id>A2000070</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
</activeSessionList>
```

已验证会话列表

您可以使用 AuthList API 呼叫检索所有当前活动的已验证的会话列表。

备注　激活最大数量验证可以显示是限制为 100,000 的终端会话。

AuthList API 输出方案

此示例架构文件是 AuthList API 呼叫的输出请求检索所有当前活动的已验证的会话列表在指定的时间段内（或使用“空 / 空”参数的未指定时间）在该目标监视器 ESS 节点的 Cisco：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:方案version="1.0 "xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:元素name="activeSessionList" type="simpleActiveSessionList"/>

<xs:complexType name="simpleActiveSessionList">
    <xs:sequence>
      <xs:元素name="activeSession"type="simpleActiveSession"minOccurs="0"maxOccurs="无边际的
"/>
    </xs:sequence>
    <xs:属性name="noOfActiveSession" type="xs:int" use="必需"/>
  </xs:complexType>

  <xs:complexType name="simpleActiveSession">
    <xs:sequence>
      <xs:元素name="user_name" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="calling_station_id" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="nas_ip_address" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="acct_session_id" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="audit_session_id" type="xs:字符串" minOccurs="0"/>
      <xs:元素name="服务器" type="xs:字符串" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="nas_ipv6_address" type="xs:string"/>
  <xs:complexType name="framed_ipv6_address_list">
    <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>

</xs:schema>
```

调用 AuthList API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 AuthList API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/Session/ <specific - api - call> ）：

注　第一下列两个示例使用已定义的开始时间和空参数，显示当前的活动会话列表在指定的启动时间后验证。第二个示例使用显示所有当前活动的已验证的会话列表为空 / 参数。参阅采样从与空 / 空选项的 AuthList API 调用返回的数据，第 2-8 页，显示四个参数设置类型示例此 API 呼叫的。

```
https://acme123/admin/API/mnt/Session/AuthList/2010-12-14 2013-04:15/null

https://acme123/admin/API/mnt/Session/AuthList/null/null
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从与空 / 空选项的 AuthList API 调用返回的数据

以下示例说明使用空 / 空选项时，返回当前活动的已验证的会话的列表，在调用 AuthList API 呼叫：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<activeSessionList noOfActiveSession="3">
 -
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address> 10.203.107.10 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3257:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3257:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:9652</ipv6_address>
</framed_ipv6_address>
<audit_session_id>0acb6b0c000000174D07F487</audit_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
```

```
<user_name>tom_wolfe</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>graham_hancock</user_name>
<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
</activeSessionList>
```

从与 endtime/ 空选项的 AuthList API 调用返回的示例数据

以下示例说明使用 endtime/ 空选项时，返回当前活动的已验证的会话的列表，在调用 AuthList API 呼叫：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<activeSessionList noOfActiveSession="3">
 -
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address> 10.203.107.10 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3257:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3257:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:9652</ipv6_address>
</framed_ipv6_address>
<audit_session_id>0acb6b0c0000001F4D08085A</audit_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>hunter_thompson</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
```

```
<acct_session_id>00000035</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>bob_ludlum</user_name>
<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
</activeSessionList>
```

从与空 /starttime 选项的 AuthList API 调用返回的示例数据

以下示例说明使用空 /starttime 选项时，返回当前活动的已验证的会话的列表，在调用 AuthList API 呼叫：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<activeSessionList noOfActiveSession="3">
 -
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address> 10.203.107.10 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3257:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3257:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:9652</ipv6_address>
</framed_ipv6_address>
<audit_session_id>0acb6b0c0000001F4D08085A</audit_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>bob_ludlum</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>tom_wolfe</user_name>
<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
```

```
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
</activeSessionList>
```

从与 statttime/endtime 选项的 AuthList API 调用返回的示例数据

以下示例说明使用开始 /endtime 选项时，返回当前活动的已验证的会话的列表，在调用 AuthList API 呼叫：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<activeSessionList noOfActiveSession="3">
 -
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address> 10.203.107.10 </nas_ip_address>
<audit_session_id>0acb6b0c0000001F4D08085A</audit_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>graham_hancock</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
 -
<activeSession>
<user_name>hunter_thompson</user_name>
<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH - R 6 - 1  - PDP2</server>
</activeSession>
</activeSessionList>
```

详细会话属性 API 呼叫

以下详细会话属性 API 呼叫可让您快速搜索最新的会话密钥信息，例如下列

- MAC 地址会话搜索（MAC 地址）
- 账号会话搜索（用户名）
- NAS IP 地址会话搜索（IP 地址与监控 ESS 节点）的目标相关联
- 跟踪会话 ID 搜索（跟踪会话 ID ）

MAC 地址会话搜索

您可以使用 MAC 地址 API 呼叫从当前，活动会话检索指定的 MAC 地址。此 API 呼叫列表从节点数据库表中获取的各种会话相关的信息。

MAC 地址 API 输出方案

此示例架构文件是 MAC 地址 API 呼叫的输出请求检索指定的 MAC 地址从当前活动会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0 "xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs: 元素 name=" 通过 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name=" 失败 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name="user_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="failure_reason" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="calling_station_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_group" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_server" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authen_protocol" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="framed_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_groups" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="access_service" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="authentication_method" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="execution_steps" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="radius_response" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="audit_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_identifier" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_policy_compliance" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_id" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="auth_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="message_code" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="service_selection_policy" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authorization_policy" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_store" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name=" 响应 " type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="service_type" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="cts_security_group" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="use_case" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="cisco_av_pair" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="ad_domain" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_username" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="radius_username" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_role" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_username" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_posture_token" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_radius_is_user_auth" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="selected_posture_server" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="selected_identity_store" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authentication_identity_store" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="azn_exp_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
```

```
        <xs: 元素 name="ext_pol_server_matched_rule" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="grp_mapping_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="identity_policy_matched_rule" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="nas_port_type" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="selected_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="sel_exp_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="selected_query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="eap_tunnel" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="tunnel_details" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="cisco_h323_attributes" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="cisco_ssg_attributes" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="other_attributes" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="response_time" type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name="nad_failure" type="xs:anyType" minOccurs="0"/>
        <xs: 元素 name="destination_ip_address" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_id" type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name="acct_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="acct_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="acct_session_id" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_status_type" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_session_time" type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name="acct_input_octets" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_output_octets" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_input_packets" type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name="acct_output_packets" type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name="acct_class" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_terminate_cause" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_multi_session_id" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_authentic" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="termination_action" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="session_timeout" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="idle_timeout" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_interim_interval" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_delay_time" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="event_timestamp" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_tunnel_connection" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acct_tunnel_packet_lost" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="security_group" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="cisco_h323_setup_time" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="cisco_h323_connect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="cisco_h323_disconnect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="framed_protocol" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name=" 入门 " type="xs:anyType" minOccurs="0"/>
        <xs: 元素 name=" 停止 " type="xs:anyType" minOccurs="0"/>
        <xs: 元素 name="ckpt_id" type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name=" 类型 " type="xs: 长期 " minOccurs="0"/>
        <xs: 元素 name="nad_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="vlan" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="dacl" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="authentication_type" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="interface_name" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name=" 原因 " type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="endpoint_policy" type="xs: 字符串 " minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

    <xs:element name="nas_ipv6_address" type="xs:string"/>
    <xs:complexType name="framed_ipv6_address_list">
      <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
```

```
  <xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>
</xs:schema>
```

调用 MAC 地址 API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点字段输入 MAC 地址 API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call>/<macaddress> ）：

```
https://acme123/admin/API/mnt/Session/MACAddress/0A:0B:0C:0D:0E:0F
```

注　确保使用 XX，您指定 MAC 地址：XX：XX：XX：XX：XX 格式。

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 MAC 地址 API 调用返回的数据

以下示例说明在触发 MACAddress API 调用时从活动会话列表返回的与会话相关的数据：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
-
<sessionParameters>
<passed xsi:type="xs: 布尔值 " >true</passed>
<failed xsi:type="xs: 布尔值 " >false</failed>
<user_name>hunter_thompson</user_name>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_port>50115</nas_port>
<identity_group>Profiled</identity_group>
```

<network_device_name>Core - Switch</network_device_name>
<acs_server>HAREESH - R 6 - 1 - PDP2</acs_server>
<authen_protocol>Lookup</authen_protocol>
 -
<network_device_groups>
设备 Type#All 设备类型，Location#All 位置
</network_device_groups>
<access_service>RADIUS</access_service>
< auth_acs_timestamp > -   - 2010 12 15T02:11: 12.359Z</auth_acs_timestamp>
<authentication_method>mab</authentication_method>
 -
<execution_steps>
11001,11017,11027,15008,15048,15004,15041,15004,15013,24209,24211,22037,15036,15048,15048,
15004,15016,11022,11002
</execution_steps>
<audit_session_id>0ACB6BA1000000351BBFBF8B</audit_session_id>
<nas_port_id>GigabitEthernet1/0/15</nas_port_id>
<nac_policy_compliance>Pending</nac_policy_compliance>
<auth_id>1291240762077361</auth_id>
< auth_acsview_timestamp > -   - 2010 12 15T02:11: 12.360Z</auth_acsview_timestamp>
<message_code>5200</message_code>
<acs_session_id>HAREESH - R 6 - 1 - PDP2/81148292/681</acs_session_id>
<service_selection_policy>MAB</service_selection_policy>
<identity_store>Internal Hosts</identity_store>
 -
<response>
{UserName= 00 - 14 - BF - 5A - 0C - 03; User - Name= 00 - 14 - BF - 5A - 0C - 03;
State=ReauthSession:0ACB6BA1000000351BBFBF8B; Class=CACS:0ACB6BA1000000351BBFBF8B:HAREESH
- R 6 - 1 - PDP2/81148292/681; Termination - Action=RADIUS - Request; cisco - av -
pair=url - redirect - acl=ACL - WEBAUTH - REDIRECT; cisco - av - pair=url - redirect=
https://HAREESH-R6-1-PDP2.cisco.com:8443/guestportal/gateway?sessionId=0ACB6BA1000000351BB
FBF8B&action=cwa; cisco - av - pair=ACS:CiscoSecure - Defined - ACL=#ACSACL# - IP - ACL -
DENY - 4ced8390; }
</response>
<service_type>Call Check</service_type>
<use_case>Host Lookup</use_case>
<cisco_av_pair>audit - session - id=0ACB6BA1000000351BBFBF8B</cisco_av_pair>
<acs_username>00:14:BF:5A:0C:0303</acs_username>
<radius_username>00:14:BF:5A:0C:0303</radius_username>
<selected_identity_store>Internal Hosts</selected_identity_store>
<authentication_identity_store>Internal Hosts</authentication_identity_store>
<identity_policy_matched_rule>Default</identity_policy_matched_rule>
<nas_port_type>Ethernet</nas_port_type>
<selected_azn_profiles>CWA</selected_azn_profiles>
 -
<other_attributes>
ConfigVersionId=44, DestinationIPAddress= 10.203.107.162, DestinationPort=1812,
Protocol=Radius, Framed - MTU=1500, EAP - Key - Name=,
CPMSessionID=0ACB6BA1000000351BBFBF8B, CPMSessionID=0ACB6BA1000000351BBFBF8B,
EndPointMACAddress = 00 - 14 - BF - 5A - 0C - 03, HostIdentityGroup=Endpoint 身份组：分
析，设备 Type=Device Type#All 设备类型，Location=Location#All 位置，型号 Name=Unknown，软件
Version=Unknown，设备 IP 地址 = 10.203.107.161, Called - Station - ID=04:FE:57F:7F:C0:8F
</other_attributes>
<response_time>77</response_time>
<acct_id>1291240762077386</acct_id>
< acct_acs_timestamp > -   - 2010 12 15T02:12: 30.779Z</acct_acs_timestamp>
< acct_acsview_timestamp > -   - 2010 12 15T02:12: 30.780Z</acct_acsview_timestamp>
<acct_session_id>00000038</acct_session_id>
<acct_status_type>Interim - Update</acct_status_type>
<acct_session_time>78</acct_session_time>
<acct_input_octets>13742</acct_input_octets>
<acct_output_octets>6277</acct_output_octets>
<acct_input_packets>108</acct_input_packets>
<acct_output_packets>66</acct_output_packets>

```
 -
<acct_class>
CACS:0ACB6BA1000000351BBFBF8B:HAREESH - R 6 - 1  - PDP2/81148292/681
</acct_class>
<acct_delay_time>0</acct_delay_time>
<started xsi:type="xs: 布尔值 " >false</started>
<stopped xsi:type="xs: 布尔值 " >false</stopped>
</sessionParameters>
```

账号会话搜索

您可以使用用户名 API 呼叫从当前，活动会话检索指定的账号。此 API 将列出从节点数据库表中获取的各种会话相关的信息。

用户名 API 输出方案

此示例架构文件是用户名 API 呼叫的输出请求检索指定的账号从当前活动会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0 "xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs: 元素 name=" 通过 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name=" 失败 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name="user_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="failure_reason" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="calling_station_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_group" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_server" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authen_protocol" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="framed_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_groups" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="access_service" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="authentication_method" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="execution_steps" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="radius_response" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="audit_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_identifier" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_policy_compliance" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_id" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="auth_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="message_code" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="service_selection_policy" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authorization_policy" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_store" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name=" 响应 " type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="service_type" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="cts_security_group" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="use_case" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="cisco_av_pair" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="ad_domain" type="xs: 字符串 " minOccurs="0"/>
```

```
<xs: 元素 name="acs_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="radius_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_role" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_posture_token" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_radius_is_user_auth" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_posture_server" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="authentication_identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="azn_exp_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="ext_pol_server_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="grp_mapping_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="identity_policy_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nas_port_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="sel_exp_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="eap_tunnel" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="tunnel_details" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_ssg_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="other_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="response_time" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="nad_failure" type="xs:anyType" minOccurs="0"/>
<xs: 元素 name="destination_ip_address" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_id" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="acct_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="acct_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_status_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_session_time" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_input_octets" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_output_octets" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_input_packets" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_output_packets" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_class" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_terminate_cause" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_multi_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_authentic" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="termination_action" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="session_timeout" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="idle_timeout" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_interim_interval" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_delay_time" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="event_timestamp" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_tunnel_connection" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_tunnel_packet_lost" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="security_group" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_setup_time" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_connect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_disconnect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="framed_protocol" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name=" 入门 " type="xs:anyType" minOccurs="0"/>
<xs: 元素 name=" 停止 " type="xs:anyType" minOccurs="0"/>
<xs: 元素 name="ckpt_id" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name=" 类型 " type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="nad_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="vlan" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="dacl" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="authentication_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="interface_name" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name=" 原因 " type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="endpoint_policy" type="xs: 字符串 " minOccurs="0"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:element name="nas_ipv6_address" type="xs:string"/>
  <xs:complexType name="framed_ipv6_address_list">
    <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>
</xs:schema>
```

调用用户名 API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址>/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点字段输入用户名 API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call>/<username> ）：

```
https://acme123/admin/API/mnt/Session/UserName/graham_hancock
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从用户名 API 调用返回的数据

当您将用户名 API 呼叫时，以下示例说明从活动会话列表返回的会话相关的数据

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<sessionParameters>
<passed xsi:type="xs: 布尔值 " >true</passed>
<failed xsi:type="xs: 布尔值 " >false</failed>
<user_name>graham_hancock</user_name>
<nas_ip_address> 10.203.107.161 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
```

<calling_station_id>00:14:BF:5A:0C:0303</calling_station_id>
<nas_port>50115</nas_port>
<identity_group>Profiled</identity_group>
<network_device_name>Core - Switch</network_device_name>
<acs_server>HAREESH - R 6 - 1 - PDP2</acs_server>
<authen_protocol>Lookup</authen_protocol>
 -
<network_device_groups>
设备 Type#All 设备类型 , Location#All 位置
</network_device_groups>
<access_service>RADIUS</access_service>
< auth_acs_timestamp > -   - 2010 12 15T02:11: 12.359Z</auth_acs_timestamp>
<authentication_method>mab</authentication_method>
 -
<execution_steps>
11001,11017,11027,15008,15048,15004,15041,15004,15013,24209,24211,22037,15036,15048,15048,
15004,15016,11022,11002
</execution_steps>
<audit_session_id>0ACB6BA1000000351BBFBF8B</audit_session_id>
<nas_port_id>GigabitEthernet1/0/15</nas_port_id>
<nac_policy_compliance>Pending</nac_policy_compliance>
<auth_id>1291240762077361</auth_id>
< auth_acsview_timestamp > -   - 2010 12 15T02:11: 12.360Z</auth_acsview_timestamp>
<message_code>5200</message_code>
<acs_session_id>HAREESH - R 6 - 1 - PDP2/81148292/681</acs_session_id>
<service_selection_policy>MAB</service_selection_policy>
<identity_store>Internal Hosts</identity_store>
 -
<response>
{UserName=graham_hancock; User - Name=graham_hancock;
State=ReauthSession:0ACB6BA1000000351BBFBF8B; Class=CACS:0ACB6BA1000000351BBFBF8B:HAREESH
- R 6 - 1 - PDP2/81148292/681; Termination - Action=RADIUS - Request; cisco - av -
pair=url - redirect - acl=ACL - WEBAUTH - REDIRECT; cisco - av - pair=url - redirect=
https://HAREESH-R6-1-PDP2.cisco.com:8443/guestportal/gateway?sessionId=0ACB6BA1000000351BB
FBF8B&action=cwa; cisco - av - pair=ACS:CiscoSecure - Defined - ACL=#ACSACL# - IP - ACL -
DENY - 4ced8390; }
</response>
<service_type>Call Check</service_type>
<use_case>Host Lookup</use_case>
<cisco_av_pair>audit - session - id=0ACB6BA1000000351BBFBF8B</cisco_av_pair>
<acs_username>graham_hancock</acs_username>
<radius_username>00:14:BF:5A:0C:0303</radius_username>
<selected_identity_store>Internal Hosts</selected_identity_store>
<authentication_identity_store>Internal Hosts</authentication_identity_store>
<identity_policy_matched_rule>Default</identity_policy_matched_rule>
<nas_port_type>Ethernet</nas_port_type>
<selected_azn_profiles>CWA</selected_azn_profiles>
 -
<other_attributes>
ConfigVersionId=44, DestinationIPAddress= 10.203.107.162, DestinationPort=1812,
Protocol=Radius, Framed - MTU=1500, EAP - Key - Name=,
CPMSessionID=0ACB6BA1000000351BBFBF8B, CPMSessionID=0ACB6BA1000000351BBFBF8B,
EndPointMACAddress = 00 - 14 - BF - 5A - 0C - 03, HostIdentityGroup=Endpoint 身份组 : 分
析，设备 Type=Device Type#All 设备类型，Location=Location#All 位置，型号 Name=Unknown，软件
Version=Unknown，设备 IP 地址 = 10.203.107.161, Called - Station - ID=04:FE:57F:7F:C0:8F
</other_attributes>
<response_time>77</response_time>
<acct_id>1291240762077386</acct_id>
< acct_acs_timestamp > -   - 2010 12 15T02:12: 30.779Z</acct_acs_timestamp>
< acct_acsview_timestamp > -   - 2010 12 15T02:12: 30.780Z</acct_acsview_timestamp>
<acct_session_id>00000038</acct_session_id>
<acct_status_type>Interim - Update</acct_status_type>
<acct_session_time>78</acct_session_time>
<acct_input_octets>13742</acct_input_octets>

```
<acct_output_octets>6277</acct_output_octets>
<acct_input_packets>108</acct_input_packets>
<acct_output_packets>66</acct_output_packets>
 -
<acct_class>
CACS:0ACB6BA1000000351BBFBF8B:HAREESH - R 6 - 1  - PDP2/81148292/681
</acct_class>
<acct_delay_time>0</acct_delay_time>
<started xsi:type="xs: 布尔值 " >false</started>
<stopped xsi:type="xs: 布尔值 " >false</stopped>
</sessionParameters>
```

NAS IP 地址会话搜索

您可以使用 IPAddress API 调用从当前会话中检索指定 NAS IP 地址（IPv4 或 IPv6 地址）的数据。此 API 将列出从节点数据库表中获取的各种会话相关的信息。

IP 地址 API 输出方案

此样本架构文件是用于从当前活动会话中检索指定 NAS IP 地址（IPv4 或 IPv6 地址）的 IPAddress API 调用的输出：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0 "xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs: 元素 name=" 通过 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name=" 失败 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name="user_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="failure_reason" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="calling_station_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_group" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_server" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authen_protocol" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="framed_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_groups" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="access_service" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="authentication_method" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="execution_steps" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="radius_response" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="audit_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_identifier" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nac_policy_compliance" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_id" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="auth_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="message_code" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="service_selection_policy" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authorization_policy" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_store" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name=" 响应 " type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="service_type" type="xs: 字符串 " minOccurs="0"/>
```

```
<xs: 元素 name="cts_security_group" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="use_case" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_av_pair" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="ad_domain" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acs_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="radius_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_role" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_posture_token" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_radius_is_user_auth" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_posture_server" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="authentication_identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="azn_exp_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="ext_pol_server_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="grp_mapping_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="identity_policy_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nas_port_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="sel_exp_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="eap_tunnel" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="tunnel_details" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_ssg_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="other_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="response_time" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="nad_failure" type="xs:anyType" minOccurs="0"/>
<xs: 元素 name="destination_ip_address" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_id" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="acct_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="acct_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_status_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_session_time" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_input_octets" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_output_octets" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_input_packets" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_output_packets" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_class" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_terminate_cause" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_multi_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_authentic" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="termination_action" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="session_timeout" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="idle_timeout" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_interim_interval" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_delay_time" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="event_timestamp" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_tunnel_connection" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_tunnel_packet_lost" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="security_group" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_setup_time" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_connect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_disconnect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="framed_protocol" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name=" 入门 " type="xs:anyType" minOccurs="0"/>
<xs: 元素 name=" 停止 " type="xs:anyType" minOccurs="0"/>
<xs: 元素 name="ckpt_id" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name=" 类型 " type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="nad_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="vlan" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="dacl" type="xs: 字符串 " minOccurs="0"/>
```

```
      <xs: 元素 name="authentication_type" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="interface_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name=" 原因 " type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="endpoint_policy" type="xs: 字符串 " minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="nas_ipv6_address" type="xs:string"/>
  <xs:complexType name="framed_ipv6_address_list">
    <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>
</xs:schema>
```

调用 NAS IP 地址 API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点字段输入 IP 地址 API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call>/<nasipaddress> ）：

```
https://acme123/admin/API/mnt/Session/IPAddress/10.10.10.10
```

注　确保分别使用 xxx.xxx.xxx.xxx 格式或压缩格式指定 IPv4 地址 /IPv6 地址（NAS IP 地址）。

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 IP 地址 API 调用返回的数据

当您将 IP 地址 API 呼叫时，以下示例说明从活动会话列表返回的会话相关的数据

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<sessionParameters>
<passed xsi:type="xs: 布尔值 " >true</passed>
<failed xsi:type="xs: 布尔值 " >false</failed>
<user_name>ipepvpnuser</user_name>
<nas_ip_address> 10.10.10.10 </nas_ip_address>
<nas_ipv6_address>2001:cdba::357:965</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>200:cdba:0000:0000:0000:0000:3157:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3247:9651</ipv6_address>
<ipv6_address>2001:cdba::3257:962</ipv6_address>
</framed_ipv6_address>
<calling_station_id> 172.23.130.90 </calling_station_id>
<nas_port>1015</nas_port>
<identity_group>iPEP - VPN - Group</identity_group>
<network_device_name>iPEP - HA - Routed</network_device_name>
<acs_server>HAREESH - R 6 - 1  - PDP2</acs_server>
<authen_protocol>PAP_ASCII</authen_protocol>
 -
<network_device_groups>
设备 Type#All 设备类型，Location#All 位置
</network_device_groups>
<access_service>RADIUS</access_service>
< auth_acs_timestamp >  -   -  2010 12 15T19:57: 29.885Z</auth_acs_timestamp>
<authentication_method>PAP_ASCII</authentication_method>
 -
<execution_steps>
11001,11017,15008,15048,15048,15004,15041,15004,15013,24210,24212,22037,15036,15048,15048,
15004,15016,11002
</execution_steps>
<audit_session_id>0acb6be4000000044D091DA9</audit_session_id>
<nac_policy_compliance>NotApplicable</nac_policy_compliance>
<auth_id>1291240762083580</auth_id>
< auth_acsview_timestamp >  -   -  2010 12 15T19:57: 29.887Z</auth_acsview_timestamp>
<message_code>5200</message_code>
<acs_session_id>HAREESH - R 6 - 1  - PDP2/81148292/693</acs_session_id>
<service_selection_policy>iPEP - VPN</service_selection_policy>
<identity_store>Internal Users</identity_store>
 -
<response>
{User - Name=ipepvpnuser; State=ReauthSession:0acb6be4000000044D091DA9;
Class=CACS:0acb6be4000000044D091DA9:HAREESH - R 6 - 1  - PDP2/81148292/693; Termination -
Action=RADIUS - Request; }
</response>
<service_type>Framed</service_type>
 -
<cisco_av_pair>
audit - session - id=0acb6be4000000044D091DA9, ipep - proxy=true
</cisco_av_pair>
<acs_username>ipepvpnuser</acs_username>
<radius_username>ipepvpnuser</radius_username>
<selected_identity_store>Internal Users</selected_identity_store>
<authentication_identity_store>Internal Users</authentication_identity_store>
<identity_policy_matched_rule>Default</identity_policy_matched_rule>
<nas_port_type>Virtual</nas_port_type>
<selected_azn_profiles>iPEP - Unknown - Auth - Profile</selected_azn_profiles>
<tunnel_details>Tunnel - Client - Endpoint=  (tag=0 )  172.23.130.90 </tunnel_details>
 -
```

```
<other_attributes>
ConfigVersionId=44, DestinationIPAddress= 10.203.107.162, DestinationPort=1812,
Protocol=Radius, Framed - Protocol=PPP, Proxy - State=Cisco 安全 ACS9e733142 - 070a - 11e0
- c 000 - 000000000000  -  2906094480 - 3222 , CPMSessionID=0acb6be4000000044D091DA9,
CPMSessionID=0acb6be4000000044D091DA9，设备 Type=Device Type#All 设备类型，
Location=Location#All 位置，型号 Name=Unknown，软件 Version=Unknown，设备 IP 地址 =
10.203.107.228, Called - Station - ID=172.23.130.94
</other_attributes>
<response_time>20</response_time>
<acct_id>1291240762083582</acct_id>
< acct_acs_timestamp >  -   -  2010 12 15T19:57: 30.281Z</acct_acs_timestamp>
< acct_acsview_timestamp >  -   -  2010 12 15T19:57: 30.283Z</acct_acsview_timestamp>
<acct_session_id>F1800007</acct_session_id>
<acct_status_type>Start</acct_status_type>
 -
<acct_class>
CACS:0acb6be4000000044D091DA9:HAREESH - R 6 - 1  - PDP2/81148292/693
</acct_class>
<acct_delay_time>0</acct_delay_time>
<framed_protocol>PPP</framed_protocol>
<started xsi:type="xs: 布尔值 " >true</started>
<stopped xsi:type="xs: 布尔值 " >false</stopped>
</sessionParameters>
```

跟踪会话 ID 搜索

您可以使用会话 ID API 呼叫从当前，活动会话检索指定的跟踪会话。此 API 呼叫列表从节点数据库表中获取的各种会话相关的信息。

跟踪会话 ID API 输出方案

此示例架构文件是跟踪会话 ID API 呼叫的输出请求检索指定的审计会话 ID 从当前活动会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0 "xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs: 元素 name=" 通过 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name=" 失败 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name="user_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="failure_reason" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="calling_station_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_port" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="identity_group" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="acs_server" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authen_protocol" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="framed_ip_address" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="network_device_groups" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="access_service" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="auth_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="authentication_method" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="execution_steps" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="radius_response" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="audit_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="nas_identifier" type="xs: 字符串 " minOccurs="0"/>
```

```
<xs: 元素 name="nas_port_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_policy_compliance" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="auth_id" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="auth_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="message_code" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acs_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="service_selection_policy" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="authorization_policy" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name=" 响应 " type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="service_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cts_security_group" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="use_case" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_av_pair" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="ad_domain" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acs_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="radius_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_role" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_username" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_posture_token" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nac_radius_is_user_auth" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_posture_server" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="authentication_identity_store" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="azn_exp_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="ext_pol_server_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="grp_mapping_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="identity_policy_matched_rule" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="nas_port_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="sel_exp_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="selected_query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="eap_tunnel" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="tunnel_details" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_h323_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="cisco_ssg_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="other_attributes" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="response_time" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="nad_failure" type="xs:anyType" minOccurs="0"/>
<xs: 元素 name="destination_ip_address" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_id" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="acct_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
<xs: 元素 name="acct_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_status_type" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_session_time" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_input_octets" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_output_octets" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_input_packets" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_output_packets" type="xs: 长期 " minOccurs="0"/>
<xs: 元素 name="acct_class" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_terminate_cause" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_multi_session_id" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_authentic" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="termination_action" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="session_timeout" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="idle_timeout" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_interim_interval" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_delay_time" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="event_timestamp" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_tunnel_connection" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="acct_tunnel_packet_lost" type="xs: 字符串 " minOccurs="0"/>
<xs: 元素 name="security_group" type="xs: 字符串 " minOccurs="0"/>
```

```
      <xs: 元素 name="cisco_h323_setup_time" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="cisco_h323_connect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="cisco_h323_disconnect_time" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="framed_protocol" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name=" 入门 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name=" 停止 " type="xs:anyType" minOccurs="0"/>
      <xs: 元素 name="ckpt_id" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name=" 类型 " type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="nad_acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
      <xs: 元素 name="vlan" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="dacl" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="authentication_type" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="interface_name" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name=" 原因 " type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="endpoint_policy" type="xs: 字符串 " minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="nas_ipv6_address" type="xs:string"/>
  <xs:complexType name="framed_ipv6_address_list">
    <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>

</xs:schema>
```

调用跟踪会话 ID API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与审计会话 ID API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/Session/Active/SessionID/ < *审计会话 ID* >/0 ）：

```
https://acme123/admin/API/mnt/Session/Active/SessionID/0A000A770000006B609A13A9/0
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从审计会话 ID API 调用返回的数据

在调用跟踪会话 ID API 呼叫时，以下示例说明从活动会话列表返回的会话相关的数据

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
- <activeSessionList noOfActiveSession="1">
   - <activeSession>
      <calling_station_id>00:50:56:10:13:02</calling_station_id>
      <session_state_bit>0</session_state_bit>
      <session_source>0</session_source>
      <acct_session_time>0</acct_session_time>
      <nas_ip_address> 10.0.10.119 </nas_ip_address>
      <nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
      <framed_ipv6_address>
      <ipv6_address>200:cdba:0000:0000:0000:0000:3257:9652</ipv6_address>
      <ipv6_address> 2001:cdba:0:0:0:0:3257:9651</ipv6_address>
      <ipv6_address>2001:cdba::3257:9652</ipv6_address>
      </framed_ipv6_address>
      <nas_port_id>GigabitEthernet1/0/15</nas_port_id>
      <auth_method>dot1x</auth_method>
      <auth_protocol>PEAP （EAP - MSCHAPv2 ） </auth_protocol>
      <posture_status>Compliant</posture_status>
      <endpoint_policy>Undetermined</endpoint_policy>
      <server>acme123</server>
      <paks_in>0</paks_in>
      <paks_out>0</paks_out>
      <bytes_in>0</bytes_in>
      <bytes_out>0</bytes_out>
   </activeSession>
</activeSessionList>
```

过时的会话

某些设备，例如无线局域网控制器 (WLC)，可以允许过时会话徘徊。在这种情况下，您可以使用 HTTP **删除** API 呼叫手动删除非活动会话。为此，请使用 **curl**，用于传输数据的免费 3 方命令行工具与 URL （HTTP，HTTPS ）语法。

ESS 不再跟踪这些会话。这是为了缓解问题，当 ESS 长时间失去网络连接，并沿着 Forensic 从 WLC/NAD 的记账停止。您可以清除使用此 API 的 ESS 的此类过时的信息。

备注　GNU Wget，检索的免费程序文件使用 HTTP，并且 HTTPS，不支持 HTTP **删除** API 呼叫。

删除已过期的会话

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

注　API 呼叫区分大小写，并且必须小心输入。可变的 <mntnode> 代表监控 ESS 节点的 Cisco。

步骤 4　手动删除 MAC 地址的过期的会话，请发出在命令行中以下 API 呼：

```
curl  -  X DELETE https:// <mntnode>/admin/API/mnt/Session/Delete/MACAddress/<madaddress>
```

步骤 5　手动删除会话 ID 的过期的会话，请发出在命令行中以下 API 呼叫：

```
curl  -  X DELETE https:// <mntnode>/admin/API/mnt/Session/Delete/SessionID/<sid#>
```

步骤 6　手动删除监控节点的所有会话，请发出在命令行中以下 API 呼叫：

```
curl  -  X DELETE https:// <mntnode>/admin/API/mnt/Session/Delete/All
```

相关主题

- 验证监控节点，第 1-2 页

故障排除的查询 API

本章提供示例并描述如何使用单独的 Cisco Prime 网络控制系统 (NCS) REST API 呼叫。

Cisco Prime NCS API 呼叫

Cisco Prime NCS API 呼叫获取有关该目标监视器 ESS 包括节点版本和类型、故障原因、身份验证状态和客户状态的节点会话的思科重要故障排除信息的框架。

使用查询 API 呼叫的故障排除 Cisco ISE

Cisco Prime 排除 API 呼叫的 NCS 发送状态请求到目标监控您的 Cisco ISE 配置的 Cisco ESS 节点并检索以下诊断相关的信息。

- 节点版本和类型（使用版本 API 呼叫）
- 故障原因（使用 FailureReasons API 呼叫）
- 身份验证状态（使用 AuthStatus API 呼叫）
- 客户状态（使用 AcctStatus API 呼叫）

节点版本和类型 API 呼叫

您可以使用版本 API 呼叫测试其它编程接口 (PI) 服务和每个节点凭证。本节提供请求 Cisco ISE 软件和节点类型的版本提供架构文件输出示例、方法通过调用此 API 呼叫和返回节点版本和类型的示例发出后，在此 API 调用。

节点类型可以是以下任意值：

- STANDALONE_MNT_NODE = 0
- ACTIVE_MNT_NODE= 1
- BACKUP_MNT_NODE = 2
- NOT_AN_MNT_NODE = 3

版本 API 输出方案

此示例架构文件是版本 API 呼叫的输出在发送后到目标监控 ESS 节点的 Cisco：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name=" 产品 " type=" 产品 "/>

  <xs:complexType name=" 产品 " >
    <xs:sequence>
      <xs: 元素 name=" 版本 " type="xs: 字符串 " minOccurs="0"/>
  <xs: 元素 name="type_of_node" type="xs:int"/>
    </xs:sequence>
    <xs: 属性 name=" 名称 " type="xs: 字符串 "/>
  </xs:complexType>
</xs:schema>
```

调用版本 API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

如果您的登录不成功，单击登录时**出现问题？**链接在登录页并按照第 2 步中的说明。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与版本 API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call> ）：

```
https://acme123/admin/API/mnt/Version
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从版本 API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的版本 API 调用以下示例说明返回的数据。此 API 调用返回目标节点的以下两个值。

- 节点版本（本示例显示 1.0.3.032 ）。
- ESS 监控节点的 Cisco 的类型（本示例显示了“1 ”，这意味着监控 ESS 节点）的虚拟化思科。

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<product name=" 思科身份服务引擎 " >
<version>1.0.3.032</version>
<type_of_node>1</type_of_node>
</product>
```

故障原因 API 呼叫

您可以使用 FailureReasons API 调用返回故障原因列表在身份验证状态检查返回的完成目标节点。本节提供架构文件输出示例，请求的 Cisco 记录的所有故障原因列表方法监控 ESS 节点通过调用此 API 调用，并发出在此 API 呼叫后返回的故障原因的示例。返回的每个故障原因包括显示的以下元素。表 3-1

备注 有关使用 Cisco ISE 故障原因编辑器的详细信息访问故障原因的完整列表，请参阅 Cisco ISE 故障原因报告，第 A-1 页。

表 3-1　　思科身份服务引擎的产品文档

故障原因元素	示例
故障原因 ID	<failureReason id="11011">
代码	<11011 RADIUS 监听程序 failed>
原因	<Could 没有开放一个或多个端口用于接收 RADIUS requests>
解决方法	端口 1812，1813，1645 和 1646 未被 system> 的另一个进程的 <Ensure

备注 使用 Cisco ISE 用户界面（单击监控 > 报告 > 目录 > 故障原因），**您还可以检查故障原因**报告，将显示故障原因报告。

FailureReasons API 输出方案

此示例架构文件是 FailureReasons API 呼叫的输出在发送请求后到目标监视器 ESS 节点的 Cisco：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="failureReasonList" type="failureReasonList"/>

  <xs:complexType name="failureReasonList">
    <xs:sequence>
      <xs: 元素 name="failureReason" type="failureReason" minOccurs="0" maxOccurs=" 无边际的
"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="failureReason">
    <xs:sequence>
      <xs: 元素 name=" 代码 " type="xs: 字符串 " minOccurs="0"/>
```

```
      <xs: 元素 name=" 原因 "  type="xs: 字符串 "  minOccurs="0"/>
      <xs: 元素 name=" 分析 "  type="xs: 字符串 "  minOccurs="0"/>
    </xs:sequence>
    <xs: 属性 name="id"  type="xs: 字符串 "/>
  </xs:complexType>
</xs:schema>
```

调用 FailureReasons API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

如果您的登录不成功，单击登录时**出现问题？**链接在登录页并按照第 2 步中的说明。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 FailureReasons API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call> ）：

```
https://acme123/admin/API/mnt/FailureReasons
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 FailureReasons API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的一 FailureReasons API 调用以下示例说明返回的数据。此 API 调用返回故障列表从目标节点证明，并且每个故障原因由故障 ID、失败代码、原因和分辨率定义的（如果知道）。

备注　以下 FailureReasons API 呼叫示例仅显示可以返回数据的小型示例。

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<failureReasonList>
 -
<failureReason id="100001">
 -
<code>
为客户端失败的 100001 个 AUTHMGR - 5 - FAIL 权限
</code>
<cause>This 可以或可能不指示 violation</cause>
```

```
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
 -
<failureReason id="100002">
 -
<code>
在接口的 100002 个 AUTHMGR - 5 - SECURITY_VIOLATION 安全违规
</code>
<cause>This 可以或可能不指示 violation</cause>
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
 -
<failureReason id="100003">
 -
<code>
未授权 100003 个 AUTHMGR - 5 - UNAUTHORIZED 的接口
</code>
<cause>This 可以或可能不指示 violation</cause>
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
 -
<failureReason id="100004">
 -
<code>
为客户端失败的 100004 个 DOT1X - 5 - FAIL 身份验证
</code>
<cause>This 可以或可能不指示 violation</cause>
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
 -
<failureReason id="100005">
为 client</code> 失败的 <code>100005 MAB - 5 - FAIL 身份验证
<cause>This 可以或可能不指示 violation</cause>
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
 -
<failureReason id="100006">
 -
<code>
100006 个 RADIUS - 4 - RADIUS_DEAD RADIUS 服务器未响应
</code>
<cause>This 可以或可能不指示 violation</cause>
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
 -
<failureReason id="100007">
```

```
 -
<code>
ACL 未配置的 100007 个 EPM - 6 - POLICY_APP_FAILURE 接口
</code>
<cause>This 可以或可能不指示 violation</cause>
 -
<resolution>
请根据您组织的策略审核并解决
</resolution>
</failureReason>
```

相关主题

- 验证监控节点，第 1-2 页
- 附录 A“Cisco ISE 故障原因报告”

身份验证状态 API 呼叫

您可以使用 AuthStatus API 呼叫检查会话的身份验证状态目标节点的。查询与此 API 呼叫关联的返回的指定的 MAC 地址至少需要一个 MAC 地址被搜索到一个匹配项，与新记录的一个用户可配置限制。

本节提供架构文件输出示例，发送方式请求搜索会话在目标监控模式的身份验证状态通过调用此 API 调用，并发出在此 API 呼叫后返回的数据的示例。

AuthStatus API 呼叫让您配置以下搜索相关的参数。

- 持续时间 - 定义尝试搜索和检索身份验证状态记录与选中的 MAC 地址关联的秒数。有效的用户可配置值范围为 1 秒至 864000 秒（10 天）。如果您输入 0 秒的值，此字段指定 10 天的默认持续时间。
- 记录 - 定义会话记录的数量以及每个 MAC 地址中搜索的。用户可配置的有效值范围是 1 至 500 条记录。如果输入 0，则指定 200 记录默认设置。

  注　如果您为持续时间和记录参数指定该值 0，此 API 调用返回只需要最新的身份验证会话记录将指定的 MAC 地址。

  这是 URL 的通用表单示例与持续时间和记录属性的：

  https://10.10.10.10/admin/API/mnt/AuthStatus/MACAddress/01:23:45:67:89:98/900000/2/All

- 属性 - 在身份验证状态表中定义使用 AuthStatus API 呼叫，从身份验证状态搜索返回属性的数量。有效值包括 0（默认），所有或 user_name+acs_timestamp（参阅 AuthStatus 方案示例，AcctStatus API 输出方案，第 3-12 页）。
  - 如果您输入“0”，定义的属性返回。表 3-2 这些在输出方案的 restAuthStatus 部分列出。
  - 如果您输入“所有”，属性全套返回。这些在输出方案的 fullRESTAuthStatus 部分列出。
  - 如果您在 user_name+acs_timestamp 的方案输入列出的值，那些属性返回。user_name 和 acs_timestamp 属性在输出方案的 restAuthStatus 部分列出。

表 3-2　　身份验证状态表属性

属性	说明
name=" 通过 " 或 name=" 失败 "	身份验证状态结果： • 已通过 • 失败
name="user_name"	用户名
name="nas_ip_address"	IP 地址 / 主机名网络接入设备的
name="nas_ipv6_address"	网络接入设备的 IPv6 地址 / 主机名
name="failure_reason"	会话身份验证失败的原因
name="calling_station_id"	源 IP 地址
name="nas_port"	网络访问服务器端口
name="identity_group"	包括相关用户和主机的逻辑组
name="network_device_name"	网络设备的名称
name="acs_server"	Cisco ISE 设备的名称
name="eap_authentication"	为验证请求的可扩展身份验证协议 (EAP) 使用的方法
name="framed_ip_address"	为特定用户配置的地址
name="framed_ipv6_address"	为特定用户配置的地址
network_device_groups"	包括相关网络设备的逻辑组
name="access_service"	应用的访问服务
name="acs_timestamp"	与 Cisco ISE 验证请求相关的时间戳
name="authentication_method"	确定在身份验证的方法
name="execution_steps"	消息代码列表中记录的每个诊断消息的，在处理请求时
name="radius_response"	RADIUS 响应的类型（例如， VLAN 或 ACL ）
name="audit_session_id"	身份验证会话的 ID
"nas_identifier" name=	网络访问服务器 (NAS) 联合特定资源
name="nas_port_id"	使用的 NAS 端口的 ID
name="nac_policy_compliance"	反映状态（兼容或不兼容）
name="selected_azn_profiles"	识别用于授权的配置文件
name="service_type"	表示帧的用户
name="eap_tunnel"	适用于 EAP 身份验证或外部使用的方法的隧道
name="message_code"	定义要处理的请求结果跟踪消息的标识符
name="destination_ip_address"	标识目标 IP 地址

AuthStatus API 输出方案

此示例架构文件是 AuthStatus API 呼叫的输出在发送后到目的地的监控 ESS 节点的 Cisco 指定的会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs: 元素 name="authStatusOutputList" type="fullRESTAuthStatusOutputList"/>

  <xs:complexType name="fullRESTAuthStatusOutputList">
    <xs:sequence>
      <xs: 元素 name="authStatusList" type="fullRESTAuthStatusList" minOccurs="0"
maxOccurs=" 无边际的 "/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="fullRESTAuthStatusList">
    <xs:sequence>
      <xs: 元素 name="authStatusElements" type="fullRESTAuthStatus" minOccurs="0"
maxOccurs=" 无边际的 "/>
    </xs:sequence>
    <xs: 属性 name=" 密钥 " type="xs: 字符串 "/>
  </xs:complexType>

  <xs:complexType name="fullRESTAuthStatus">
    <xs:complexContent>
      <xs: 分机 base="restAuthStatus">
        <xs:sequence>
          <xs: 元素 name="id" type="xs: 长期 " minOccurs="0"/>
          <xs: 元素 name="acsview_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
          <xs: 元素 name="acs_session_id" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="service_selection_policy" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="authorization_policy" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="identity_store" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name=" 响应 " type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="cts_security_group" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="use_case" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="cisco_av_pair" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="ad_domain" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="acs_username" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="radius_username" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="nac_role" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="nac_username" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="nac_posture_token" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="nac_radius_is_user_auth" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="selected_posture_server" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="selected_identity_store" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="authentication_identity_store" type="xs: 字符串 "

minOccurs="0"/>
          <xs: 元素 name="azn_exp_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="ext_pol_server_matched_rule" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="grp_mapping_pol_matched_rule" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="identity_policy_matched_rule" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="nas_port_type" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="sel_exp_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="selected_query_identity_stores" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="tunnel_details" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="cisco_h323_attributes" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="cisco_ssg_attributes" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="other_attributes" type="xs: 字符串 " minOccurs="0"/>
          <xs: 元素 name="response_time" type="xs: 长期 " minOccurs="0"/>
          <xs: 元素 name="nad_failure" type="xs:anyType" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="restAuthStatus">
    <xs:sequence>
```

```
        <xs: 元素 name=" 通过 " type="xs:anyType" minOccurs="0"/>
        <xs: 元素 name=" 失败 " type="xs:anyType" minOccurs="0"/>
        <xs: 元素 name="user_name" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="nas_ip_address" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="failure_reason" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="calling_station_id" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="nas_port" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="identity_group" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="network_device_name" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acs_server" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="eap_authentication" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="framed_ip_address" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="network_device_groups" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="access_service" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="acs_timestamp" type="xs: 日期 - 时间 " minOccurs="0"/>
        <xs: 元素 name="authentication_method" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="execution_steps" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="radius_response" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="audit_session_id" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="nas_identifier" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="nas_port_id" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="nac_policy_compliance" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="selected_azn_profiles" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="service_type" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="eap_tunnel" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="message_code" type="xs: 字符串 " minOccurs="0"/>
        <xs: 元素 name="destination_ip_address" type="xs: 字符串 " minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

    <xs:element name="nas_ipv6_address" type="xs:string"/>
    <xs:complexType name="framed_ipv6_address_list">
      <xs:sequence minOccurs="0" maxOccurs="8"><xs:element name="ipv6_address"
type="xs:string" />
      </xs:sequence>
    </xs:complexType>
    <xs:element name="framed_ipv6_address" type="framed_ipv6_address_list" minOccurs="1"
maxOccurs="1"/>
</xs:schema>
```

调用 AuthStatus API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

如果您的登录不成功，单击登录时**出现问题？**链接在登录页并按照第 2 步中的说明。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 AuthStatus API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call>/MACAddress/ <macaddress>/<seconds>/<numberofrecordspermacaddress>/All ）：

```
https://acme123/admin/API/mnt/AuthStatus/MACAddress/00:50:56:10:13:02/120/100/All
```

注　REST API 呼叫区分大小写。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 AuthStatus API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的一 AuthStatus API 调用以下示例说明返回的数据：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<authStatusOutputList>
 -
<authStatusList key="00: 0C:29:DE:94:29:46:F3:B8"><authStatusElements>
 -
<passed xsi:type="xs: 布尔值 " >true</passed>
<failed xsi:type="xs: 布尔值 " >false</failed>
<user_name>suser77</user_name>
<nas_ip_address> 10.77.152.209 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<calling_station_id>00:0C:0329:46:F3:B8</calling_station_id>
<identity_group>User 身份组 :Guest</identity_group>
<acs_server>guest - 240</acs_server>
< acs_timestamp >  -   -  2012 10 05T10:50: 56.515Z</acs_timestamp>
<execution_steps>5231</execution_steps>
<message_code>5231</message_code>
<id>1349422277270561</id>
< acsview_timestamp >  -   -  2012 10 05T10:50: 56.517Z</acsview_timestamp>
<identity_store>Internal Users</identity_store>
<response_time>146</response_time>
<other_attributes>ConfigVersionId=81, EndPointMACAddress= 00 - 0 C -  29 - 46  - F3 - B8,
PortalName=DefaultGuestPortal,
CPMSessionID=0A4D98D1000001F26F0C04D9, CiscoAVPair=</other_attributes>
</authStatusElements>
 -
<authStatusElements>
<passed xsi:type="xs: 布尔值 " >true</passed>
<failed xsi:type="xs: 布尔值 " >false</failed>
<user_name>00:0C:0329:46:F3:B8</user_name>
<nas_ip_address> 10.77.152.209 </nas_ip_address>
<nas_ipv6_address>2001:cdba::3257:9652</nas_ipv6_address>
<framed_ipv6_address>
<ipv6_address>2001:cdba:0000:0000:0000:0000:3257:9652</ipv6_address>
<ipv6_address> 2001:cdba:0:0:0:0:3257:9652</ipv6_address>
<ipv6_address>2001:cdba::3257:9652</ipv6_address>
</framed_ipv6_address>
<calling_station_id>00:0C:0329:46:F3:B8</calling_station_id>
<identity_group>Guest_IDG</identity_group>
<network_device_name>switch</network_device_name>
<acs_server>guest - 240</acs_server>
<authentication_method>mab</authentication_method>
<authentication_protocol>Lookup</authentication_protocol>
< acs_timestamp >  -   -  2012 10 05T10:49: 47.915Z</acs_timestamp>
```

```
<execution_steps>11001,11017,11027,15049,15008,15048,15048,15004,15041,15006,15013,24209,2
421
1,22037,15036,15048,15004,15016,11022,11002</execution_steps>
<response> { 用户名
=00: 0C:29:DE:94:29:46:F3:B8; User - Name= 00 - 0 C -  29 - 46  - F3 - B8;
State=ReauthSession:0A4D98D1000001F26F0C04D9;
Class=CACS:0A4D98D1000001F26F0C04D9: 访客 240/138796808/76;
Termination - Action=RADIUS - Request; Tunnel - Type= （tag=1） VLAN;
Tunnel - Medium - Type= （tag=1） 802; Tunnel - Private - Group - ID= （tag=1） 2;
cisco - av - pair=url - redirect - acl=ACL - WEBAUTH - REDIRECT;
cisco - av - pair=url - redirect= https://guest-240.cisco.com:8443/guestportal/gateway ?
sessionId=0A4D98D1000001F26F0C04D9&action=cwa;
cisco - av - pair=ACS:CiscoSecure - Defined - ACL=#ACSACL# - IP - pre - posture -
506e980a;
cisco - av - pair=profile - name=WindowsXP - Workstation; } </response
><audit_session_id>0A4D98D1000001F26F0C04D9</audit_session_id><nas_po
rt_id>GigabitEthernet1/0/17</nas_port_id><posture_status>Pending</posture_status>
<selected_azn_profiles>CWA_Redirect</selected_azn_profiles>
<service_type>Call Check</service_type>
<message_code>5200</message_code>
<nac_policy_compliance>Pending</nac_policy_compliance>
<id>13494222777270556</id>
< acsview_timestamp >  -   -  2012 10 05T10:49: 47.915Z</acsview_timestamp>
<identity_store>Internal Endpoints</identity_store>
<response_time>13</response_time>
<other_attributes>ConfigVersionId=81, DestinationPort=1812, Protocol=Radius,
AuthorizationPolicyMatchedRule=CWA_Redirect,
NAS - Port=50117, Framed - MTU=1500, NAS - Port - Type=Ethernet, EAP 密钥 N
ame=, cisco - nas - port=GigabitEthernet1/0/17, AcsSessionID=guest - 240/138796808/76，我们
eCase=Host 查找 , SelectedAuthenticationIdentityStores=Internal
终端 , ServiceSelectionMatchedRule=MAB, IdentityPolicyMatchedRule=Default, CPMS
essionID=0A4D98D1000001F26F0C04D9, EndPointMACAddress= 00 - 0 C -  29 - 46  - F3 - B8,
EndPointM
atchedProfile=WindowsXP - Workstation, ISEPolicySetName=Default, HostIdentityGroup=E
ndpoint 身份组 :Guest_IDG, 设备 Type=Device Type#All 设备
类型 , Location=Location#All 位置 , 设备 IP
Address= 10.77.152.209, Called - Station - ID=00:24:F7:73: 9A:91, CiscoAVPair=audit - sess
ion - id=0A4D98D1000001F26F0C04D9</other_attributes>
 -
</authStatusElements>
 -
</authStatusList>
 -
</authStatusOutputList>
```

客户状态 API 呼叫

您可以使用 AcctStatus API 呼叫检索有关目的节点的最新的设备和会话帐户信息。本节提供架构文件输出示例，发送的一个需要方法最新的设备和会话信息通过调用此 API 调用，并发出在此 API 呼叫后返回的数据的示例。AcctStatus API 呼叫让您配置一个与时间相关的参数。

- 持续时间 - 定义尝试搜索和检索最新客户设备记录与选中的 MAC 地址关联的秒数。有效的用户可配置值范围为 1 秒至 432000 秒（5 天）。例如，
  - 如果您输入2400秒（40分钟）的值，这意味着您想要是可用在过去40分钟内指定的MAC地址的最新客户设备记录。
  - 如果您输入0秒的值，这为15分钟（900秒）的默认持续时间。这意味着您希望在此时段可用的指定的 MAC 地址的最新客户设备记录。

AcctList API 呼叫提供以下客户状态数据字段，当 API 输出（请参阅表 3-3）：

表 3-3　　客户状态数据字段

数据字段	说明
MAC 地址	客户端的 MAC 地址
跟踪会话 ID	身份验证会话 ID
传入的数据包数	数据包收到的计数总计
传出的数据包数	数据包传输的计数总计
收到的字节数	字节收到的计数总计
外发的字节数	字节传输的计数总计
会话超时	当前会话的持续时间

AcctStatus API 输出方案

此示例架构文件是 AcctStatus API 呼叫的输出在发送后到目的地的监控 ESS 节点的 Cisco 指定的会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="acctStatusOutputList" type="restAcctStatusOutputList"/>

  <xs:complexType name="restAcctStatusOutputList">
    <xs:sequence>
      <xs: 元素 name="acctStatusList" type="restAcctStatusList" minOccurs="0" maxOccurs=" 无边际的 "/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="restAcctStatusList">
    <xs:sequence>
      <xs: 元素 name="acctStatusElements" type="restAcctStatus" minOccurs="0" maxOccurs=" 无边际的 "/>
    </xs:sequence>
    <xs: 属性 name="macAddress" type="xs: 字符串 "/>
    <xs: 属性 name=" 用户名 " type="xs: 字符串 "/>
  </xs:complexType>

  <xs:complexType name="restAcctStatus">
    <xs:sequence>
      <xs: 元素 name="calling_station_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="audit_session_id" type="xs: 字符串 " minOccurs="0"/>
      <xs: 元素 name="paks_in" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="paks_out" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="bytes_in" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="bytes_out" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name="session_time" type="xs: 长期 " minOccurs="0"/>
      <xs: 元素 name=" 用户名 " type="xs: 字符串 " minOccurs="0"/>
  <xs: 元素 name=" 服务器 " type="xs: 字符串 " minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

调用 AcctStatus API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址>/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

如果您的登录不成功，单击登录时**出现问题？**链接在登录页并按照第 2 步中的说明。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 AcctStatus API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/ <specific - api - call>/MACAddress/<macaddress>/<durationofcurrenttime> ）：

```
https://acme123/admin/API/mnt/AcctStatus/MACAddress/00:26:82:7B:D2:51/1200
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 AcctStatus API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的一 AcctStatus API 调用以下示例说明返回的数据：

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
 -
<acctStatusOutputList>
 -
<acctStatusList macAddress="::9C:A3:7D:48">
 -
<acctStatusElements>
<calling_station_id>00:25: 9C:A3:7D:48</calling_station_id>
<audit_session_id>0acb6b0b0000000B4D0C0DBD</audit_session_id>
<paks_in>0</paks_in>
<paks_out>0</paks_out>
<bytes_in>0</bytes_in>
<bytes_out>0</bytes_out>
<session_time>240243</session_time>
<server>HAREESH - R 6 - 1  - PDP1</server>
</acctStatusElements>
</acctStatusList>
</acctStatusOutputList>
```

权限 REST API 更改

本章提供示例并描述如何使用授权思科身份服务引擎此版本支持的 (CoA) REST API 调用以下单独更改。

简介

CoA API 呼叫用于发送会话身份验证和会话断开命令提供了到监控您的 Cisco ISE 配置的指定的 Cisco ESS 节点。

CoA Session Management API 呼叫

CoA 会话管理 API 呼叫允许您发送重新进行身份验证和断开命令到目标的监控您的 Cisco ISE 配置的 Cisco 指定的会话 ESS 节点：

- 默认端口（Reauth ）
- 会话断开（断开）

默认端口 API 呼叫

默认端口 API 呼叫构成以下类型的

- REAUTH_TYPE_DEFAULT = 0
- REAUTH_TYPE_LAST = 1
- REAUTH_TYPE_RERUN = 2

Reauth API 输出方案

此示例架构文件是 Reauth API 呼叫的输出在发送后到目的地的监控 ESS 节点的 Cisco 指定的会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs: 元素 name="remoteCoA" type="coAResult"/>
<xs:complexType name="coAResult">
    <xs:sequence>
      <xs: 元素 name=" 结果 " type="xs: 布尔值 " minOccurs="0"/>
    </xs:sequence>
    <xs: 属性 name="requestType" type="xs: 字符串 "/>
  </xs:complexType>
</xs:schema>
```

调用 Reauth API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与 Reauth API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/CoA/ <specific - api - call>/<macaddress>/<reauthtype> ：

```
https://acme123/admin/API/mnt/CoA/Reauth/server12/00:26:82:7B:D2:51/1
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样从 Reauth API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的一 Reauth API 调用以下示例说明返回的数据。两种可能的结果可以从调用此命令返回：

- true 表明命令已成功执行。
- 错误表示命令并未执行（由于各种情况）。

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
-
<remoteCoA requestType="next - ccode - and - reauth">
<results>true</results>
</remoteCoA>
```

会话断开 API 呼叫

会话断开 API 呼叫构成以下断开端口选项类型：

- DYNAMIC_AUTHZ_PORT_DEFAULT = 0
- DYNAMIC_AUTHZ_PORT_BOUNCE = 1
- DYNAMIC_AUTHZ_PORT_SHUTDOWN = 2

断开 API 输出方案

此示例架构文件是断开 API 呼叫的输出在发送后到目的地的监控 ESS 节点的 Cisco 指定的会话数

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs: 方案 version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs: 元素 name="remoteCoA" type="coAResult"/>

  <xs:complexType name="coAResult">
    <xs:sequence>
      <xs: 元素 name=" 结果 " type="xs: 布尔值 " minOccurs="0"/>
    </xs:sequence>
    <xs: 属性 name="requestType" type="xs: 字符串 "/>
  </xs:complexType>
</xs:schema>
```

调用断开 API 呼叫

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，*https://<ISE 主机名或 IP 地址 >/admin/*）。

步骤 2　输入在 Cisco ISE 初始设置过程中指定和配置的用户名及密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

例如，当您最初记录到监控与主机名的 Cisco ESS 节点 - acme123 时，将显示此节点的以下 URL 地址：

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

步骤 4　在 URL 地址目标节点领域参与断开 API 呼叫通过替换“/admin/ ”组件使用的 API 呼叫组件（/admin/API/mnt/CoA/ <Disconnect>/<serverhostname>/<macaddress>/<portoptiontype>/<nasipaddress>/<destinationipaddress> ：

```
https://acme123/admin/API/mnt/CoA/Disconnect/server12/00:26:82:7B:D2:51/2/10.10.
10.10/192.168.1.1
```

注　因为这些呼叫区分大小写，您必须在 URL 地址目标节点领域仔细输入每个 API 调用。使用“安装”在 API 呼叫约定代表监控 ESS 节点的 Cisco。

步骤 5　按 **Enter** 发出 API 呼叫。

相关主题

- 验证监控节点，第 1-2 页

采样断开与 API 调用返回的数据

当您将在目标监视器 ESS 节点时，思科的断开 API 调用以下示例说明返回的数据。两种可能的结果可通过调用此命令返回：

- true 表明命令已成功执行。
- 错误表示命令并未执行（由于各种情况）。

此 XML 文件不会有任何样式信息与其关联。本文档树如下所示。

```
-
<remoteCoA requestType="next - ccode - and - reauth">
<results>true</results>
</remoteCoA>
```

第 2 部分

Cisco ISE 外部 RESTful 服务 API

ERS API 简介

概述

本章提供指南和示例用于支持的外部宁静的服务 API 和相关 API 呼叫。通过这些调用，您可以对 Cisco ISE 资源执行 CRUD（创建、读取、更新、删除）操作。

支持的 Cisco ISE 资源

Cisco ISE 外部宁静的服务允许您在 ESS 资源的以下类型的操作

- 终端设备
- 终端身份组
- 访客用户
- 身份组
- 内部用户

- 门户
- 分析器策略
- 网络设备
- 网络设备组
- 安全组

第 7 章“外部 RESTful 服务 API 操作”中提供完整的请求和响应示例。

外部宁静的服务 API 身份验证和授权

外部宁静的服务 API 根据 HTTPS 协议和其他方式和使用端口 9060。

外部宁静的服务 API 支持基本身份验证。身份验证凭证加密并是请求报头的一部分。

ESS 管理员分配种类到用户执行操作使用外部宁静的服务 API。使用外部宁静的服务 API， ESS 管理员可以指定以下两个角色执行操作

- 外部宁静的服务管理员为完全访问所有 ERS API （GET， POST，删除， PUT ）。
- 外部宁静的服务运营商为只读访问（请仅收到请求）。

使用外部宁静的服务 API，如果您没有所需的权限和仍不尝试执行操作，您将收到错误响应。

相关主题

- 创建一个新的 Cisco ISE 管理员
- 发起人身份验证和授权，第 6-1 页

通过 GUI 的外部宁静的服务 API

您必须启用 Cisco ISE REST API 对于 Cisco ISE 开发的应用 REST API 可以访问 Cisco ISE。Cisco REST API 使用 HTTPS 端口 9060，默认情况下会关闭。Cisco ISE REST API 在 Cisco ISE 管理员服务器上未启用，客户端应用程序从所有访客 REST API 请求的服务器将收到超时错误。

所有类型外部宁静的服务请求的主要 ESS 节点有效。辅助节点可以访问（GET 请求）。

操作步骤

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL（例如，`https://<ISE 主机名或 IP 地址>/admin/`）。

步骤 2　输入您的用户名和密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

步骤 4　选择**管理 > 设置 > ERS 设置**。

步骤 5　选择**启用读取 / 写入的 ERS** 主要管理节点的。

步骤 6　如果**有任何辅助节点**，请选择启用读取 ERS 为其他节点。

步骤 7　点击**提交**。

备注　所有其它操作进行审核，并记录登录系统日志。

相关主题

- 使用外部 RESTful 服务 API 调用的必备条件，第 7-2 页

外部 RESTful 服务 API 状态

您可以验证外部 RESTful API 服务是否为在 GUI 的**管理 > 设置 > ERS 设置的主要**和辅助节点启用页面。

默认情况下外部 RESTful API 服务未启用。如果您尝试调用 API 在启用之前呼叫的外部宁静的服务，您将收到错误响应。

外部宁静的服务 API 具有调试记录 class，您可以从 Cisco ISE GUI 的调试日志记录的页面启用。

备注　有关详细信息，请参阅思科身份服务引擎管理员指南，版本 1.4 的*调试日志配置选项*。

数据验证

CRUD 数据发送到服务器验证与 Cisco ISE 为 GUI 使用相同的验证规则。所有检验验证层集中。发布的所有 XML 数据经过验证方案。

验证的以下两种类型发生：数据验证和架构的验证。数据验证数据兼容的 ESS，例如，必填字段，字段长度，类型，等等。其中，架构的验证发生方案，例如，字段顺序，名称，等等。

命名空间

您必须维护在资源名称和 URI 中的强大的命名空间如下

- 内部用户、终端、终端组和身份组的身份
- SGA 的 SGT
- 其他对象资源的外部宁静的服务（如搜索结果（显示在回复消息的客户端需要发送在请求的 ERS 命名空间）

接受 / 目录类型的邮件头必须包含以下命名空间：

```
application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major
version>.<minor version>+xml
```

例如：应用 /vnd.com.cisco、ise.identity.internaluser.1.0+xml

请求 XML 应包含命名空间定义如下

```
identity.ers.ise、cisco.com

sga.ers.ise、cisco.com
```

例如：`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`

```
<ns: 端点 xmlns:ns="identity.ers.ise、cisco.com" id="id" >

<group>Profiled</group>

…

</ns:endpoint>
```

外部宁静的服务 SDK

您可以使用外部宁静的服务 SDK 开始迁移工具支持工具。您可以从以下 URL 访问外部宁静的服务 SDK：`https:// <ISE - ADMIN - NODE>:9060/ers/sdk`、外部宁静的服务 SDK 可以只通过外部宁静的服务管理员用户访问。SDK 包括以下组件

- 快速参考 API 文档
- 所有可用的 API 操作列表
- 架构文件可下载
- 在 Java 的示例应用程序可以下载
- 使用案例提供 curl 脚本格式
- 在使用 Chrome 邮差的说明

外部宁静的服务架构文件

外部宁静的服务 SDK 附带描述对象结构 ESS ERS 接口支持的三种 XSD 架构文件。

三个 XSD 文件如下

- ers.xsd
- identity.xsd
- sga.xsd
- network.xsd

您可以使用与可用工具的方案例如 JAXB 生成方案类别。

您可以开发 HTTP 客户端或采用第三方 HTTP 客户端代码和集成它与 XSD 文件生成的方案类别。

备注　在目录发送的 XML 验证方案，因此，字段顺序和语法在 XML 应是作为类似于方案的相同。否则，您将收到一份 Bad Request 状态代码。

下载架构文件

操作步骤

步骤 1　在 Web 浏览器中输入以下 URL 地址栏记录到外部宁静的服务 SDK 页面：`https:// <ISE - ADMIN - NODE>:9060/ers/sdk`

步骤 2　输入用户名和这个区分大小写的密码与外部宁静的服务管理员相应。

步骤 3　点击 **Login** 或按 **Enter**。

步骤 4　在**下载 (Downloads)** 类别中，点击**架构 Jar (ers-schema-2.0.0-306.jar) (Schema Jar [ers-schema-2.0.0-306.jar])**。

步骤 5　将文件保存到您的本地计算机。

相关主题

- 外部宁静的服务 API 身份验证和授权，第 5-2 页

外部宁静的服务请求和响应

本节在请求和响应报头提供信息以及 ERS 退货状态代码。

外部宁静的服务请求报头

表 5-1　　ERS 请求头

标头	支持的值	使用说明	必要
接受	访客 REST API 资源介质类型	向服务器指示此客户端愿意接受的介质类型，包括资源版本。	是，在 GET/GET ALL/DELETE/GET 版本操作（这些不包含邮件正文）
授权	“Basic”加上用户名和密码（根据 RFC 2617）	识别提交此请求的授权用户。	是，用于所有请求。
Content-Length	访客 REST API 资源介质类型	向服务器指示此客户端愿意接受的介质类型，包括资源版本。	是，在包含邮件正文的请求。
Content-Type	描述请求消息正文的介质类型	描述请求消息正文的表示和语法。	是，在包含邮件正文的请求。

外部宁静的服务响应报头

表 5-2　　ERS 强制响应头

标头	支持的值	使用说明	必要
Content-Length	长度（以字节为单位）的回复消息正文。	描述邮件正文的范围。	是，在包含邮件正文的响应。
Content-Type	描述响应消息正文的介质类型。	描述响应消息正文的表示和语法。	是，在包含邮件正文的响应。
位置	一种新创建的资源的规范 URI。	返回可用于请求这种新创建的资源的代表的新的 URI。	是，在创建新的服务器端资源可通过 URI 的请求的响应。

通用外部宁静的服务 HTTP 状态代码

表 5-3　　ERS 返回 HTTP 响应代码的说明

HTTP 状态	描述
200 OK	请求已成功完成。如果创建了可路由的带有 URI 和响应正文的新资源返回包含新的资源的代表的此请求，200 状态将返回包含这种新创建的资源的位置报头规范 URI。
201 创建	创建新资源的请求完成和包含新资源的表示无响应正文返回。还应返回包含这种新创建的资源的位置报头规范 URI。
202 接收	请求用于处理收到的，但是，处理未完成。每个该 HTTP/1.1 说明，返回的实体（如果有）应包括请求的当前状态的指示，和指针到状态监控或某些预估，当用户可以预期请求执行时。
204 无内容	服务器执行请求，但是，不需要返回回复消息正文。
400 Bad Request	请求无法处理，因为它包含缺失或无效信息（例如在输入字段，一个缺少所需的值的验证错误，等）。
401 未授权	身份验证凭证包含在此请求缺失或无效。
403 禁止	服务器意识到了您的凭证，但是，您不拥有权限执行此请求。
404 Not Found	请求中指定了不存在资源的 URI。
405 方法不允许	在请求中指定的 HTTP 动词（删除，获取，方向，发布，将）未为此请求 URI 中支持。
406 不可接受	此请求确定的资源不能够生成表示与之一在请求的接受报头的介质类型对应的。
409 冲突	创建或更新请求无法完成，因为它在服务器支持的资源将导致冲突（例如，尝试的当前状态使用唯一 ID 创建新资源已分配到某现有资源）。
415 Unsupported Media Type	服务器不支持在接收信头中指定的介质类型。当服务器，不再支持客户端资源版本将通用响应。
429 个大量请求	有多种并发 ERS 请求。
500 内部服务器错误	服务器遇到阻止其执行请求中的意外情况。
501 未实施	服务器（当前）不支持功能需要完成请求。
503 未提供的服务	服务器当前无法处理请求由于服务器的临时过载或维护。

备注　除了状态代码以外返还在响应报头，每个请求可能会根据请求的性质具有不同的 xml 目录。

具有外部宁静的服务 API 的版本控制

外部宁静的服务 API 提供向后兼容与以前的 Cisco ISE 版本。外部宁静的服务 API 具有 API 版本管理的一个版本的机制。所有非访客资源是第 1.0 版，无需向后兼容。

每个 RESTful 资源都具有一个模型版本 (major.minor)。版本必须包含在请求标头内，并采用如下语法：

```
application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml
```

例如，要获取内部用户资源版本 1.0，需要通过以下请求：

```
DELETE https://<ISE-ADMIN-NODE>:9060/ers/config/internaluser/333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
```

在身份验证和授权请求后，版本匹配检查执行以及提及的匹配结果在下表中的

版本匹配	结果
未发送版本	服务器返回状态 415“Unsupported Media Type”
客户端版本等于服务器版本	服务器继续处理请求。
客户端次要版本不等于服务器次要版本	服务器添加描述版本差异的响应警告消息，并继续处理请求。
客户端和服务器主要版本不匹配	服务器返回状态 415 以及相应的错误消息。

备注　每种资源有检索的 API 服务器支持版本列表。

搜索和过滤

所有过滤和搜索操作都通过使用过滤完成。

您可通过向资源 URI 发送 GET 请求搜索资源。默认情况下，结果为第一页（页面索引 = 0），默认大小为 20。通过向 URI 添加以下部分介绍的过滤器、排序和分页参数，客户端可以控制此搜索。

从分页、过滤器或排序请求生成的资源绑定在 `<resources>` 集合中，此集合包含每个资源的名称、ID、说明及其完整表示的链接。这使客户端能够轻松地向下展开到资源。

外部宁静的服务 API 的过滤参数

您可通过过滤器查询字符串参数执行简单的过滤的操作。您可以将多个过滤器。默认情况下逻辑运算符通用的所有过滤条件是和。通过使用“filtertype=or ”查询字符串参数，您可以更改此。

每个资源数据模型说明应指定属性是否为已过滤字段。

例如，具备开始从“a ”和属于身份组“财务”的名字的内部用户，以下请求通过：

```
获取 /ers/config/internaluser/ ?
page=0&size=20&sortacs=name&filter=name.STARTSW.a&filter=identityGroup.EQ.Finance
```

下表显示可用的过滤器运算符：

参数	说明
EQ	等于
GT	大于
LT	然后更少
STARTSW	开头为
ENDSW	结尾为
CONTAINS	包含

下表显示可过滤的属性列表每种资源的：

资源	可过滤属性
终端	Mac， portalUser，配置文件， profileId，staticGroupAssignment， staticProfileAssignment
内部用户	名称
终端身份组	名称
身份组	名称
SGT	名称

外部宁静的服务 API 的寻呼参数

所有外部宁静的 Servive 搜索结果中呼叫。使用查询参数，寻呼参数在 URI 通过。

例如，获取内部用户之前 20 记录进行排序按 accending 的顺序由“名称”字段，以下请求通过：

获取 `/ers/config/internaluser ? page=0&size=20&sortasc=name`

下表显示可用的呼叫的参数。

参数	说明	默认值
page	页面起始索引	0
size	页面大小	20（最大值 = 100）
sortasc	按升序对字段排序，从一组可用的排序字段中选择字段。（对于字母字符，按 A 到 Z 排序。）	名称
sortdsc	按降序对字段排序，从一组可用的排序字段中选择字段。（对于字母字符，按 Z 到 A 排序。）	名称

外部宁静的服务系统流

外部宁静的服务流的常见包括从客户端发送的 HTTPS 请求和 HTTPS 响应从服务器。流由请求类型、URI、请求报头、响应报头和响应内容不同。

图 5-1　　ERS 成功流量顺序

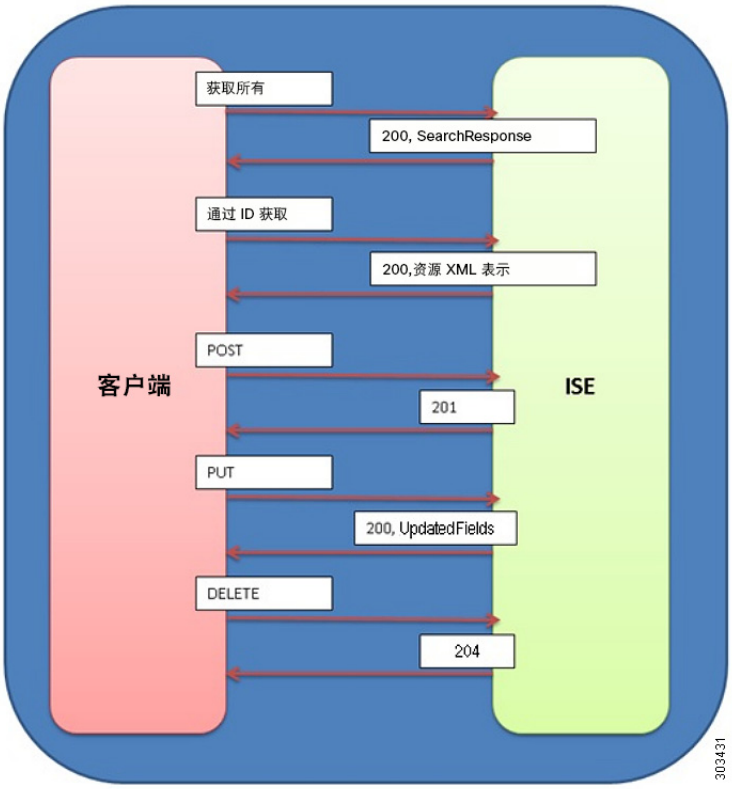
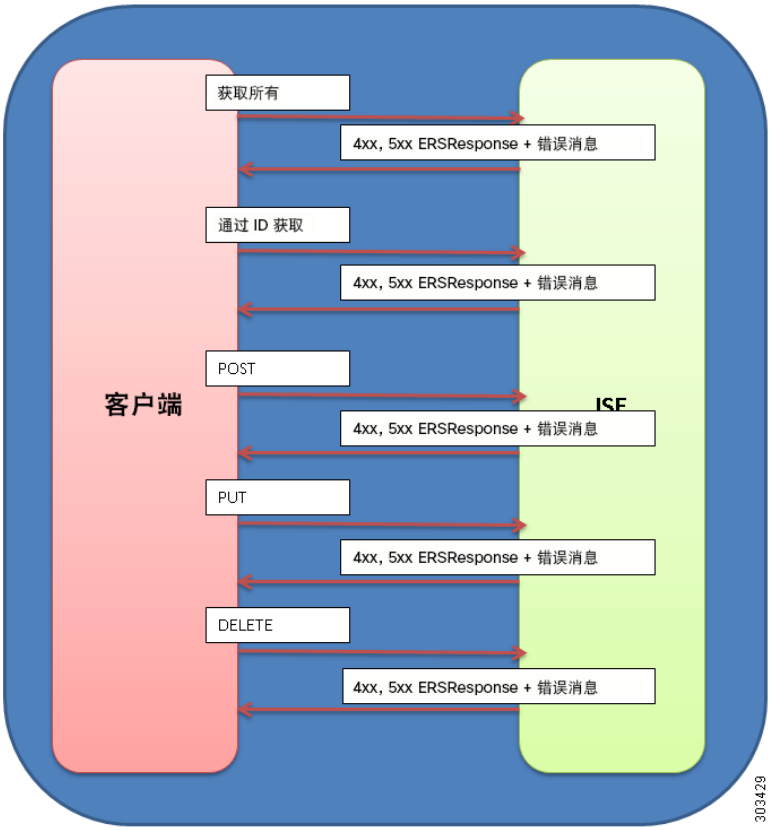


图 5-2　　ERS 故障流量顺序



超链接

超链接使用在 XML 中表示的是一个超媒体的最强的特征是引擎应用状态（HATEOS ）。超链接主要用于传达给客户端该资源有表示在特定 URI。

外部宁静的服务链路与以下命名空间宣布的链路的原子定义对齐：

http://www.w3.org/2005/Atom 命名空间。

下表显示外部宁静的服务的必填链结属性：

属性	说明
Href	这包含链路的 URI。
Rel	此属性，最初表示“关系”，表示链路的以下类型的 • “自” - 链路刷新当前表示 • “Next ” - 获取的下一页设置 • “信息” - 获取更多信息该资源
类型	这是提示对代表的介质类型服务器可以对链路的 URI 返回 - 通常是“应用 /xml ”

链路示例在搜索结果中的

以下示例显示在搜索结果中的链路：

```
<ns2:searchResult xmlns:ns2="ers.ise.cisco.com"   total="1163">
<link rel=" 自 "
href="http://cisco.com/ers/config/internaluser?page=0&size=20" type=" 应用 /xml"/>
<link rel=" 下 " href="http://cisco.com/ers/config/internaluser?page=20&size=20" type=" 应用
/xml"/>
<resources>
<link rel=" 三个 " href="http://cisco.com/ers/config/internaluser/333" type=" 应用 /xml"/>
<link rel="Jeff smit" href="http://cisco.com/ers/config/internaluser/444" type=" 应用
/xml"/>
.
.
.
</resources>
</ns2:searchResult>
```

批量操作

通过批量操作请求，您可以在单个请求中发送最多 500 项操作（或每个 ID 5000 项操作）。您可以运行创建，更新，删除操作所有资源以及与注册终端的一些资源特定操作。

请求中的所有操作必须具有相同类型，也就是说您不能发送混合资源请求。

每种资源都有自己的事务，并且由于是多线程执行，因此不保证事务的顺序。

Cisco ISE 服务器解析请求并验证其结构。如果请求有效且未进行其他批量操作，系统将开始执行，且服务器将返回状态代码 `202 (ACCEPTED)` 并在 LOCATION 响应头中显示唯一的批量标识符。通过此 ID，您稍后可以使用获取批量状态操作跟踪批量状态。您可以获取操作开始之后至少 2 小时的状态报告。

如果对资源执行操作时出现故障，故障将记录在状态报告中，系统会继续执行到下一个资源。

一次只允许执行一个批量操作。如果在另一个批量操作仍在执行时发布批量操作请求，服务器将返回响应状态 `503 (Service Unavailable)`，并包含要求客户端稍后重试的消息。

访客 REST API

用于访客用户资源的 API

思科访客 API 是基于 REST（表述性状态转移）的操作集合，提供对思科访客用户进行经过身份验证的安全访问及管理功能。利用此 API，您可以创建、读取、更新、删除和搜索访客用户。

当调用此 API 时，您将作为使用 Cisco ISE 发起人门户的发起人，调用 API 来管理访客用户。要使用此 API，您必须先在 ISE 中启用相应的访问权限，然后设置发起人身份验证。

适用于访客用户的外部 RESTful 服务 API，第 7-24 页中提供完整的请求和响应示例。

发起人身份验证和授权

发起人是特殊类型的 Cisco ISE 用户，可以使用发起人门户创建并管理访客用户。访客 REST API 具有与发起人门户相同的功能。访客 REST API 的身份验证与发起人进行身份验证的过程类似。在发起人组的策略中指定的权限适用于访客 REST API。

备注 使用访客 REST API 的发起人不能属于 ERS-ADMIN 角色。

有关发起人和发起人组的详细信息，请参阅《Cisco ISE 用户指南》。

准备工作

与其他 ISE 用户相似，Cisco ISE 通过本地数据库或者外部轻量级目录访问协议 (LDAP) 或 Microsoft Active Directory 身份库对发起人进行身份验证。如果您不打算使用外部身份库，则必须在 Cisco ISE 中创建用户 (**Administration > Identity Management > Identities > Users**)。

操作步骤

步骤 1　在浏览器的地址栏内输入 Cisco ISE URL （例如， `https://<ISE 主机名或 IP 地址 >/admin/`）。

步骤 2　输入您的用户名和密码（区分大小写）。

步骤 3　点击 **Login** 或按 **Enter**。

步骤 4　将用户分配到适当的身份组。

a. 选择 **Administration > Identity Management > Groups > Identity Groups**。

b. 创建新组或编辑现有组。Cisco ISE 包括以下默认发起人用户身份组：

- SponsorAllAccount
- SponsorGroupAccounts
- SponsorOwnAccounts

c. 将用户添加到成员列表中。

d. 点击**保存**。

步骤 5　将用户身份组添加到发起人组中。

a. 选择 **Guest Access > Configure > Sponsor Groups**。

b. 创建新发起人组或编辑现有发起人组。

c. 点击 **Members**。

d. 将用户的身份组添加到发起人组成员列表中，然后点击 **OK**。

e. 添加用户可以发起的访客类型和位置。

f. 选中 **Access ISE guest accounts via the programatic interface (REST API)** 复选框。

g. 点击**保存**。

步骤 6　确保 Sponsor_Portal_Sequence 可以访问用户的身份源。

a. 选择 **Administration > Identity Management > Identity Source Sequences > Sponsor_Portal_Sequence**。

b. 如果没有为 Authentication Search List 选择用户的身份源，请选择用户的身份源。

c. 点击**保存**。

相关主题

有关发起人和发起人组的详细信息，请参阅《Cisco ISE 用户指南》。

访客 REST API 请求

对 API 执行的请求具有以下特征：

- 用户请求通过 HTTPS 发送到 Cisco ISE 服务器，然后返回响应。
- 访问所有 API 的 URI 是主节点的域名 `https://<ISE-ADMIN-NODE>:9060/ers/config`
- API 请求区分大小写，应按照本手册所示进行输入。
- 每个访客请求需要将 Content Type 设置为：

```
application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

- 每个访客请求需要将 Accept 设置为：

  `application/vnd.com.cisco.ise.identity.guestuser.2.0+xml`
- 每个访客批量操作请求需要将 Content Type 设置为：

  `application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml`
- 每个访客批量操作请求需要将 Accept 设置为：

  `application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml`
- 您还可以包括可选的 `Accept-Encoding:gzip` 标头，用于压缩响应负载。

请求结构

下表列出可与思科访客 REST API 结合使用的请求操作类型：

表 6-1　　请求方法类型

请求类型	运营
GET	获取所有资源（搜索）、根据资源 ID 获取某项资源，以及获取资源版本信息。
POST	创建新访客用户。
PUT	更新访客用户。
DELETE	删除访客用户

下表列出请求的强制性标头：

表 6-2　　强制性请求标头

标头	价值观	描述
ACCEPT	访客 REST API 资源介质类型	向服务器指示此客户端愿意接受的介质类型，包括资源版本。
AUTHORIZATION	“Basic”加上用户名和密码（根据 RFC 2617）	识别提交此请求的授权用户。
CONTENT-TYPE	描述请求消息正文的介质类型	描述请求消息正文的表示和语法。

相关主题

Content Type 和 Accept 标头，第 7-25 页

请求内容

以下 XML 内容显示访客用户帐户的结构，包括自定义字段：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:guestuser name="guestUser" id="123456789" description="ERS Example user "
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <customFields>
        <entry>
            <key>some key</key>
            <value>some value</value>
        </entry>
        <entry>
```

```
            <key>another key</key>
            <value>and its value</value>
        </entry>
    </customFields>
    <guestInfo>
        <emailAddress>email@some.uri.com</emailAddress>
        <enabled>true</enabled>
       <phoneNumber>3211239034</phoneNumber>
        <smsServiceProvider>GLobal Default</smsServiceProvider>
        <userName>DS3ewdsa34wWE</userName>
    </guestInfo>
    <guestType>Contractor</guestType>
    <portalId>23423432523</portalId>
    <sponsorUserName>Mr Spons</sponsorUserName>
</ns3:guestuser>
```

以下请求示例显示有关使用 POST（创建）操作创建访客用户帐户所需的 XML 内容。

备注 创建访客用户帐户所需的字段无需与发起人门户中显示的必填字段对应。但是，如果发起人门户不提供创建访客用户帐户所需的信息，访客 REST API 会引发异常。

POST 请求示例

```
POST https://<ISE-Admin-Node>:9060/ers/config/guestuser/

Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Content-Type: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Authorization: Basic xxxxxxxxxxxxxxxx

<?xml version="1.0" encoding="UTF-8"?>
<ns2:guestuser xmlns:ns2="identity.ers.ise.cisco.com">
    <guestAccessInfo>
        <fromDate>08/06/2014 23:22</fromDate>
        <toDate>08/07/2014 23:22</toDate>
        <validDays>1</validDays>
    </guestAccessInfo>
    <guestInfo>
        <company>New Company</company>
        <emailAddress>john@example.com</emailAddress>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <notificationLanguage>English</notificationLanguage>
       <phoneNumber>9999998877</phoneNumber>
        <smsServiceProvider>Global Default</smsServiceProvider>
        <userName>autoguestuser1</userName>
    </guestInfo>
    <guestType>Daily</guestType>
    <personBeingVisited>sponsor</personBeingVisited>
    <portalId>portal101</portalId>
    <reasonForVisit>interview</reasonForVisit>
</ns2:guestuser>
```

备注 JSON 不用于访客 REST API。

访客密码

创建访客时，ISE 自动生成密码。可以使用访客 REST API 通过调用 `resetpassword` 操作重置访客的密码。

您无法使用此 REST API 将访客的密码更改为指定字符串。

使用 GET 操作可检索访客用户的信息以及查看其密码。只要密码符合以下条件，就可以在 GET 操作的响应中显示 Cisco ISE 访客密码：

1. 由 ISE 自动生成。
2. 通过此 API 或发起人门户重置。

在有些访客流中，访客能够更改自己的密码。访客更改过的 Cisco ISE 访客密码在发起人门户中不可见，也无法通过 REST API 显示。

相关主题

- 重置访客用户帐户的密码，第 7-38 页
- 有关访客用户密码策略的更多详细信息，请参阅《Cisco ISE 管理员指南》。

批量执行

通过批量操作请求，您可以在单个请求中发送最多 500 项操作，或根据 ID 发送最多 5000 项操作。

请求中的所有操作必须具有相同类型，也就是说您不能发送混合资源请求。

每种资源都有自己的事务，并且由于是多线程执行，因此不保证事务的顺序。

Cisco ISE 服务器解析请求并验证其结构。如果请求有效且未进行其他批量操作，系统将开始执行，且服务器将返回状态代码 `202 (ACCEPTED)` 并在 LOCATION 响应头中显示唯一的批量标识符。通过此 ID，您稍后可以使用获取批量状态操作跟踪批量状态。您可以获取操作开始之后至少 2 小时的状态报告。

如果对资源执行操作时出现故障，故障将记录在状态报告中，系统会继续执行到下一个资源。

一次只允许执行一个批量操作。如果在另一个批量操作仍在执行时发布批量操作请求，服务器将返回响应状态 `503 (Service Unavailable)`，并包含要求客户端稍后重试的消息。

批量执行（包括获取批量状态）使用不同的标头：

- Content Type 为：`application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml`
- Accept 为：`application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml`

相关主题

- 启动访客用户的批量执行，第 7-39 页

访客 REST API 响应

每个请求随后都会收到服务器 HTTPS 响应，此响应包括标准标头和响应内容。

响应状态代码

响应包含一个 HTTP 状态代码。对于访客 REST API，这些代码可以是下列代码之一：

- 20x - 操作成功
- 4xx - 客户端错误
- 5xx - 服务器错误

下表提供有关状态代码的更多信息。

表 6-3　响应内容类型

请求类型	状态	响应负载
根据访客用户的 ID 获取访客用户	200	XML 表示的访客用户
全部	4xx（客户端错误）	响应包含描述错误的消息，例如：版本不受支持或非法 URI
全部	5xx（服务器错误）	响应包含描述错误的消息，例如，运行时异常 ...
创建 A 访客用户 [POST]	201	如果没有信息或警告，不会发回任何内容。新访客用户 ID 将在响应的“Location”标头中发送。如果还有其他信息或警告，将封装在响应中。
更新访客用户 [PUT]	200	返回更新字段的列表。
删除访客用户 [Delete]	204	无内容。

相关主题

响应错误消息，第 6-7 页

响应结构

下表列出响应的标头。

表 6-4　强制性响应标头

标头	价值观	说明
LOCATION	新建资源的 ID	仅适用于 POST - 获取有新资源 ID（以 URI 表示）
CONTENT-TYPE	描述响应消息正文的介质类型	描述响应消息正文的表示和语法

响应错误消息

您在执行 POST 或 PUT 请求时，需要使用特定标头。如有遗漏，您会收到错误消息。

在 Cisco ISE UI 的以下位置可以找到包含详细消息的日志文件，您可以启用这些日志文件以更正操作：

Operations > Node > Debug Logs > guest.log

表 6-5　　错误代码

错误代码	说明	HTTP 状态
资源版本异常	服务器不支持请求内容中发送的资源版本。	415
资源介质类型异常	客户端在 ACCEPT 或 Content-Type 标头中发送的介质类型无效。	415
不受支持的资源异常	服务器不支持 URI 中列出的资源。	400
不受支持的方法异常	指定 URI 不支持请求方法类型。	400
查询字符串验证异常	搜索过滤器或分页参数无效。	400
架构验证异常	请求的 XML 内容不符合资源的 API 架构规则，例如包含 API 无法识别的字段或不包含必填字段。	400
应用资源验证异常	API 无法验证某些请求的内容，例如属性值中使用了无效字符。	400
未授权用户	发起人的用户名和密码无效。	401
内部异常	运行时发生的所有内部服务器意外错误。	500
转换异常	一些内部转换，应视为内部异常。	500
CRUD 操作异常	执行 CRUD 操作，应视为内部异常。	500

不支持的介质类型示例

以下示例展示的是客户端在 ACCEPT 标头中发送不受支持的介质类型时出现的响应。

响应

```
STATUS 415 Unsupported Media Type
Cache-Control: no-cache
Content-Length: 411
Content-Type: application/vnd.com.cisco.ise.ers.ersresponse.1.0+xml
Date: Wed, 11 Dec 2013 05:27:37 GMT
Expires: Wed, 31 Dec 2015 16:00:00 PST
Pragma: No-cache

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ersResponse
    xmlns:ns2="ers.ise.cisco.com" operation="GET-getAll-guestuser">
    <link type="application/xml"
href="https://<ISE-Admin-Node>:9060/ers/config/guestuser/" rel="related"/>
```

```
    <messages>
        <ns2:message type="ERROR" code="Resource media type exception">
            <title>Wrong media type, check Accept request header.</title>
        </ns2:message>
    </messages>
</ns2:ersResponse>
```

版本

Cisco ISE 版本 1.4 的访客 REST API 是 2.0 版本。它可与未来版本兼容。

此 REST API 与早期 alpha 版本和 beta 版本的访客 REST API 不兼容。

每个 RESTful 资源都具有一个模型版本 (major.minor)。版本必须包含在请求标头内，并采用如下语法：

```
application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml
```

例如，要获取访客用户资源版本 2.0，需要通过以下请求：

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

对请求进行身份验证和授权后，系统将执行版本匹配检查，并具有下列匹配结果之一：

表 6-6　　版本匹配结果

版本匹配	结果
未发送版本	服务器返回状态 `415` “`Unsupported Media Type`”
客户端版本等于服务器版本	服务器继续处理请求。
客户端次要版本不等于服务器次要版本	服务器添加描述版本差异的响应警告消息，并继续处理请求。
客户端和服务器主要版本不匹配	服务器返回状态 `415` 以及相应的错误消息。

搜索和过滤

所有过滤和搜索操作都通过使用过滤完成。

您可通过向资源 URI 发送 GET 请求搜索资源。默认情况下，结果为第一页（页面索引 = 0），默认大小为 20。通过向 URI 添加以下部分介绍的过滤器、排序和分页参数，客户端可以控制此搜索。

从分页、过滤器或排序请求生成的资源绑定在 `<resources>` 集合中，此集合包含每个资源的名称、ID、说明及其完整表示的链接。这使客户端能够轻松地向下展开到资源。

过滤参数

过滤功能可通过过滤器查询字符获取。过滤器的结构是字段运算符、参数和值三者的组合，以圆点分隔。例如，使用 `filter=name.STARTSW.g` 查找用户名以“g”开头的所有访客用户。

如果您在过滤器中使用多个参数，结果为对这些参数进行 AND 运算的结果。这意味着结果是与发送到 API 的所有参数匹配的用户。每个资源说明指定属性是否为已过滤字段。

备注　无效的过滤器值将生成状态 `400 (Bad Request)` 并显示相应的消息。

下表显示过滤器查询中可用的参数。

表 6-7　过滤参数

参数	说明
firstName	访客的名字
lastName	访客的姓氏
emailAddress	访客的邮件地址
userName	访客帐户用户名
creationTime	访客帐户的创建时间
toDate	访客帐户的到期日期
guestType	访客的类型
company	访客的公司
phoneNumber	访客的电话号码
groupTag	组
sponsor	访客的发起人
status	访客帐户的状态
名称	资源标识符

下表列出可以在过滤器查询中使用的操作。

表 6-8　可用的过滤器操作

参数	说明
EQ	等于
GT	大于
LT	小于
STARTSW	开头为
ENDSW	结尾为
CONTAINS	包含

过滤示例

使用 GET 请求获取名字的开头为“br”的访客用户

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/?filter=firstName.STARTSW.br
```

使用 GET 请求获取名字的开头为“b”且邮件地址包括“bob”的访客用户

```
GET
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser?filter=firstName.STARTSW.b&filter=email
Address.CONTAINS.bob
```

使用 GET 请求获取状态为“Approved”且到期日期为十月份的访客用户

```
GET
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser?filter=status.EQ.Approved&filter=toDate
.STARTSW.10
```

相关主题

获取访客用户，第 7-25 页

页面大小参数

搜索结果默认为每页 20 项资源。页面编号从“0”页开始。每页的最大资源数不能超过 100。非法参数值将产生状态 `400 (Bad Request)` 以及相应的消息。

通过使用表 6-9 中介绍的分页参数，页面大小参数可以覆盖默认值。

在下面的示例中，页面大小更改为 50 项资源：

```
GET
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser?filter=name.STARTSW.b&filter=emailAddre
ss.CONTAINS.bob&size=50&page=0
```

排序参数

默认情况下，排序列是 `name`，排序方向是 `sortasc`。您可以按照下例所示使用参数覆盖默认值：

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser?size=50&page=0&sortdsc=name
```

您可以在表 6-7，第 6-9 页中找到可用于排序的值列表。

表 6-9　　可用的分页和排序首选项

参数	说明	默认值
page	页面起始索引	0
size	页面大小	20（最大值 = 100）
sortasc	按升序对字段排序，从一组可用的排序字段中选择字段。（对于字母字符，按 A 到 Z 排序。）	名称
sortdsc	按降序对字段排序，从一组可用的排序字段中选择字段。（对于字母字符，按 Z 到 A 排序。）	名称

示例：获取前 20 个访客用户记录并根据姓氏按升序排序

请求

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser?page=0&size=20&sortasc=lastName
```

响应

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<ns2:searchResult
    xmlns:ns2="ers.ise.cisco.com" total="22">
    <nextPage type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser?page=1&size=20&sortasc=lastName"
rel="next"/>
    <resources>
        <resource name="aname001" id="8e4bf290-6229-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-Admin-Node:9060/ers/config/guestuser/8e4bf290-6229-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="aname002" id="8fe86480-6229-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-Admin-Node:9060/ers/config/guestuser/8fe86480-6229-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
...
        <resource name="aname020" id="908b5b40-6229-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-Admin-Node:9060/ers/config/guestuser/908b5b40-6229-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
    </resources>
</ns2:searchResult>
```

相关主题

- 通过开头为“ilucky”的用户名进行过滤的示例，第 7-26 页
- 通过开头为“ilucky”的用户名和开头为“J”的姓氏进行过滤的示例，第 7-27 页
- 通过名字“John”进行过滤并按用户名进行排序的示例，第 7-28 页

外部 RESTful 服务 API 操作

概述

本章提供有关外部 RESTful 服务 API 调用的示例，并介绍其使用方法。此外，本章还提供对发出外部 RESTful 服务 API 调用、API 输出架构文件示例以及返回的样本数据的说明。

使用外部 RESTful 服务 API 调用的必备条件

您必须满足以下必备条件，才能发起外部 RESTful 服务 API 调用：

- 您必须已通过 GUI 启用外部 RESTful 服务。
- 您必须具有外部 RESTful 服务管理员权限。

您可以使用 REST 客户端（如 JAVA）、curl linux 命令、python 或任何其他客户端来调用外部 RESTful 服务 API 调用。

相关主题

- 通过 GUI 的外部宁静的服务 API（第 5-2 页）
- 外部宁静的服务 API 身份验证和授权（第 5-2 页）

GetVersion

GetVersion 操作对所有可用资源是通用的。它可获取所需资源的版本信息。下表列出此操作的主要特征：

表 7-1　GetVersion 操作的主要特征

说明	检索指定资源的版本信息
摘要	GET/ers/config/<resource-name>/versioninfo
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	版本信息
响应状态	200，400，401，403，404，415，500

GetVersion 操作的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/<resource-type>/versioninfo
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.<resource-namespace>.1.0+xml
```

GetVersion 操作的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.<resource-namespace>.versoininfo.1.0+xml
Content-Length: 122
{
<?xml version="1.0" encoding="UTF-8"?>
<ns2:versionInfo xmlns:ns2="ers.ise.cisco.com">
    <currentServerVersion>1.2</currentServerVersion>
    <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/<resource-type>/versioninfo" rel="self"/>
    <supportedVersions>1.0,1.1,1.2,1.3<supportedVersion>
</ns2:versionInfo>
}
```

适用于内部用户的外部 RESTful 服务 API

适用于内部用户的外部 RESTful 服务 API 支持完整的 CRUD 功能。下表列出适用于内部用户的外部 RESTful 服务 API：

表 7-2　适用于内部用户的外部 RESTful 服务 API

操作	HTTP 方法	URL	内容	查询字符串
获取所有用户	GET	/ers/config/internaluser	不适用	Page、Size、sortacs 或 sortdsn、Filter
获取用户	GET	/ers/config/internaluser/{internal user-id[1]}	不适用	
创建用户	POST	/ers/config/internaluser/	internaluser	
更新用户	PUT	/ers/config/internaluser/{internal user-id}	internaluser	
删除用户	DELETE	/ers/config/internaluser/{internal user-id}	不适用	
获取内部用户资源版本信息	GET	/ers/config/internaluser/version	不适用	

1. 内部用户 ID 是 Cisco ISE 数据库中存储的 UUID 类型。

检索所有内部用户

您可以使用此 API 调用检索 Cisco ISE 中存在的所有内部用户。下表列出此 API 调用的主要特征：

表 7-3　检索所有内部用户 API 调用的主要特征

描述	检索内部用户的集合
摘要	GET /ers/config/internaluser
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，429，500

检索所有内部用户 API 的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/internaluser?page=0&size=20&sortacs=name
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
```

检索所有内部用户 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <resources>
        <ns2:resource description="description1" name="name1" id="id1"/>
        <ns2:resource description="description2" name="name2" id="id2"/>
    </resources>
</ns2:searchResult>
}
```

通过 ID 获取内部用户

您可以使用此 API 调用通过 ID 获取 Cisco ISE 中的内部用户。下表列出此 API 调用的主要特征：

表 7-4　　读取内部用户 API 调用的主要特征

说明	检索指定的内部用户
摘要	GET /ers/config/internaluser/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	200，400，401，403，404，415，429，500

读取内部用户 API 的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/internaluser/333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.internaluser.1.0+xml
```

读取内部用户 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser description="description" name="name" id="id"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
```

```
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@example.com</email>
    <enabled>true</enabled>
    <firstName>John</firstName>
    <identityGroups>83fb7800-86e0-11e5-800e-005056941420</identityGroups>
    <lastName>Doe</lastName>
    <password>12345</password>
</ns3:internaluser>
}
```

备注　*identityGroups* 属性表示身份组 ID。

创建内部用户

您可以使用此 API 调用在 Cisco ISE 中创建内部用户。使用外部 RESTful 服务 API 创建内部用户时，密码为必填内容。下表列出此 API 调用的主要特征：

表 7-5　　创建内部用户 API 调用的主要特征

说明	创建指定的内部用户
摘要	POST /ers/config/internaluser/
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	内部用户
响应头	Content-Length、Content-Type、Location
响应消息正文	内部用户类型的资源
响应状态	201，400，401，403，415，429，500

创建内部用户 API 的请求示例

```
POST https://<ISE-ADMIN-NODE>:9060/ers/config/internaluser/
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser description="description" name="name" id="id"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@example.com</email>
```

```
    <enabled>true</enabled>
    <firstName>John</firstName>
    <identityGroups>83fb7800-86e0-11e5-800e-005056941420</identityGroups>
    <lastName>Doe</lastName>
    <password>12345</password>
</ns3:internaluser>
}
```

备注　*customAttributes* 对于创建内部用户是必需的。至少必须包含根元素：<customAttributes/>

备注　*identityGroups* 属性表示身份组 ID。

创建内部用户 API 的响应示例

```
HTTP/1.1 201 OK  (see the location header for the new user's ID)
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
Location: https://<ISE-ADMIN-NODE>/ers/config/internaluser/444
```

更新内部用户

您可以使用此 API 调用更新 Cisco ISE 中的内部用户。如果在使用外部 RESTful 服务 API 更新内部时不更改密码，您必须将密码设置为“********”。下表列出此 API 调用的主要特征：

表 7-6　　更新内部用户 API 调用的主要特征

说明	更新指定的内部用户
摘要	PUT /ers/config/internaluser/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	内部用户
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，404，415，429，500

更新内部用户 API 的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/internaluser/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser description="description" name="name" id="333"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
```

```
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@example.com</email>
    <enabled>true</enabled>
    <firstName>John</firstName>
    <identityGroups>83fb7800-86e0-11e5-800e-005056941420</identityGroups>
    <lastName>Doe</lastName>
    <password>********</password>
</ns3:internaluser>
}
```

备注　*identityGroups* 属性表示身份组 ID。

更新内部用户 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
    <updatedField field="lastname">
        <newValue>Doe</newValue>
        <oldValue>name</oldValue>
    </updatedField>
    <updatedField field="identityGroups">
        <newValue>identityGroups</newValue>
        <oldValue>zzz</oldValue>
    </updatedField>
</ns2:updatedFields>
}
```

删除内部用户

您可以使用此 API 调用删除 Cisco ISE 中的内部用户。下表列出此 API 调用的主要特征：

表 7-7　　删除内部用户 API 调用的主要特征

说明	删除指定的内部用户
摘要	DELETE /ers/config/internaluser/{internaluser-id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	204，400，401，403，404，415，429，500

删除内部用户 API 的请求示例

```
DELETE https://<ISE-ADMIN-NODE>:9060/ers/config/internaluser/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
```

删除内部用户 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

适用于终端的外部 RESTful 服务 API

下表列出适用于终端的外部 RESTful 服务 API：

表 7-8　　适用于终端的外部 RESTful 服务 API

操作	方法	URL	内容	查询字符串
获取所有终端	GET	/ers/config/endpoint	不适用	page、size、sortacs 或 sortdsn、filter
获取终端	GET	/ers/config/endpoint/{id[1]}	不适用	
创建终端	POST	/ers/config/endpoint/	终端	
更新终端	PUT	/ers/config/endpoint/{id}	终端	
删除终端	DELETE	/ers/config/endpoint/{id}	不适用	
注册终端	PUT[2]	/ers/config/endpoint/register	终端	
撤销注册终端	PUT	/ers/config/endpoint/{id}/deregister	不适用	
获取终端资源版本信息	GET	/ers/config/endpoint/version	不适用	

1. 终端 ID 是 Cisco ISE 数据库中存储的 UUID 类型。
2. 如果终端已经存在，则会执行终端注册。如果终端不存在，则会先创建终端，再执行终端注册。在这两种情况下，返回的状态均为 204。

获取所有终端

获取所有终端的 API 仅用于检索与过滤器中指定的用户关联的终端。下表列出此 API 调用的主要特征：

表 7-9　　获取所有终端 API 调用的主要特征

说明	检索与指定内部用户关联的终端的集合
摘要	GET /ers/config/endpoint/
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A

表 7-9　获取所有终端 API 调用的主要特征（续）

说明	检索与指定内部用户关联的终端的集合
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，429，500

获取所有终端 API 的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint?filter=userid.EQ.123
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
```

获取所有终端 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
<resources>
<resource name=name1" id="id1">
   <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/id1" rel="self"/>
  </resource>
  <resource name="name2" id="id2">
   <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/id2" rel="self"/>
  </resource>
</resources>
</ns2:searchResult>
}
```

通过 ID 获取终端

您可以使用此 API 调用通过 ID 获取 Cisco ISE 中的终端。下表列出此 API 调用的主要特征：

表 7-10　读取终端 API 调用的主要特征

说明	检索指定的终端
摘要	GET /ers/config/endpoint/{endpoint-id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	200，400，401，403，404，415，429，500

读取终端 API 的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
```

读取终端 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint description="description" name="name" id="id" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
    <group>group</group>
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profile>profile</profile>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
}
```

创建终端

您可以使用此 API 调用在 Cisco ISE 中创建终端。下表列出此 API 调用的主要特征：

表 7-11　创建终端 API 调用的主要特征

描述	创建指定的终端
摘要	POST /ers/config/endpoint/
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	200，400，401,403，404，415，429，500

创建终端 API 的请求示例

```
POST https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint description="description" name="name" id="id" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
```

```
    <group>group</group>
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profile>profile</profile>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
}
```

创建终端 API 的响应示例

```
HTTP/1.1 201 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
Location: https://cisco.com/ers/config/endpoint/444
```

更新终端

您可以使用此 API 调用更新 Cisco ISE 中的终端。下表列出此 API 调用的主要特征：

表 7-12　　更新终端 API 调用的主要特征

说明	更新指定的终端
摘要	PUT /ers/config/endpoint/{endpoint-id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	200，400，401，403，404，415，429，500

更新终端 API 的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:endpoint xmlns:ns2="identity.ers.ise.cisco.com" description="updated"
name="Endpoint">
    <mac>AA:AA:AA:AA:AA:AA</mac>
    <staticGroupAssignment>false</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns2:endpoint>}
```

更新终端 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
<updatedField field="mac">
  <newValue>AA:AA:AA:AA:AA:AA</newValue>
  <oldValue>BB:BB:BB:BB:BB:BB</oldValue>
 </updatedField>
 <updatedField field="staticGroupAssignment">
  <newValue>false</newValue>
  <oldValue>true</oldValue>
 </updatedField>
<updatedField field="staticProfileAssignment">
  <newValue>false</newValue>
  <oldValue>true</oldValue>
 </updatedField>
</ns2:updatedFields>
}
```

删除终端

您可以使用此 API 调用更新 Cisco ISE 中的终端。下表列出此 API 调用的主要特征：

表 7-13　　删除终端 API 调用的主要特征

说明	删除指定的终端
摘要	DELETE /ers/config/endpoint/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	200，400，401，403，404，415，429，500

删除终端 API 的请求示例

```
DELETE https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
```

删除终端 API 的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

注册终端

您可以使用此 API 调用注册 Cisco ISE 中的终端。如果终端尚不存在，将创建终端。与 GUI 注册流程类似，终端静态分配给已注册的设备组，并按照内容中指定的值设置门户用户和身份库。

下表列出此 API 调用的主要特征：

表 7-14　　注册终端 API 调用的主要特征

说明	注册指定的终端
摘要	PUT /ers/config/endpoint/register
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	终端
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	202，400，401，403，404，415，429，500

注册终端 API 的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/register
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint description="description" name="name" id="id" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
    <group>group</group>
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profile>profile</profile>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
}
```

注册终端 API 的响应示例

```
HTTP/1.1 204 No Content
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

撤销注册终端

您可以使用此 API 调用对 Cisco ISE 中的终端撤销注册。结果中不显示任何内容。下表列出此 API 调用的主要特征：

表 7-15　撤销注册终端 API 调用的主要特征

说明	撤销注册指定的终端
摘要	PUT /ers/config/endpoint/{id}/deregister
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应消息正文	N/A
响应状态	202，400，401，403，404，415，429，500

撤销注册终端 API 调用的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/123/deregister
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
```

撤销注册终端 API 调用的响应示例

```
HTTP/1.1 204 No Content
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

启动终端批量执行

通过批量执行，您可以在单个请求中发送最多 500 项类型相同的 CRUD 操作。

如果请求有效，服务器将返回状态代码 `202 (ACCEPTED)`，并在 LOCATION 响应头中包含唯一的批量标识符，您可以使用此标识符通过获取批量状态操作跟踪批量状态。

一次只允许运行一个批量操作。如果在另一个批量操作仍在执行时发布批量操作请求，服务器将返回响应状态 `503 (Service Unavailable)`，并包含相应的要求客户端稍后重试的描述性消息。

表 7-16　启动终端批量执行的主要特征

说明	启动执行
摘要	PUT /ers/config/endpoint/bulk
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	批量请求
响应头	Content-Length、Content-Type
响应消息正文	不适用
响应状态	202，400，401，403，404，415，500

启动终端批量执行 API 调用的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/bulk HTTP/1.1
Host: {some-ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx Content-Type:
application/vnd.com.cisco.ise.ers.endpointbulkrequest.1.0+xml
{
<ns3:endpointBulkRequest
resourceMediaType = "vnd.com.cisco.ise.ers.identity.endpoint.1.0+xml" operationType =
"create"
xmlns:ns2 = "ers.ise.cisco.com"
xmlns:ns3 = "identity.ers.ise.cisco.com">
<resourcesList> <resource
xsi:type = "ns3:ersEndPoint"
description = "created by bulk request"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"> <mac>11:22:33:44:55:66</mac>
<staticGroupAssignment>false</staticGroupAssignment>
<staticProfileAssignment>false</staticProfileAssignment>
</resource>
...
<resource
xsi:type = "ns3:ersEndPoint"
description = "created by bulk request"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"> <group>Profiled</group>
<groupId>804f5350-7808-11e2-bdd0-0050568e01f0</groupId> <identityStore></identityStore>
<identityStoreId></identityStoreId> <mac>11:22:33:44:55:77</mac>
<portalUser></portalUser>
<profile>Apple-iPod</profile> <profileId>b8128870-7808-11e2-bdd0-0050568e01f0</profileId>
<staticGroupAssignment>true</staticGroupAssignment>
<staticProfileAssignment>true</staticProfileAssignment>
</resource> </resourcesList>
</ns3:endpointBulkRequest>
}
```

启动终端批量执行 API 调用的响应示例

```
HTTP/1.1 202 ACCEPTED
        Date: Thu, 12 Jul 2012 23:59:59 GMT
        Location: https://ise-node-ip:9060/ers/config/endpoint/123443545334
```

获取终端批量状态

如果批量执行请求有效且没有其他正在处理的批量操作，服务器将在 LOCATION 响应头中返回唯一的批量标识符。使用此 ID 可跟踪批量状态。您可以获取操作开始之后至少 2 小时的状态报告。

表 7-17　　获取批量状态的主要特征

说明	监控指定的批量执行进度
摘要	GET /ers/config/endpoint/bulk/{bulkid}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	批量状态
响应状态	200，400，401，403，404，415，500

获取终端批量状态示例

请求

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/bulk/53454354534 HTTP/1.1
    Host: {some-ise-node-ip}
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.bulkStatus.1.0+xml
```

响应

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.bulkStatus.1.0+xml Content-Length: 16347
{
<ns2:bulkStatus
           xmlns:ns2 = "ers.ise.cisco.com"
           successCount = "750"
           startTime = "Thu Mar 07 17:17:35 IST 2013"
           resourcesCount = "750"
           operationType = "create"
           mediaType =
    "vnd.com.cisco.ise.ers.identity.endpoint.1.0+xml"
           failCount = "0"
           executionStatus = "COMPLETED"
           bulkId = "1362669455284">
           <resourcesStatus>
                  <resourceStatus status = "SUCCUESS"
                         description = "created by bulk request"
                         id = "23d068d0-873a-11e2-bad4-00215edbb2a8”  />
....
                  <resourceStatus
                         status = "SUCCUESS"
                         description = "created by bulk request"
                         id = "23cfa580-873a-11e2-bad4-00215edbb2a8"/>
           </resourcesStatus>
    </ns2:bulkStatus>
}
}
```

适用于终端证书的外部 RESTful 服务 API

下表列出适用于终端证书的外部 RESTful 服务 API：

表 7-18　　适用于终端证书的 ERS API

操作	方法	URL	内容	查询字符串
创建终端证书	POST	/ers/config/endpointcert/certRequest/	表示 ZIP 文件的八位字节流	
获取证书资源版本信息	GET	/ers/config/endpointcert/version	不适用	

创建终端证书

下表列出创建终端证书 API 调用的主要特征：

表 7-19　　创建终端身份组 API 调用的主要特征

说明	创建终端证书
摘要	POST /ers/config/endpointcert/certRequest/
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	终端证书
响应头	Content-Length、Content-Type、Location
响应消息正文	表示 ZIP 文件的八位字节流
响应状态	201，400，401，403，415，429，500

创建终端证书 API 调用的请求示例

```
PUT https://<ISE-ADMIN-NODE>/ers/config/endpointcert/certRequest
HTTP Content-Type header:
application/vnd.com.cisco.ise.ca.endpointcert.1.0+xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointcert xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="ca.ers.ise.cisco.com">
    <certTemplateName>Certificate_Template_Name</certTemplateName>
    <certificateRequest>
        <entry>
            <key>san</key>
            <value>11-22-33-44-55-66</value>
        </entry>
        <entry>
            <key>cn</key>
            <value>userName [or] machineName</value>
        </entry>
    </certificateRequest>
    <format>PKCS8 [or] PKCS8_CHAIN [or] PKCS12 [or] PKCS12_CHAIN</format>
    <password>password</password>
</ns3:endpointcert>
```

创建终端证书 API 调用的响应示例

```
HTTP Status: 200 (OK)
Content:
[Response is returned as an Octet Stream representing a ZIP file.]
```

适用于终端身份组的外部 RESTful 服务 API

下表列出适用于终端身份组的外部 RESTful 服务 API：

表 7-20　适用于终端身份组的 ERS API

操作	方法	URL	内容	查询字符串
获取所有终端组	GET	/ers/config/endpointgroup	不适用	page、size、sortacs 或 sortdsn、filter
获取终端组	GET	/ers/config/endpointgroup/{id[1]}	不适用	
创建终端组	POST	/ers/config/endpointgroup/	终端组	
更新终端组	PUT	/ers/config/endpointgroup/{id}	终端组	
删除终端组	DELETE	/ers/config/endpointgroup/{id}	不适用	
获取身份组资源版本信息	GET	/ers/config/ endpointgroup /version	不适用	

1. 终端组 ID 是 Cisco ISE 数据库中存储的 UUID 类型。

获取所有终端身份组

下表列出获取所有终端身份组 API 调用的主要特征：

表 7-21　获取所有终端身份组 API 调用的主要特征

说明	检索终端组的集合
摘要	GET /ers/config/endpoint
请求头	Accept、Authorization、Host
查询字符串	page、size、sortasc、sortdsc、filter
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	202，400，401，403，404，415，429，500

获取所有终端身份组 API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/endpointgroup
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
```

获取所有终端身份组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
<resources>
  <resource name="name1" id="id1" description="description1">
   <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/endpointgroup/id1" rel="self"/>
  </resource>
  <resource name="name2" id="id2" description="description2">
   <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/endpointgroup/id2" rel="self"/>
  </resource>
</resources>
</ns2:searchResult>
}
```

通过 ID 获取终端身份组

下表列出通过 ID 获取终端身份组 API 调用的主要特征：

表 7-22　　读取终端身份组 API 调用的主要特征

说明	检索指定的终端组
摘要	GET /ers/config/endpoint/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	终端资源类型
响应状态	200，400，401，403，404，415，429，500

读取终端身份组 API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
```

读取终端身份组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointgroup description="description" name="name" id="id"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
```

```
    <systemDefined>true</systemDefined>
</ns3:endpointgroup>
}
```

创建终端身份组

下表列出创建终端身份组 API 调用的主要特征：

表 7-23　创建终端身份组 API 调用的主要特征

说明	创建指定的终端组
摘要	POST /ers/config/endpoint/
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	终端组
响应头	Content-Length、Content-Type、Location
响应消息正文	终端组
响应状态	201，400，401，403，415，429，500

创建终端身份组 API 调用的请求示例

```
POST https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointgroup description="description" name="name" id="id"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <systemDefined>true</systemDefined>
</ns3:endpointgroup>
}
```

创建终端身份组 API 调用的响应示例

```
HTTP/1.1 201 OK（请从 Location 标头查看新终端 ID）
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type:
Location: https://cisco.com/ers/config/endpointgroup/444
```

更新终端身份组

下表列出更新终端身份组 API 调用的主要特征：

表 7-24　更新终端身份组 API 调用的主要特征

说明	更新指定的终端组
摘要	PUT /ers/config/endpoint/{id}

表 7-24　更新终端身份组 API 调用的主要特征（续）

请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	终端组
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，404，415，429，500

更新终端身份组 API 调用的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/ endpoint /333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:endpointgroup xmlns:ns2="identity.ers.ise.cisco.com" description="updated" id="0"
name="Group">
    <systemDefined>false</systemDefined>
</ns2:endpointgroup>
}
```

更新终端身份组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
 <updatedField field="description">
  <newValue>updated</newValue>
  <oldValue>Group</oldValue>
 </updatedField>
</ns2:updatedFields>
}
```

删除终端身份组

下表列出删除终端身份 API 调用的主要特征：

表 7-25　删除终端身份组 API 调用的主要特征

说明	删除指定的终端组
摘要	DELETE /ers/config/endpointgroup/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A

表 7-25　删除终端身份组 API 调用的主要特征（续）

响应头	Content-Length、Content-Type
响应消息正文	不适用
响应状态	204，400，401，403，404，415，429，500

删除终端身份组 API 调用的请求示例

```
DELETE https://<ISE-ADMIN-NODE>:9060/ers/config/endpoint/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
```

删除终端身份组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

适用于身份组的外部 RESTful 服务 API

下表列出适用于身份组的外部 RESTful 服务 API：

表 7-26　适用于身份组的 API

操作	方法	URL	内容	查询字符串
获取所有身份组	GET	/ers/config/identitygroup	不适用	page、size、sortacs 或 sortdsn、filter
通过 ID 获取身份组	GET	ers/config/identitygroup/{id}	不适用	
获取身份组资源版本信息	GET	/ers/config/identitygroup/version	不适用	

检索所有身份组

您可以使用此 API 调用检索 Cisco ISE 中的所有身份组。下表列出此 API 调用的主要特征：

表 7-27　检索所有身份组 API 调用的主要特征

说明	检索身份组资源的集合
摘要	GET /ers/config/identitygroup
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type

表 7-27　检索所有身份组 API 调用的主要特征（续）

响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，429，500

检索所有身份组 API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/identitygroup?page=0&size=20&sortacs=name
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.identitygroup.1.0+xml
```

检索所有身份组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <resources>
      <resource name="name1" id="id1" description="description1">
        <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/identitygroup/id1" rel="self"/>
      </resource>
      <resource name="name2" id="id2" description="description2">
        <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/identitygroup/id2" rel="self"/>
      </resource>
    </resources>
</ns2:searchResult>
}
```

通过 ID 获取身份组

下表列出通过 ID 获取身份组 API 调用的主要特征：

表 7-28　读取终端身份组 API 调用的主要特征

说明	检索指定的身份组
摘要	GET /ers/config/identitygroup/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	终端资源类型
响应状态	200，400，401，403，404，415，429，500

通过 ID 获取身份组 API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/identitygroup/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.identitygroup.1.0+xml
```

通过 ID 获取身份组 API 调用的响应示例

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:identitygroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <parent>parent</parent>
</ns3:identitygroup>
```

适用于访客用户的外部 RESTful 服务 API

下表列出适用于访客用户的外部 RESTful 服务 API：

表 7-29　　支持的场景

操作	方法	URL	内容
获取特定访客用户	GET	/ers/config/guestuser/{id}	不适用
获取所有访客用户	GET	/ers/config/guestuser/	不适用
创建访客用户	POST	/ers/config/guestuser /	访客用户信息 (XML)
更新访客用户	PUT	/ers/config/guestuser/{id}	部分或全部访客用户信息 (XML)
删除访客用户	DELETE	/ers/config/guestuser/{id}	不适用
暂停访客用户	PUT	/ers/config/guestuser/suspend/{id}	原因
恢复访客用户	PUT	/ers/config/guestuser/reinstate/{id}	不适用
发送电子邮件	PUT	/ers/config/guestuser/email/{id}/ portalId/{portalId}	发件人邮件
发送 SMS	PUT	/ers/config/guestuser/sms/{id}/portalId/ {portalId}	不适用
批准访客用户	PUT	/ers/config/guestuser/approve/{id}	不适用
拒绝访客用户	PUT	/ers/config/guestuser/deny/{id}	不适用
重置密码	PUT	/ers/config/guestuser/resetpassword/ {id}	不适用
启动批量执行	PUT	/ers/config/ guestuser/bulk	批量请求
获取批量状态	GET	/ers/config/ guestuser/bulk/{bulkId}	不适用
更改发起人的密码	PUT	/ers/config/guestuser/changeSponsor Password/{portalId}	operationAdditional Data
获取所有门户	GET	/ers/config/portal	不适用

表 7-29　　支持的场景

操作	方法	URL	内容
通过 ID 获取门户	GET	/ers/config/portal/{id}	不适用
获取访客 API 信息	GET	/ers/config/guestuser/versioninfo	不适用

Content Type 和 Accept 标头

每个访客帐户请求都需要进行以下设置：

- Content Type 设置为：`application/vnd.com.cisco.ise.identity.guestuser.2.0+xml`
- Accept 设置为：`application/vnd.com.cisco.ise.identity.guestuser.2.0+xml`

每个批量执行请求都需要进行以下设置：

- Content Type 设置为：`application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml`
- Accept 设置为：`application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml`

获取访客用户

您可以通过 GET 操作，使用访客的用户名或数据库记录 ID 从 ISE 数据库检索特定访客用户。

表 7-30　　获取访客用户主要特征

说明	检索指定的访客用户
摘要	GET /ers/config/guestuser/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	访客用户类型的资源
响应状态	200，400，401，403，404，415，500

获取访客用户示例

- 通过 ID 获取访客用户的示例（第 7-26 页）
- 通过开头为“ilucky”的用户名进行过滤的示例（第 7-26 页）
- 通过开头为“ilucky”的用户名和开头为“J”的姓氏进行过滤的示例（第 7-27 页）
- 通过名字“John”进行过滤并按用户名进行排序的示例（第 7-28 页）
- 使用 curl 的访客用户请求和响应示例（第 7-29 页）

通过 ID 获取访客用户的示例

请求

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/3333


Content-Type - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Accept - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Authorization - Basic xxxxxxxxxxxxxxxxxxx
```

响应

```
<?xml version="1.0" encoding="UTF-8"?>
<ns3:guestuser xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com"
name="fzvhervocnz" id="af827350-1f0f-11e4-b961-005056103001">
 <link type="application/xml" href="https://10.0.10.130:9060/ers/config/guestuser/3333"
rel="self"/>
 <customFields/>
 <guestAccessInfo>
  <fromDate>08/08/2014 8:21</fromDate>
  <toDate>08/09/2014 8:21</toDate>
  <validDays>1</validDays>
 </guestAccessInfo>
 <guestInfo>
  <company>Cisco</company>
  <creationTime>08/08/2014 08:21</creationTime>
  <emailAddress>doe@example.com</emailAddress>
  <enabled>true</enabled>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
  <notificationLanguage>English</notificationLanguage>
  <password>12345</password>
  <phoneNumber>9999998877</phoneNumber>
  <smsServiceProvider>ATT</smsServiceProvider>
  <userName>Guest1</userName>
 </guestInfo>
 <guestType>Daily (default)</guestType>
 <personBeingVisited>sponsor@example.com</personBeingVisited>
 <reasonForVisit>Interview</reasonForVisit>
 <sponsorUserName>SponsoredUser1</sponsorUserName>
 <status>Awaiting Initial Login</status>
</ns3:guestuser>
```

相关主题

有关在 API 中是否显示密码的详细信息，请参阅响应错误消息（第 6-7 页）。

通过开头为“ilucky”的用户名进行过滤的示例

请求

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/?filter=userName.STARTSW.ilucky
```

响应

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult
    xmlns:ns2="ers.ise.cisco.com" total="8">
    <resources>
        <resource name="ilucky101" id="a0957160-6224-11e3-9bc2-000c2932c73c">
```

```
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/a0957160-6224-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky102" id="e14f4460-6224-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/e14f4460-6224-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky201" id="123581f0-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/123581f0-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky301" id="154f9330-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/154f9330-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky401" id="172c6980-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/172c6980-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky501" id="19631fa0-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/19631fa0-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky601" id="1b44b0e0-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/1b44b0e0-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky602" id="2e1ac600-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/2e1ac600-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
    </resources>
</ns2:searchResult>
```

通过开头为“ilucky”的用户名和开头为“J”的姓氏进行过滤的示例

请求

```
GET
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/?filter=userName.STARTSW.ilucky&filter=
lastName.STARTSW.j
```

响应

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult
    xmlns:ns2="ers.ise.cisco.com" total="8">
    <resources>
        <resource name="ilucky101" id="a0957160-6224-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/a0957160-6224-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky102" id="e14f4460-6224-11e3-9bc2-000c2932c73c">
```

```
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/e14f4460-6224-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky201" id="123581f0-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/123581f0-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky301" id="154f9330-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/154f9330-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky401" id="172c6980-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/172c6980-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky501" id="19631fa0-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/19631fa0-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky601" id="1b44b0e0-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/1b44b0e0-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
        <resource name="ilucky602" id="2e1ac600-6227-11e3-9bc2-000c2932c73c">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/2e1ac600-6227-11e3-9bc2-000c2932c73
c" rel="self"/>
        </resource>
    </resources>
</ns2:searchResult>
```

通过名字“John”进行过滤并按用户名进行排序的示例

请求

```
GET
https://<ISE-Admin-node>:9060/ers/config/guestuser/?page=0&size=10&sortdsc=name&filter=fir
stName.eq.john
```

响应

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult
    xmlns:ns2="ers.ise.cisco.com" total="2">
    <resources>
        <resource name="jdoe0002" id="886f5b40-5ece-11e3-8faf-000c29c56fc6">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/886f5b40-5ece-11e3-8faf-000c29c56fc
6" rel="self"/>
        </resource>
        <resource name="jdoe0001" id="79e5a5a0-5df9-11e3-84f5-000c29c56fc6">
            <link type="application/xml"
href="https://ISE-ADMIN-NODE:9060/ers/config/guestuser/79e5a5a0-5df9-11e3-84f5-000c29c56fc
6" rel="self"/>
        </resource>
    </resources>
</ns2:searchResult>
```

使用 curl 的访客用户请求和响应示例

以下示例介绍使用 `curl` Linux 命令通过向 ISE 发送的 ID 获取访客用户的请求及其响应。

curl 命令

```
$ curl -v -k -H 'ACCEPT:application/vnd.com.cisco.ise.identity.guestuser.2.0+xml'
https://username:password@<ISE-ADMIN-NODE>:9060/ers/config/guestuser/user1
* About to connect() to <ISE-ADMIN-NODE> port 9060
*   Trying 111.11.11.111...* connected
* Connected to <ISE-ADMIN-NODE> (<ISE-ADMin-NODE-IP>) port 9060
* successfully set certificate verify locations:
*   CAfile: /usr/share/ssl/certs/ca-bundle.crt
  CApath: none
* SSL connection using DHE-RSA-AES256-SHA
* Server certificate:
*        subject: /CN=<ISE-ADMIN-NODE>
*        start date: 2013-11-26 00:56:55 GMT
*        expire date: 2014-11-26 00:56:55 GMT
*        common name: <ISE-ADMIN-NODE>
*        issuer: /CN=<ISE-ADMIN-NODE>
* Server auth using Basic with user 'username'
```

通过 ID 获取访客用户的请求

```
> GET /ers/config/guestuser/444
Authorization: Basic xxxxxxxxxxxxxxxx
User-Agent: curl/7.12.1 (i386-redhat-linux-gnu) libcurl/7.12.1 OpenSSL/0.9.7a zlib/1.2.1.2
libidn/0.5.6
Host: <ISE-ADMIN-NODE>:9060
Pragma: no-cache
ACCEPT:application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

响应

```
< HTTP/1.1 200 OK
< Pragma: No-cache
< Cache-Control: no-cache
< Expires: Wed, 31 Dec 1969 16:00:00 PST
< Set-Cookie: JSESSIONIDSSO=0FCBC2621A0897193FE3105B3FBA8F16; Path=/; Secure
< Set-Cookie: JSESSIONID=5B6092B3FCCE047F7282C52592FAFC7A; Path=/ers; Secure
< Date: Thu, 02 Jan 2014 23:01:59 GMT
< Content-Type: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
< Content-Length: 1162
< Server:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:guestuser xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com"
name="user1" id="b4bdf2b0-73e1-11e3-8cdf-000c29c56fc6">
<link type="application/xml" href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/444"
rel="self"/>
<guestAccessInfo>
<fromDate>08/06/2014 23:26</fromDate>
<toDate>08/07/2014 23:26</toDate>
<validDays>1</validDays>
</guestAccessInfo>
<guestInfo>
<company>New Company</company>
<emailAddress>john@example.com</emailAddress>
<firstName>John</firstName>
<lastName>Doe</lastName>
<notificationLanguage>English</notificationLanguage>
<phoneNumber>9999998877</phoneNumber>
<smsServiceProvider>Global Default</smsServiceProvider>
<userName>user1</userName>
```

```
</guestInfo>
<guestType>Daily (default)</guestType>
<personBeingVisited>sponsor@example.com</personBeingVisited>
<portalId>ff2d99e0-2101-11e4-b5cf-005056bf2f0a</portalId>
<reasonForVisit>Interview</reasonForVisit>
</ns3:guestuser>
```

相关主题

有关在 API 中是否显示密码的详细信息，请参阅响应错误消息（第 6-7 页）。

获取所有访客用户

您可以使用 GET 操作检索 ISE 数据库中的所有访客用户，并根据名称、用户名或邮件地址等条件过滤结果。响应包括访客的用户名、ID 和连接到其完整表示的链接。

表 7-31　获取所有访客用户的主要特征

说明	检索访客用户的集合
摘要	GET /ers/config/guestuser/
请求头	Accept、Authorization、Host
查询字符串	page、size、sortasc、sortdsc、filter
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

相关主题

过滤参数（第 6-8 页）

获取所有访客用户示例

在下面的示例中，GET 操作会检索用户名以 `ilu` 开头且名字以 `b` 开头的所有访客用户。

请求

```
GET
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/?page=0&size=10&sortasc=name&filter=nam
e.STARTSW.ilu&filter=firstName.STARTSW.b

Content-Type - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Accept - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Authorization - Basic xxxxxxxxxxxxxxxxxxx
```

响应

```
HTTP/1.1 200 OK;
Date:Sat, 15 Dec 2012 21:55:05 GMT;
Content-Length:1439;
Content-Type:application/vnd.com.cisco.ise.ers.searchresult.1.0+xml;

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<ns2:searchResult xmlns:ns2="ers.ise.cisco.com" total="6">
  <resources>
    <ns2:resource name="ilucky01" id="61dc9060-46a1-11e2-b141-000c290fcf9a">
      <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/61dc9060-46a1-11e2-b141-000c290fc
f9a" rel="self"/>
    </ns2:resource>
    <ns2:resource name="ilucky02" id="3f43bb40-468e-11e2-8f92-000c290fcf9a">
      <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/3f43bb40-468e-11e2-8f92-000c290fc
f9a" rel="self"/>
    </ns2:resource>
    <ns2:resource name="ilucky03" id="6c65d6d0-468e-11e2-8f92-000c290fcf9a">
      <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/6c65d6d0-468e-11e2-8f92-000c290fc
f9a" rel="self"/>
    </ns2:resource>
    <ns2:resource name="ilucky04" id="6948bdb0-46a1-11e2-b141-000c290fcf9a">
      <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/6948bdb0-46a1-11e2-b141-000c290fc
f9a" rel="self"/>
    </ns2:resource>
    <ns2:resource name="ilucky05" id="abbb6440-46a1-11e2-b141-000c290fcf9a">
      <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/abbb6440-46a1-11e2-b141-000c290fc
f9a" rel="self"/>
    </ns2:resource>
    <ns2:resource name="ilucky06" id="4d9a1530-46fd-11e2-b70b-000c290fcf9a">
      <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/4d9a1530-46fd-11e2-b70b-000c290fc
f9a" rel="self"/>
    </ns2:resource>
  </resources>
</ns2:searchResult>
```

创建访客用户

您可以使用 POST 操作创建新的访客用户帐户，此帐户使访客可以通过访客流程登录。创建访客用户帐户时必须提供 guestType。

表 7-32　　创建访客用户的主要特征

说明	创建指定的内部用户
摘要	POST /ers/config/guestuser/
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	访客用户
响应头	Content-Length、Content-Type、Location
响应消息正文	访客用户类型的资源
响应状态	201，400，401，403，415，500

访客用户 XML 结构

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:guestuser name="guestUser" id="123456789" description="ERS Example user "
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <customFields>
        <entry>
            <key>some key</key>
            <value>some value</value>
        </entry>
        <entry>
            <key>another key</key>
            <value>and its value</value>
        </entry>
    </customFields>
    <guestInfo>
        <emailAddress>email@some.uri.com</emailAddress>
        <enabled>true</enabled>
        <password>asdlkj324ew</password>
        <phoneNumber>3211239034</phoneNumber>
        <smsServiceProvider>GLobal Default</smsServiceProvider>
        <userName>DS3ewdsa34wWE</userName>
    </guestInfo>
    <guestType>Contractor</guestType>
    <portalId>23423432523</portalId>
    <sponsorUserName>Mr Spons</sponsorUserName>
</ns3:guestuser>
```

创建访客用户示例

请求

```
POST https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/

Content-Type - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Accept - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Authorization - Basic xxxxxxxxxxxxxxxxxxx

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:guestuser xmlns:ns2="identity.ers.ise.cisco.com">
    <guestAccessInfo>
        <fromDate>08/08/2014 08:15</fromDate>
        <toDate>08/09/2014 08:15</toDate>
        <validDays>1</validDays>
    </guestAccessInfo>
    <guestInfo>
        <company>New Company</company>
        <emailAddress>doe@example.com</emailAddress>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <notificationLanguage>English</notificationLanguage>
        <phoneNumber>9999998877</phoneNumber>
        <smsServiceProvider>Global Default</smsServiceProvider>
        <userName>guestuser1</userName>
    </guestInfo>
    <guestType>Daily (default)</guestType>
    <personBeingVisited>sponsor@example.com</personBeingVisited>
    <portalId>ff2d99e0-2101-11e4-b5cf-005056bf2f0a</portalId>
    <reasonForVisit>Interview</reasonForVisit>
</ns2:guestuser>
```

响应

```
HTTP/1.1 201 Created;
Date:Sat, 15 Dec 2012 21:20:51 GMT;
Content-Length:0;
Location:https://<ISE-ADMIN-NODE>/ers/config/guestuser/e1bb8290-6ccb-11e3-8cdf-000c29c56fc
6;
Set-Cookie:JSESSIONID=28CF43F1ACCC7448BED7255DC7B787EE; Path=/ers;
Secure;JSESSIONIDSSO=DB6D6900088D1863CA84863570392E4C; Path=/; Secure;
Content-Type:application/xml;
```

相关主题

有关在 API 中是否显示密码的详细信息，请参阅响应错误消息（第 6-7 页）。

更新访客用户

通过使用 PUT 操作更新资源，您可以更改现有访客用户的属性。您可以全部或部分更新访客用户的属性。

表 7-33　更新访客用户的主要特征

说明	更新指定的访客用户
摘要	PUT /ers/config/guestuser/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	访客用户
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，404，415，500

更新用户示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/name/ilucky101

Content-Type - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Accept - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Authorization - Basic xxxxxxxxxxxxxxxxxx

<?xml version="1.0" encoding="UTF-8"?>
<ns2:guestuser xmlns:ns2="identity.ers.ise.cisco.com">
    <portalId>ff2d99e0-2101-11e4-b5cf-005056bf2f0a</portalId>
    <reasonForVisit>Interview</reasonForVisit>
</ns2:guestuser>
```

响应

```
Status:200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
 <updatedField field="ReasonForVisit">
  <newValue>Interview</newValue>
  <oldValue>no reason</oldValue>
```

```
 </updatedField>
 <updatedField field="validDays">
  <newValue>0</newValue>
  <oldValue>1</oldValue>
 </updatedField>
</ns2:updatedFields>
```

删除访客用户

您可以使用数据库记录 ID 从 ISE 数据库删除访客用户的记录。用户在下次尝试时将无法登录。

表 7-34　　删除访客用户的主要特征

说明	删除指定的访客用户
摘要	DELETE /ers/config/guestuser/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	不适用
响应状态	204，400，401，403，404，415，500

删除访客用户示例

请求

```
DELETE https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/3333

Content-Type - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Accept - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Authorization - Basic xxxxxxxxxxxxxxxxxx
```

响应

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

暂停访客用户

使用 PUT 操作可暂停特定访客用户。用户在下次尝试时将无法登录。您必须包含暂停原因。原因可以包含空格。

表 7-35　　暂停访客用户的主要特征

说明	暂停指定的访客用户
摘要	PUT /ers/config/guestuser/suspend/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	原因

表 7-35　暂停访客用户的主要特征

响应头	Content-Length、Content-Type
响应消息正文	访客用户类型的资源
响应状态	204，400，401，403，404，415，500

通过 ID 暂停访客用户示例

请求

```
PUT https:/<ISE-ADMIN-NODE>:9060/ers/config/guestuser/suspend/3333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
Content-Type - application/vnd.com.cisco.ise.identity.guestuser.2.0+xml

<?xml version="1.0" encoding="UTF-8"?>
<ns3:operationAdditionalData xmlns:ns2="identity.ers.ise.cisco.com"
xmlns:ns3="ers.ise.cisco.com">
    <requestAdditionalAttributes>
        <additionalAttribute name="reason" value="AUP not accepted"/>
    </requestAdditionalAttributes>
</ns3:operationAdditionalData>
```

响应

```
HTTP/1.1 204 No Content
Sat, 15 Dec 2012 10:14:38 GMT
```

恢复访客用户

使用 PUT 操作可恢复暂停的访客用户帐户。

表 7-36　恢复访客用户的主要特征

说明	恢复指定的访客用户
摘要	PUT /ers/config/guestuser/reinstate/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	访客用户类型的资源
响应状态	204，400，401，403，404，415，500

恢复访客用户示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/reinstate/33
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2012 10:20:48 GMT
```

向访客用户发送邮件

使用 PUT 操作可向访客用户的邮件帐户发送邮件。此操作需要在 Cisco ISE 中配置 SMTP 服务器。请求需要使用门户 ID，因为门户配置包含邮件正文和主题所需的信息。

表 7-37　向访客用户发送邮件的主要特征

说明	向指定访客用户发送邮件
摘要	PUT /ers/config/guestuser/email/{id}/portalId/{portalID}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	发件人邮件
响应头	Content-Length、Content-Type
响应消息正文	不适用
响应状态	204，400，401，403，404，415，500

向访客用户发送邮件示例

请求

```
PUT
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/email/4444/portalId/ff2d99e0-2101-11e4-
b5cf-005056bf2f0a
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept:
    application/vnd.com.cisco.ise.identity.guestuser.2.0+xml

<?xml version="1.0" encoding="UTF-8"?>
 <ns3:operationAdditionalData xmlns:ns2="identity.ers.ise.cisco.com"
xmlns:ns3="ers.ise.cisco.com">
       <requestAdditionalAttributes>
           <additionalAttribute name="senderEmail" value="sender Email"/>
       </requestAdditionalAttributes>
</ns3:operationAdditionalData>
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2012 10:20:48 GMT
```

向访客用户发送 SMS 文本

使用 PUT 操作可向访客用户的移动电话发送文本消息。此操作需要在 Cisco ISE 中配置 SMTP 服务器。

请求需要使用门户 ID，因为门户配置包含文本正文所需的信息。

表 7-38　向访客用户发送邮件的主要特征

说明	向指定访客用户发送 SMS。
摘要	PUT /ers/config/guestuser/sms/{id}/portalId/{portalID}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	不适用
响应状态	204，400，401，403，404，415，500

发送 SMS 示例

请求

```
PUT
https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/sms/444/portalId/ff2d99e0-2101-11e4-b5cf-005056bf2f0a
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept:
    application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2012 10:20:48 GMT
```

批准访客用户

此操作允许您批准访客用户帐户。此操作需要使用访客帐户 ID。

表 7-39　获取 API 版本的主要特征

说明	批准指定的访客用户
摘要	PUT /ers/config/guestuser/approve/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	API 版本
响应状态	200，400，401，403，404，415，500

批准访客用户示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/approve/3333
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2012 10:20:48 GMT
```

拒绝批准访客用户帐户

此操作允许您拒绝批准访客用户帐户。此操作需要使用访客帐户 ID。

表 7-40　获取 API 版本的主要特征

说明	拒绝指定的访客用户
摘要	PUT /ers/config/guestuser/deny/{id}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	API 版本
响应状态	200，400，401，403，404，415，500

拒绝批准访客用户示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/deny/7777
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2012 10:20:48 GMT
```

重置访客用户帐户的密码

此操作允许您重置访客用户帐户的密码。此操作需要使用访客帐户 ID。此操作会返回生成的新密码。您无法使用 REST API 指定自己的密码。

表 7-41　获取 API 版本的主要特征

说明	重置指定访客用户的密码
摘要	PUT /ers/config/guestuser/resetpassword/{id}

表 7-41　获取 API 版本的主要特征

请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	新密码
响应状态	200，400，401，403，404，415，500

重置访客用户的密码示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/resetpassword/7777
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2014 10:20:48 GMT
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:operationResult xmlns:ns2="ers.ise.cisco.com">
    <attributesList>
        <attribute value="DdsAASDs%$##@ssds12" name="password"/>
    </attributesList>
</ns2:operationResult>
```

启动访客用户的批量执行

通过批量操作请求，您可以在单个请求中发送最多 500 项操作，或根据 ID 发送最多 5000 项操作。

如果请求有效，服务器将返回状态代码 `202 (ACCEPTED)`，并在 LOCATION 响应头中包含唯一的批量标识符，您可以使用此标识符通过获取批量状态操作跟踪批量状态。

一次只允许运行一个批量操作。如果在另一个批量操作仍在执行时发布批量操作请求，服务器将返回响应状态 `503 (Service Unavailable)`，并包含相应的要求客户端稍后重试的描述性消息。

表 7-42　启动批量执行的主要特征

说明	启动执行
摘要	PUT /ers/config/guestuser/bulk
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	批量请求
响应头	Content-Length、Content-Type
响应消息正文	不适用
响应状态	202，400，401，403，404，415，500

创建访客批量执行示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/bulk
Authorization: Basic
Content-Type: application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:bulkRequest xsi:type="ns2:guestUserBulkRequest"
                resourceMediaType="vnd.com.cisco.ise.identity.guestuser.1.0+xml"
                operationType="create"
                xmlns:ns2="identity.ers.ise.cisco.com"
     xmlns:ns3="ers.ise.cisco.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
 <resourcesList>
  <resource xsi:type="ns2:GuestUser" description="created by bulk">
  <portalId>6ab68890-d0f1-11e3-a1d5-005056bf4687</portalId>
   <guestAccesstInfo>
        <groupTag>group</groupTag>
        <validDays>2</validDays>
        <location>London</location>
        <ssid>guest_ssid</ssid>
    </guestAccesstInfo>
<guestInfo>
        <company>new company</company>
        <emailAddress>joe@example.com</emailAddress>
        <enabled>true</enabled>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <phoneNumber>6033203311</phoneNumber>
        <userName>lucky7</userName>
        <password>1234</password>
        <notificationLanguage>English</notificationLanguage>
        <smsServiceProvider>ATT</smsServiceProvider>
    </guestInfo>
    <guestType>DAILY</guestType>
    <reasonForVisit>interview</reasonForVisit>
    <personBeingVisited>sponsor@cisco.com</personBeingVisited>
  </resource>

...
  <resource xsi:type="ns2:GuestUser" description="created by bulk">
  <portalId>6ab68890-d0f1-11e3-a1d5-005056bf4687</portalId>
   <guestAccesstInfo>
        <groupTag>group</groupTag>
        <validDays>3</validDays>
        <location>London</location>
        <ssid>guest_ssid</ssid>
    </guestAccesstInfo>
    <guestInfo>
        <company>new company</company>
        <emailAddress>mary@example.com</emailAddress>
        <enabled>true</enabled>
        <firstName>Mary</firstName>
        <lastName>Sue</lastName>
        <phoneNumber>6039990000</phoneNumber>
        <userName>lucky13</userName>
        <password>1234</password>
        <notificationLanguage>English</notificationLanguage>
        <smsServiceProvider>ATT</smsServiceProvider>
    </guestInfo>
    <guestType>DAILY</guestType>
    <reasonForVisit>interview</reasonForVisit>
```

```
    <personBeingVisited>sponsor@cisco.com</personBeingVisited>
  </resource>
 </resourcesList>
</ns3:bulkRequest>
```

响应

```
HTTP/1.1 202 ACCEPTED
        Date: Thu, 12 Jul 2012 23:59:59 GMT
        Location: https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/123443545334
```

相关主题

获取终端批量状态（第 7-15 页）

获取访客用户的批量状态

如果批量执行请求有效且没有其他正在处理的批量操作，服务器将在 LOCATION 响应头中返回唯一的批量标识符。使用此 ID 可跟踪批量状态。您可以获取操作开始之后至少 2 小时的状态报告。

表 7-43　　获取批量状态的主要特征

说明	监控指定的批量执行进度
摘要	GET /ers/config/guestuser/bulk/{bulkid}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	批量状态
响应状态	200，400，401，403，404，415，500

获取访客用户的批量状态示例

请求

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/bulk/53454354534 HTTP/1.1
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.guestuserbulkrequest.1.0+xml
```

响应

```
HTTP/1.1 200 OK
Date: Thu Mar 07 18:17:35 IST 2013 GMT
Content-Type: application/vnd.com.cisco.ise.ers.guestuserbulkrequest.1.0+xml
Content-Length: 16347
{
<ns2:bulkStatus
  xmlns:ns2 = "ers.ise.cisco.com"
  successCount = "50"
  startTime = "Thu Mar 07 17:17:35 IST 2013"
  resourcesCount = "50"
  operationType = "create"
  resourceMediaType = "vnd.com.cisco.ise.ers.identity.guestuser.1.0+xml"
  failCount = "0"
  executionStatus = "COMPLETED"
```

```
  bulkId = "53454354534">

  <resourcesStatus>
    <resourceStatus
     status = "SUCCUESS"
     description = "created by bulk request"
     id = "23d068d0-873a-11e2-bad4-00215edbb2a8"/>

…

    <resourceStatus
     status = "SUCCUESS"
     description = "created by bulk request"
     id = "23cfa580-873a-11e2-bad4-00215edbb2a8"/>
  </resourcesStatus>
</ns2:bulkStatus>
}
```

更改发起人的密码

此操作允许您更改当前登录的发起人的密码。此操作需要使用门户 ID。

表 7-44　　获取 API 版本的主要特征

说明	更新已登录发起人的密码
摘要	PUT /ers/config/guestuser/changeSponsorPassword/{portalId}
请求头	Accept、Authorization、Host
查询字符串	不适用
请求消息正文	不适用
响应头	Content-Length、Content-Type
响应消息正文	API 版本
响应状态	200，400，401，403，404，415，500

更改发起人的密码示例

请求

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/guestuser/changeSponsorPassword/88888
    Host: cisco.com
    Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.identity.guestuser.2.0+xml

<?xml version="1.0" encoding="UTF-8"?>
<ns3:operationAdditionalData xmlns:ns2="identity.ers.ise.cisco.com"
xmlns:ns3="ers.ise.cisco.com">
    <requestAdditionalAttributes>
        <additionalAttribute name="newPassword" value="Cisco1234"/>
        <additionalAttribute name="currentPassword" value="Autom8me"/>
    </requestAdditionalAttributes>
</ns3:operationAdditionalData>
```

响应

```
HTTP/1.1 204 OK
Date: Sat, 15 Dec 2012 10:20:48 GMT
```

适用于门户的外部 RESTful 服务 API

下表列出适用于门户的外部 RESTful 服务 API：

表 7-45　　适用于门户的 API

操作	方法	URL	内容	查询字符串
获取所有门户	GET	/ers/config/portal	不适用	Page、Size、sortacs 或 sortdsn、Filter
通过 ID 获取门户	GET	/ers/config/portal/{id}	不适用	

获取所有门户

下表列出获取所有门户 API 调用的主要特征：

表 7-46　　获取所有门户 API 调用的主要特征

说明	检索门户的集合
摘要	GET /ers/config/portal
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有门户调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/portal
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.portal.1.0+xml
```

获取所有门户调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="utf-8" standalone="yes"?> <ns2:searchResult total="2"
xmlns:ns2="ers.ise.cisco.com">
        <resources>
            <resource name="portal1" id="id1">
```

```
                <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/portal/id1" rel="self"/>
            </resource>
            <resource name="portal2" id="id2">
                <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/portal/id2" rel="self"/>
            </resource>
        </resources>
</ns2:searchResult>
```

通过 ID 获取门户

下表列出通过 ID 获取门户 API 调用的主要特征：

表 7-47　　通过 ID 获取门户 API 调用的主要特征

说明	检索指定的门户
摘要	GET /ers/config/portal/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	门户资源类型
响应状态	200，400，401，403，404，415，500

通过 ID 获取门户调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/portal/333
    Authorization: Basic xxxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.portal.1.0+xml
```

通过 ID 获取门户调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.identity.portal.1.0+xml Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:portal name="sponsor" id="d7b703f0-b073-11e3-bd6c- 005056a15fa7"
        xmlns:ns2="ers.ise.cisco.com"
        xmlns:ns3="identity.ers.ise.cisco.com">
        <link type="application/xml"
href="https://<ISE-ADMIN-NODE>:9060/ers/config/portal/333" rel="self"/>
    <allowSponsorToChangeOwnPassword>false</allowSponsorToChangeOwnPassword>
        <GuestUserFieldList>
            <GuestUserField>
                <customType>false</customType>
                <dataType>DROPDOWN</dataType>
    <dictionaryLabelKey>ui_sms_provider_label</dictionaryLabelKey>
                <labelName>SMS Service Provider</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
```

```
                <dataType>TEXT</dataType>
    <dictionaryLabelKey>ui_company_label</dictionaryLabelKey>
                <labelName>Company</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
                <dataType>TEXT</dataType>
    <dictionaryLabelKey>ui_first_name_label</dictionaryLabelKey>
                <labelName>First name</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
                <dataType>TEXT</dataType>
    <dictionaryLabelKey>ui_reason_visit_label</dictionaryLabelKey>
                <labelName>Reason for visit</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>true</customType>
                <dataType>TEXT</dataType>
    <dictionaryLabelKey>ui_ssn-number_text_label</dictionaryLabelKey>
                <instructionText>social </instructionText>
                <labelName>ssn-number</labelName>
                <required>false</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
                <dataType>PHONE</dataType>
    <dictionaryLabelKey>ui_phone_number_label</dictionaryLabelKey>
                <labelName>Phone number</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
                <dataType>EMAIL</dataType>
    <dictionaryLabelKey>ui_person_visited_label</dictionaryLabelKey>
                <labelName>Person being visited</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
                <dataType>EMAIL</dataType>
    <dictionaryLabelKey>ui_email_address_label</dictionaryLabelKey>
                <labelName>Email address</labelName>
                <required>true</required>
            </GuestUserField>
            <GuestUserField>
                <customType>false</customType>
                <dataType>TEXT</dataType>
    <dictionaryLabelKey>ui_last_name_label</dictionaryLabelKey>
                <labelName>Last name</labelName>
                <required>true</required>
            </GuestUserField>
    </GuestUserFieldList>
</ns3:portal>
}
```

适用于配置文件的外部 RESTful 服务 API

下表列出适用于配置文件的外部 RESTful 服务 API：

表 7-48　　适用于门户的 API

操作	方法	URL	内容	查询字符串
获取所有配置文件	GET	/ers/config/profilerprofile	不适用	Page、Size、sortacs 或 sortdsn、Filter
通过 ID 获取配置文件	GET	/ers/config/profilerprofile/{id}	不适用	

获取所有配置文件

下表列出获取所有门户 API 调用的主要特征：

表 7-49　　获取所有门户 API 调用的主要特征

说明	检索配置文件的集合
摘要	GET /ers/config/profilerprofile
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有配置文件调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/profilerprofile
    Authorization: Basic xxxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.profilerprofile.1.0+xml
```

获取所有配置文件调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2014 23:59:59 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <nextPage type="application/xml" href="link-to-next-page" rel="next"/>
    <previousPage type="application/xml" href="link-to-previous-page" rel="previous"/>
    <resources>
        <resource name="name1" id="id1" description="description1"/>
        <resource name="name2" id="id2" description="description2"/>
    </resources>
</ns2:searchResult>
```

通过 ID 获取门户

下表列出通过 ID 获取配置文件 API 调用的主要特征：

表 7-50　通过 ID 获取门户 API 调用的主要特征

说明	检索指定的配置文件
摘要	GET /ers/config/profilerprofile/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	配置文件资源类型
响应状态	200，400，401，403，404，415，500

通过 ID 获取门户调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/profilerprofile/333
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.identity.profilerprofile.1.0+xml
```

通过 ID 获取门户调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2014 23:59:59 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:profilerprofile name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com"/
```

适用于网络设备的外部 RESTful 服务 API

下表列出适用于网络设备的外部 RESTful 服务 API：

表 7-51　适用于门户的 API

操作	方法	URL	内容	查询字符串
获取所有网络设备	GET	/ers/config/networkdevice	不适用	Page、Size、sortacs 或 sortdsn、Filter
获取网络设备	GET	/ers/config/networkdevice/{id}	不适用	
创建网络设备	POST	/ers/config/networkdevice	网络设备	
更新网络设备	PUT	/ers/config/networkdevice/{id}	网络设备	
删除网络设备	DELETE	/ers/config/networkdevice/{id}	不适用	
获取网络设备资源版本信息	GET	/ers/config/ networkdevice /versioninfo	不适用	

获取所有网络设备

下表列出获取所有网络设备 API 调用的主要特征：

表 7-52　　获取所有网络设备 API 调用的主要特征

说明	检索网络设备资源的集合
摘要	GET /ers/config/networkdevice
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有网络设备调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevice?page=1&size=20&sortacs=name
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml
```

获取所有网络设备调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:searchResult
        xmlns:ns2="ers.ise.cisco.com" total="1">
        <resources>
            <resource name="nd1" id="0d008bb0-2539-11e3-84ad-
    00215edbb2a8">
                <link type="application/xml"
    href="https://10.56.13.196:9060/ers/config/networkdevice/0d0
    08bb0-2539-11e3-84ad-00215edbb2a8" rel="self"/>
            </resource>
        </resources>
    </ns2:searchResult>
}
```

通过 ID 获取网络设备

下表列出通过 ID 获取网络设备 API 调用的主要特征：

表 7-53　　通过 ID 获取网络设备 API 调用的主要特征

说明	检索指定的网络设备
摘要	GET /ers/config/networkdevice/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A

表 7-53　通过 ID 获取网络设备 API 调用的主要特征（续）

请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	网络设备类型的资源
响应状态	200，400，401，403，404，415，500

通过 ID 获取网络设备调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevice/333
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml
```

通过 ID 获取网络设备调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml Content-Length:
16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns3:networkdevice
        xmlns:ns2="ers.ise.cisco.com"
        xmlns:ns3="network.ers.ise.cisco.com" name="nd1"
    id="0d008bb0-2539-11e3-84ad-00215edbb2a8">
        <link type="application/xml"
    href="https://10.56.13.196:9060/ers/config/networkdevice/0d0
    08bb0-2539-11e3-84ad-00215edbb2a8" rel="self"/>
        <authenticationSettings>
            <enableKeyWrap>false</enableKeyWrap>
            <keyInputFormat>ASCII</keyInputFormat>
            <networkProtocol>RADIUS</networkProtocol>
            <radiusSharedSecret>*****</radiusSharedSecret>
        </authenticationSettings>
<NetworkDeviceIPList>
        <NetworkDeviceIP>
            <ipaddress>1.2.3.4</ipaddress>
            <mask>32</mask>
        </NetworkDeviceIP>
    </NetworkDeviceIPList>
    <modelName>Unknown</modelName>
    <NetworkDeviceGroupList>
        <NetworkDeviceGroup>1d8c62b0-2539-11e3-84ad-
00215edbb2a8</NetworkDeviceGroup>
        <NetworkDeviceGroup>37053aa0-2539-11e3-84ad-
00215edbb2a8</NetworkDeviceGroup>
    </NetworkDeviceGroupList>
    <softwareVersion>Unknown</softwareVersion>
</ns3:networkdevice>
}
```

创建网络设备

下表列出创建网络设备 API 调用的主要特征：

表 7-54　　创建网络设备 API 调用的主要特征

说明	创建指定的网络设备
摘要	POST /ers/config/networkdevice/
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	网络设备
响应头	Content-Length、Content-Type、Location
响应消息正文	N/A
响应状态	200，400，401，403，415，500

创建网络设备调用的请求示例

```
POST https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevice/
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml {
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <ns3:networkdevice
        xmlns:ns2="ers.ise.cisco.com"
        xmlns:ns3="network.ers.ise.cisco.com" name="nd2">
        <authenticationSettings>
            <enableKeyWrap>false</enableKeyWrap>
            <keyInputFormat>ASCII</keyInputFormat>
            <networkProtocol>RADIUS</networkProtocol>
            <radiusSharedSecret>acsi</radiusSharedSecret>
        </authenticationSettings>
        <NetworkDeviceIPList>
            <NetworkDeviceIP>
                <ipaddress>1.2.3.4</ipaddress>
                <mask>32</mask>
            </NetworkDeviceIP>
        </NetworkDeviceIPList>
        <modelName>Unknown</modelName>
        <NetworkDeviceGroupList>
            <NetworkDeviceGroup>1d8c62b0-2539-11e3-84ad-
    00215edbb2a8</NetworkDeviceGroup>
            <NetworkDeviceGroup>37053aa0-2539-11e3-84ad-
    00215edbb2a8</NetworkDeviceGroup>
        </NetworkDeviceGroupList>
        <softwareVersion>Unknown</softwareVersion>
    </ns3:networkdevice>
}
```

创建网络设备调用的响应示例

```
HTTP/1.1 201 OK (see location header for the ID of the new device)
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml
Location: https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevice/444
```

更新网络设备

下表列出更新网络设备 API 调用的主要特征：

表 7-55　更新网络设备 API 调用的主要特征

说明	更新指定的网络设备
摘要	PUT /ers/config/networkdevice/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	网络设备
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，415，500

更新网络设备调用的请求示例

```
PUT https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevice/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml {
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <ns3:networkdevice
        xmlns:ns2="ers.ise.cisco.com"
        xmlns:ns3="network.ers.ise.cisco.com"
    name="nd2_updated">
        <authenticationSettings>
            <enableKeyWrap>true</enableKeyWrap>
        </authenticationSettings>
    </ns3:networkdevice>
}
```

更新网络设备调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml Content-Length: 529
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:updatedFields
xmlns:ns2="ers.ise.cisco.com">
    <updatedField field="name">
        <newValue>nd2_updated</newValue>
        <oldValue>nd2</oldValue>
    </updatedField>
    <updatedField field="enableKeywrap">
        <newValue>true</newValue>
        <oldValue>false</oldValue>
    </updatedField>
</ns2:updatedFields>
}
```

删除网络设备

下表列出删除网络设备 API 调用的主要特征：

表 7-56　　删除网络设备 API 调用的主要特征

说明	删除指定的网络设备
摘要	DELETE /ers/config/networkdevice/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

更新网络设备调用的请求示例

```
DELETE https://<ISE-ADMIN-NODE>:9060/ers/config/networjdevice/333
    Authorization: Basic xxxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.network.networkdevice.1.0+xml
```

更新网络设备调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

适用于网络设备组的外部 RESTful 服务 API

下表列出适用于网络设备组的外部 RESTful 服务 API：

表 7-57　　适用于 SGT 的 API

操作	方法	URL	内容	查询字符串
获取所有网络设备组	GET	/ers/config/networkdevicegroup	不适用	page、size、sortacs 或 sortdsn、filter
获取网络设备组	GET	/ers/config/networkdevicegroup/{id}	不适用	
获取网络设备组资源版本信息	GET	/ers/config/networkdevicegroup/versioninfo	不适用	

获取所有网络设备组

下表列出获取所有网络设备组 API 调用的主要特征：

表 7-58　获取所有网络设备组 API 调用的主要特征

说明	检索网络设备组资源的集合
摘要	GET /ers/config/networkdevicegroup
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有网络设备组 API 调用的请求示例

```
GET
https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevicegroup?page=1&size=20&sortacs=name
    Authorization: Basic xxxxxxxxxxxxxxxxxx
    Accept: application/vnd.com.cisco.ise.network.networkdevicegroup.1.0+xml
```

获取所有网络设备组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml Content-Length: 16347
{
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
                        <ns2:searchResult
             xmlns:ns2="ers.ise.cisco.com" total="0">
                            <resources>
  <resource name="Location#All Locations#loc1" id="1d8c62b0-2539-11e3-84ad-00215edbb2a8"
description="xxx">
                  <link type="application/xml"
href="https://10.56.13.196:9060/ers/config/networkdevicegroup/1d8c62b0-2539-11e3-84ad-0021
5edbb2a8" rel="self"/>
                            </resource>
   <resource name="Device Type#All Device Types#device type 555"
id="37053aa0-2539-11e3-84ad-00215edbb2a8" description="vvv">
                  <link type="application/xml"
 href="https://10.56.13.196:9060/ers/config/networkdevicegrou
     p/37053aa0-2539-11e3-84ad-00215edbb2a8" rel="self"/>
                            </resource>
                            </resources>
                       </ns2:searchResult>
}
```

获取网络设备组

下表列出获取网络设备组 API 调用的主要特征：

表 7-59　　获取网络设备组 API 调用的主要特征

说明	检索指定的网络设备组
摘要	GET /ers/config/networkdevicegroup/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	网络设备组类型的资源
响应状态	200，400，401，403，404，415，429，500

获取网络设备组 API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/networkdevicegroup/333
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.network.networkdevicegroup.1.0+xml
```

获取网络设备组 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.network.networkdevicegroup.1.0 +xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns3:networkdevicegroup
        xmlns:ns2="ers.ise.cisco.com"
        xmlns:ns3="network.ers.ise.cisco.com" name="Location#All
    Locations#loc1" id="1d8c62b0-2539-11e3-84ad-00215edbb2a8"
    description="xxx">
        <link type="application/xml"
    href="https://10.56.13.196:9060/ers/config/networkdevicegrou
    p/1d8c62b0-2539-11e3-84ad-00215edbb2a8" rel="self"/>
        <type>Location</type>
</ns3:networkdevicegroup>
}
```

适用于 SGT 的外部 RESTful 服务 API

下表列出适用于 SGT 的外部 RESTful 服务 API：

表 7-60　　适用于 SGT 的 API

操作	方法	URL	内容	查询字符串
获取所有 SGT	GET	/ers/config/sgt	不适用	page、size、sortacs、sortdsn、filter
获取 SGT	GET	/ers/config/sgt/{id}	不适用	
创建 SGT	POST	/ers/config/sgt/	SGT	
更新 SGT	PUT	/ers/config/sgt/	SGT	
删除 SGT	DELETE	/ers/config/sgt/	不适用	
启动批量执行	PUT	/ers/config/sgt/bulk	SGTBulkRequest	
批量监控	PUT	/ers/config/sgt/bulk/{bulkid}	不适用	
获取 SGT 资源版本信息	GET	/ers/config/sgt/versioninfo	不适用	

获取所有 SGT

下表列出获取所有 SGT API 调用的主要特征：

表 7-61　　主获取所有 SGT API 调用的主要特征

说明	检索 SGT 资源的集合
摘要	GET /ers/config/sgt
请求头	Accept、Authorization、Host
查询字符串	page、size、sortbyacn、sortbydcn、filter
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，429，500

获取所有 SGT API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/sgt?page=0&size=20&sortacs=name
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.sga.sgt.1.0+xml
```

获取所有 SGT API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
<resources>
  <resource name="name1" id="id1" description="description1">
   <link type="application/xml" href="https://<ISE-ADMIN-NODE>:9060/ers/config/sgt/id1"
rel="self"/>
  </resource>
 </resources>
</ns2:searchResult>
}
```

通过 ID 获取 SGT

下表列出通过 ID 获取 SGT API 调用的主要特征：

表 7-62　　通过 ID API 调用获取 SGT 的主要特征

说明	检索指定的 SGT
摘要	GET /ers/config/sgt/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	内部用户类型的资源
响应状态	200，400，401，403，404，415，429，500

通过 ID 获取 SGT API 调用的请求示例

```
GET https://<ISE-ADMIN-NODE>:9060/ers/config/sgt/333
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.sga.sgt.1.0+xml
```

通过 ID 获取 SGT API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.sga.sgt.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgt description="description" name="name" id="id" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="sga.ers.ise.cisco.com">
    <generationId>generationId</generationId>
    <isTagFromRange>isTagFromRange</isTagFromRange>
    <value>1</value>
</ns3:sgt>
}
```

创建 SGT

下表列出创建 SGT API 调用的主要特征：

表 7-63　创建 SGT API 调用的主要特征

说明	创建指定的 SGT
摘要	POST /ers/config/sgt/
请求头	Content-Type、Authorization、Host
查询字符串	N/A
请求消息正文	SGT
响应头	Content-Type（错误 / 验证的 ERSResponse - 400）、Location
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SGT API 调用的请求示例

```
POST /ers/config/sgt/ HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgt.1.1+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgt description="sgt description" name="newsgt" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">


</ns3:sgt>
```

创建 SGT API 调用的响应示例

```
HTTP/1.1 201 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Location: https://<ISE-ADMIN-NODE>/ers/config/sgt/333
```

更新 SGT

下表列出更新 SGT API 调用的主要特征：

表 7-64　更新 SGT API 调用的主要特征

说明	更新指定的 SGT
摘要	PUT /ers/config/sgt/{id}
请求头	Accept、Content=Type、Authorization: Basic、Host Content-Type: application/vnd.com.cisco.ise.trustsec.sgt.1.1+xml Accept: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
查询字符串	N/A

表 7-64　更新 SGT API 调用的主要特征（续）

请求消息正文	Sgt
响应头	Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
响应消息正文	已更新的字段
响应状态	200，400，401，403，404，415，500

更新 SGT API 调用的请求示例

```
PUT /ers/config/sgt/333
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgt.1.1+xml
Accept:
application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgt description="description" name="name" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .


</ns3:sgt>
```

更新 SGT API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.
cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns2:updatedFields
    xmlns:ns2="ers.ise.cisco.com">

   .
   .
   .


</ns2:updatedFields>
```

删除 SGT

下表列出删除 SGT API 调用的主要特征：

表 7-65　删除 SGT API 调用的主要特征

说明	删除指定的 SGT
摘要	DELETE /ers/config/sgt/{id}
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除 SGT API 调用的请求示例

```
DELETE /ers/config/sgt/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxxx
```

删除 SGT API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

适用于 SGACL 的外部 RESTful 服务 API

下表列出适用于 SGT 的外部 RESTful 服务 API：

表 7-66　适用于 SGACL 的 API

操作	方法	URL	内容	查询字符串
获取所有 SGACL	GET	/ers/config/sgacl	不适用	分页字符串
通过 ID 获取 SGACL	GET	/ers/config/sgacl/{id}	不适用	
创建 SGACL	POST	/ers/config/sgacl/	SGACL	
更新 SGACL	PUT	/ers/config/sgacl/	SGACL	
获取 SGACL 资源版本信息	GET	/ers/config/sgacl/versioninfo	不适用	

获取所有 SGACL

下表列出获取所有 SGACL API 调用的主要特征：

表 7-67　　获取所有 SGACL API 调用的主要特征

说明	检索排序和分页的所有 SGACL 资源
摘要	GET /ers/config/sgacl
请求头	Accept、Authorization: Basic、Host Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
查询字符串	start、size、sortasc、sortdsc
过滤器字段	不适用
排序字段	Name、description
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有 SGACL API 调用的响应示例

```
GET /ers/config/sgacl HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
```

获取所有 SGACL API 调用的请求示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <resources>
        <ns2:resource description="description1" name="name1" id="id1"/>
        <ns2:resource description="description2" name="name2" id="id2"/>
    </resources>
</ns2:searchResult>
}
```

通过 ID 获取 SGACL

下表列出通过 ID 获取 SGACL API 调用的主要特征：

表 7-68　通过 ID 获取 SGACL API 调用的主要特征

说明	检索指定的 SGT
摘要	GET /ers/config/sgacl/{id}
请求头	Accept、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	SGACL
响应状态	200，400，401，403，404，415，500

通过 ID 获取 SGACL API 调用的请求示例

```
GET /ers/config/sgt/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.trustsec.sgacl.1.0+xml
```

通过 ID 获取 SGACL API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.trustsec.sgacl.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgacl description="description" name="name" id="id" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .




</ns3:sgacl>
}
```

创建 SGACL

下表列出创建 SGACL API 调用的主要特征：

表 7-69　创建 SGACL API 调用的主要特征

说明	创建指定的 SGACL
摘要	POST /ers/config/sgacl/

表 7-69　创建 SGACL API 调用的主要特征（续）

请求头	Content-Type、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	SGACL
响应头	Location、Content-Type（错误 / 验证的 ERSResponse - 400）
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SGACL API 调用的请求示例

```
POST /ers/config/sgacl/ HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgacl.1.1+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgt description="sgt description" name="newsgt" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .



</ns3:sgt>
```

创建 SGACL API 调用的响应示例

```
HTTP/1.1 201 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Location: https://<ISE-ADMIN-NODE>/ers/config/sgacl/333
```

更新 SGACL

下表列出更新 SGACL API 调用的主要特征：

表 7-70　更新 SGACL API 调用的主要特征

说明	更新指定的 SGT
摘要	PUT /ers/config/sgacl/{id}
请求头	Accept、Content-Type、Authorization: Basic、Host Content-Type: application/vnd.com.cisco.ise.trustsec.sgacl.1.0+xml Accept: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
查询字符串	N/A
请求消息正文	SGACL

表 7-70　更新 SGACL API 调用的主要特征（续）

响应头	Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
响应消息正文	已更新的字段
响应状态	200，400，401，403，404，415，500

更新 SGACL API 调用的请求示例

```
PUT /ers/config/sgacl/333
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgacl.1.0+xml
Accept:
application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgacl description="description" name="name" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .


</ns3:sgacl>
```

更新 SGACL API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns2:updatedFields
    xmlns:ns2="ers.ise.cisco.com">

   .
   .
   .


</ns2:updatedFields>
```

删除 SGACL

下表列出删除 SGACL API 调用的主要特征：

表 7-71　删除 SGACL API 调用的主要特征

说明	删除指定的 SGACL
摘要	DELETE /ers/config/sgacl/{id}
请求头	Authorization: Basic、Host

表 7-71　删除 SGACL API 调用的主要特征（续）

查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

更新 SGACL API 调用的请求示例

```
DELETE /ers/config/sgacl/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
```

更新 SGACL API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

适用于 EgressMatrixCell 的外部 RESTful 服务 API

下表列出适用于 EgressMatrixCell 的外部 RESTful 服务 API：

表 7-72　适用于 EgressMatrixCell 的 API

操作	方法	URL	内容	查询字符串
获取所有单元	GET	/ers/config/egressmatrixcell	不适用	page、size、sortacs、sortdsn、filter
通过 ID 获取单元	GET	/ers/config/egressmatrixcell/{cell-id}	不适用	
创建单元	POST	/ers/config/egressmatrixcell/	EgressMatrixCell	
更新单元	PUT	/ers/config/egressmatrixcell/	EgressMatrixCell	
获取单元版本	GET	/ers/config/egressmatrixcell/versioninfo	不适用	
删除单元	DELETE	/ers/config/egressmatrixcell/{cell-id}	不适用	
设置所有单元状态	PUT	/ers/config/egressmatrixcell /status/{status-type}	statustype: MONITOR ENABLED DISABLED	
清除矩阵单元	PUT	/ers/config/egressmatrixcell/clearall	清除所有矩阵单元	
克隆单元	PUT	/ers/config/egressmatrixcell /clonecell/{cell-id}/srcsgt/{srcsgtid}/ dstsgt/{dstsgtid}	不适用	

获取所有矩阵单元

下表列出获取所有矩阵单元 API 调用的主要特征：

表 7-73　获取所有矩阵单元 API 调用的主要特征

说明	检索矩阵单元的集合
摘要	GET /ers/config/egressmatrixcell
请求头	Accept、Authorization: Basic、Host Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
查询字符串	start、size、sortasc、sortdsc、filter
过滤器字段	sgtSrcName、sgtDstName、matrixStatus、description、sgtSrcTagValue、sgtDstTagValue
排序字段	sgtSrcName、sgtDstName、description
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200、400、401、403、500

获取所有矩阵单元 API 调用的请求示例

```
GET /ers/config/egressmatrixcell HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
```

获取所有矩阵单元 API 调用的请求示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <resources>
        <ns2:resource description="description1" name="name1" id="id1"/>
        <ns2:resource description="description2" name="name2" id="id2"/>
    </resources>
</ns2:searchResult>
}
```

通过 ID 获取矩阵单元

下表列出通过 ID 获取矩阵单元 API 调用的主要特征：

表 7-74　通过 ID 获取矩阵单元 API 调用的主要特征

说明	检索指定的矩阵单元
摘要	GET /ers/config/egressmatrixcell/{id}

表 7-74　通过 ID 获取矩阵单元 API 调用的主要特征（续）

请求头	Accept、Authorization : Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	SGACL
响应状态	200，400，401，403，404，415，500

通过 ID 获取矩阵单元 API 调用的请求示例

```
GET /ers/config/egressmatrixcell/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.trustsec.egressmatrixcell.1.0+xml
```

通过 ID 获取矩阵单元 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.trustsec.egressmatrixcell.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:egressmatrixcell description="description" name="name" id="id"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .

</ns3:egressmatrixcell >
}
```

创建矩阵单元

下表列出创建矩阵单元 API 调用的主要特征：

表 7-75　创建矩阵单元 API 调用的主要特征

说明	创建矩阵单元
摘要	POST /ers/config/egressmatrixcell/
请求头	Content-Type、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	EgressMatrixCell
响应头	Location、Content-Type（错误 / 验证的 ERSResponse - 400）
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建矩阵单元 API 调用的请求示例

```
POST /ers/config/egressmatrixcell/ HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.egressmatrixcell.1.1+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:egressmatrixcell description="sgt description" name="newsgt"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="trustsec.ers.ise.cisco.com">
.
   .
   .



</ns3:egressmatrixcell >
```

创建矩阵单元 API 调用的响应示例

```
HTTP/1.1 201 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Location: https://<ISE-ADMIN-NODE>/ers/config/egressmatrixcell/333
```

更新矩阵单元

下表列出更新矩阵单元 API 调用的主要特征：

表 7-76　　更新矩阵单元 API 调用的主要特征

说明	更新指定的单元
摘要	PUT /ers/egressmatrixcell/{id}
请求头	Accept、Content-Type、Authorization: Basic、Host Content-Type: application/vnd.com.cisco.ise.trustsec. egressmatrixcell.1.0+xml Accept: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
查询字符串	N/A
请求消息正文	EgressMatrixCell
响应头	Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
响应消息正文	已更新的字段
响应状态	200，400，401，403，404，415，500

更新矩阵单元 API 调用的请求示例

```
PUT /ers/config/egressmatrixcell/333
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.egressmatrixcell.1.0+xml
Accept:
application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<ns3:egressmatrixcell description="description" name="name" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

    .
    .
    .


</ns3:egressmatrixcell >
```

更新矩阵单元 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns2:updatedFields
    xmlns:ns2="ers.ise.cisco.com">

    .
    .
    .


</ns2:updatedFields>
```

删除矩阵单元

下表列出删除矩阵单元 API 调用的主要特征：

表 7-77　　删除矩阵单元 API 调用的主要特征

说明	删除指定的单元
摘要	DELETE /ers/config/egressmatrixcell/{cell-id}
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除矩阵单元 API 调用的请求示例

```
DELETE /ers/config/egressmatrixcell/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxxx
```

删除矩阵单元 API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

设置矩阵状态

下表列出设置矩阵状态 API 调用的主要特征：

表 7-78　设置矩阵状态 API 调用的主要特征

说明	设置所有矩阵单元状态
摘要	PUT /ers/egressmatrixcell /status/{statustype} statustype: MONITOR | ENABLED | DISABLED
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

设置矩阵状态 API 调用的请求示例

```
PUT /ers/config/egressmatrixcell/status/ENABLED
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
```

设置矩阵状态 API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

克隆矩阵单元

下表列出克隆矩阵单元 API 调用的主要特征：

表 7-79　克隆矩阵单元 API 调用的主要特征

说明	将指定的单元 ID 克隆到指定的矩阵坐标。
摘要	PUT /ers/config/ egressmatrixcell /clonecell/{cell-to-be-cloned-id}/srcsgt/{src-sgt-id}/dstsgt/{dst-sgt-id}
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Type
响应消息正文	OperationResult 结果属性：Cloned-cell-id
响应状态	200，400，401，403，404，415，500

克隆矩阵单元 API 调用的请求示例

```
PUT /ers/config/egressmatrixcell/status/ENAB/ers/config/ egressmatrixcell
/clonecell/24355453/srcsgt/2456653456/dstsgt/545434565
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.operationresult.1.0+xml
```

克隆矩阵单元 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2014 23:59:59 GMT
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:operationResult xmlns:ns2="ers.ise.cisco.com">
    <attributesList>
        <attribute name="cloned-cell-id" value="23443543-er243523-wer433234

    </attributesList>
</ns2:operationResult>
}
```

清除矩阵单元

下表列出清除矩阵单元 API 调用的主要特征：

表 7-80　　清除矩阵单元 API 调用的主要特征

说明	清除矩阵单元
摘要	PUT /ers/egressmatrixcell/clear/{srcsgtid}/{dstsgtid}
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

清除矩阵单元 API 调用的请求示例

```
PUT /ers/config/egressmatrixcell/clear/{srcsgtid}/{dstsgtid}
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
```

清除矩阵单元 API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

适用于 SGMapping 的外部 RESTful 服务 API

下表列出适用于 SGMapping（IP 到 SGT 的映射）的外部 RESTful 服务 API：

表 7-81　适用于 SGMapping 的 API

操作	方法	URL	内容	查询字符串
获取所有映射	GET	/ers/config/sgmapping	不适用	page、size、sortacs、sortdsn、filter
通过 ID 获取映射	GET	/ers/config/sgmapping/{mapping-id}	不适用	
创建映射	POST	/ers/config/sgmapping/	SGMapping	
更新映射	PUT	/ers/config/sgmapping/{mapping-id}	SGMapping	
部署单一映射	PUT	/ers/config/sgmapping/{mapping-id}/deploy	基于设置部署指定的映射。	
部署所有映射	PUT	/ers/config/sgmapping/deployall	不适用	
获取部署状态	GET	/ers/config/sgmapping/deploy/status	OperationResult	
批量启动	PUT	/ers/config/sgmapping/bulk	BulkRequest	
获取版本	GET	/ers/config/sgmapping/versioninfo	不适用	

获取所有 SGMapping

下表列出获取所有 SGMapping API 调用的主要特征：

表 7-82　获取所有 SGMapping API 调用的主要特征

说明	检索 SGMapping 的集合
摘要	GET /ers/config/sgmapping
请求头	Accept、Authorization: Basic、Host Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
查询字符串	start、size、sortasc、sortdsc
过滤器字段	Name、Description、SGT Name、DeployType、DeployTo、HostName、HostIp
排序字段	Name、Description、SGT Name、DeployType、DeployTo、HostName、HostIp
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有 SGMapping API 调用的请求示例

```
GET /ers/config/sgmapping HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
```

获取所有 SGMapping API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <resources>
        <ns2:resource description="description1" name="host1" id="id1"/>
        <ns2:resource description="description2" name="1.2.3.4" id="id2"/>
    </resources>
</ns2:searchResult>
}
```

通过 ID 获取 SGMapping

下表列出通过 ID 获取 SGMapping API 调用的主要特征：

表 7-83　　通过 ID 获取 SGMapping API 调用的主要特征

说明	检索指定的 SGMapping
摘要	GET /ers/config/sgmapping/{id}
请求头	Accept、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	SGMapping
响应状态	200，400，401，403，404，415，500

通过 ID 获取 SGMapping API 调用的请求示例

```
GET /ers/config/sgmapping/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.trustsec.sgmapping.1.0+xml
```

通过 ID 获取 SGMapping API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.trustsec.sgmappinggroup.1.0+xml
Content-Length: 16347
{
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgmapping description="description" name="name" id="id" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .




</ns3:sgmapping >
}
```

创建 SGMapping

下表列出创建 SGMapping API 调用的主要特征：

表 7-84　　创建 SGMapping API 调用的主要特征

说明	创建 SGMapping
摘要	POST /ers/config/sgmapping/
请求头	Content-Type、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	SGMapping
响应头	Location、Content-Type（错误 / 验证的 ERSResponse - 400）
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SGMapping API 调用的请求示例

```
POST /ers/config/sgmapping/ HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgmapping.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgmapping description="sgt description" name="newsgt" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .




</ns3:sgmapping>
```

创建 SGMapping API 调用的响应示例

```
HTTP/1.1 201 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Location: https://<ISE-ADMIN-NODE>/ers/config/sgmapping/333
```

更新 SGMapping

下表列出更新 SGMapping API 调用的主要特征：

表 7-85　更新 SGMapping API 调用的主要特征

说明	更新指定的 SGMapping
摘要	PUT /ers/config/sgmapping/{id}
请求头	Accept、Content-Type、Authorization: Basic、Host Content-Type: application/vnd.com.cisco.ise.trustsec. sgmapping.1.0+xml Accept: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
查询字符串	N/A
请求消息正文	SGMapping
响应头	Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
响应消息正文	已更新的字段
响应状态	200，400，401，403，404，415，500

更新 SGMapping API 调用的请求示例

```
PUT /ers/config/sgmapping/333
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgmapping.1.0+xml
Accept:
application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgmapping description="description" name="name" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .


</ns3:sgmapping>
```

更新 SGMapping API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns2:updatedFields
    xmlns:ns2="ers.ise.cisco.com">
.
   .
   .


</ns2:updatedFields>
```

删除 SGMapping

下表列出删除 SGMapping API 调用的主要特征：

表 7-86　删除 SGMapping API 调用的主要特征

说明	删除指定的 SGMapping
摘要	DELETE /ers/config/sgmapping/{id}
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除 SGMapping API 调用的请求示例

```
DELETE /ers/config/sgmapping/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxxx
```

更新 SGMapping API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

部署单一 SGMapping

下表列出部署单一 SGMapping API 调用的主要特征：

表 7-87　部署单一 SGMapping API 调用的主要特征

说明	部署指定的 SGMapping
摘要	PUT /ers/config/sgmapping/{maping-group-id}/deploy
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	202、400、401、403、404、500、503

部署单一 SGMapping API 调用的请求示例

```
PUT /ers/config/sgmapping/{mapping-id}/deploy
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxxx
```

部署单一 SGMapping API 调用的响应示例

```
HTTP/1.1 202 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

部署所有 SGMapping

下表列出部署所有 SGMapping API 调用的主要特征：

表 7-88　部署所有 SGMapping API 调用的主要特征

说明	部署所有 SGMapping
摘要	PUT /ers/config/sgmapping/deployall
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	202、400、401、403、404、500、503

部署所有 SGMapping API 调用的请求示例

```
PUT /ers/config/sgmapping/deployall
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
```

部署所有 SGMapping API 调用的响应示例

```
HTTP/1.1 202 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

获取部署状态

下表列出获取部署状态 API 调用的主要特征：

表 7-89　获取部署状态 API 调用的主要特征

说明	获取活动部署状态
摘要	GET /ers/config/sgmapping/deploy/status
请求头	Accept、Authorization
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type

表 7-89　获取部署状态 API 调用的主要特征（续）

响应消息正文	具有以下属性的 OperationResponse： DeviceCount CompleteCount
响应状态	200、400、401、403、404、500

获取部署状态 API 调用的请求示例

```
PUT /ers/config/sgmapping/deploy/status HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.operationresult.1.0+xml
```

获取部署状态 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2014 23:59:59 GMT
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:operationResult xmlns:ns2="ers.ise.cisco.com">
    <attributesList>
        <attribute name="DeviceCount" value="137"/>
<attribute name ="CompleteCount" value ="19"/>
    </attributesList>
</ns2:operationResult>
}
```

适用于 SGMappingGroup 的外部 RESTful 服务 API

下表列出适用于 SGMappingGroup 的外部 RESTful 服务 API：

表 7-90　适用于 SGMappingGroup 的 API

操作	方法	URL	内容/coomment	查询字符串
获取所有映射	GET	/ers/config/sgmappinggroup	不适用	page、size、sortacs、sortdsn、filter
通过 ID 获取映射	GET	/ers/config/sgmappinggroup/{group-id}	不适用	
创建映射	POST	/ers/config/sgmappinggroup/	SGMappingGroup	
更新映射	PUT	/ers/config/sgmappinggroup/{mapping-group-id}	SGMappingGroup	
部署单一映射	PUT	/ers/config/sgmappinggroup/{group-id}/deploy	基于设置部署选定组。	
部署所有映射	PUT	/ers/config/sgmappinggroup/deployall	部署所有组	
获取部署状态	GET	/ers/config/sgmappinggroup/deploy/status	OperationResult	

获取所有 SGMappingGroup

下表列出获取所有 SGMappingGroup API 调用的主要特征：

表 7-91　获取所有 SGMappingGroup API 调用的主要特征

说明	搜索 SGMappingGroup
摘要	GET /ers/config/sgmappinggroup
请求头	Accept、Authorization: Basic、Host Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
查询字符串	start、size、sortasc、sortdsc
过滤器字段	Name、Description、SGT Name、DeployType、DeployTo
排序字段	Name、Description、SGT Name、DeployType、DeployTo
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	搜索结果
响应状态	200，400，401，403，404，415，500

获取所有 SGMappingGroup API 调用的请求示例

```
GET /ers/config/sgmappinggroup HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
```

获取所有 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.searchresult.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <resources>
        <ns2:resource description="group1" name="group1" id="id1"/>
        <ns2:resource description="group2" name="group2" id="id2"/>
    </resources>
</ns2:searchResult>
}
```

通过 ID 获取 SGMappingGroup

下表列出通过 ID 获取 SGMappingGroup API 调用的主要特征：

表 7-92　通过 ID 获取 SGMappingGroup API 调用的主要特征

说明	检索指定的 SGMappingGroup
摘要	GET /ers/config/sgmappinggroup/{id}

表 7-92　通过 ID 获取 SGMappingGroup API 调用的主要特征（续）

请求头	Accept、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	SGMapping
响应状态	200，400，401，403，404，415，500

通过 ID 获取 SGMappingGroup API 调用的请求示例

```
GET /ers/config/sgmappinggroup/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.trustsec.sgmappinggroup.1.0+xml
```

通过 ID 获取 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.trustsec.sgmappinggroup.1.0+xml
Content-Length: 16347
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgmappinggroup description="description" name="name" id="id"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .
</ns3:sgmappinggroup >
}
```

创建 SGMappingGroup

下表列出创建 SGMappingGroup API 调用的主要特征：

表 7-93　创建 SGMappingGroup API 调用的主要特征

说明	创建 SGMappingGroup
摘要	POST /ers/config/sgmappinggroup/
请求头	Content-Type、Authorization: Basic、Host
查询字符串	N/A
请求消息正文	SGMapping
响应头	Location、Content-Type（错误 / 验证的 ERSResponse - 400）
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SGMappingGroup API 调用的请求示例

```
POST /ers/config/sgmappinggroup/ HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgmappinggroup.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgmappinggroup description="sgt description" name="newsgt"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="trustsec.ers.ise.cisco.com">
.
   .
   .



</ns3:sgmappinggroup>
```

创建 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 201 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Location: https://172.21.76.94/ers/config/sgmappinggroup/333
```

更新 SGMappingGroup

下表列出更新 SGMappingGroup API 调用的主要特征：

表 7-94　　更新 SGMappingGroup API 调用的主要特征

说明	更新指定的 SGMappingGroup
摘要	PUT /ers/config/sgmappinggroup/{id}
请求头	Accept、Content-Type、Authorization: Basic、Host Content-Type: application/vnd.com.cisco.ise.trustsec. sgmappinggropup.1.0+xml Accept: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
查询字符串	N/A
请求消息正文	SGMappingGroup
响应头	Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
响应消息正文	已更新的字段
响应状态	200，400，401，403，404，415，500

更新 SGMappingGroup API 调用的请求示例

```
PUT /ers/config/sgmappinggroup/333
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Content-Type: application/vnd.com.cisco.ise.trustsec.sgmappinggroup.1.0+xml
Accept:
application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<ns3:sgmappinggroup description="description" name="name" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="trustsec.ers.ise.cisco.com">

   .
   .
   .


</ns3:sgmappinggroup>
```

更新 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
Content-Type: application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml
Content-Length: 529
{
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns2:updatedFields
    xmlns:ns2="ers.ise.cisco.com">

   .
   .
   .
</ns2:updatedFields>
```

删除 SGMappingGroup

下表列出删除 SGMappingGroup API 调用的主要特征：

表 7-95　　删除 SGMappingGroup API 调用的主要特征

说明	删除指定的 SGMappingGroup
摘要	DELETE /ers/config/sgmappinggroup/{id}
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除 SGMappingGroup API 调用的请求示例

```
DELETE /ers/config/sgmappinggroup/333 HTTP/1.1
Host: {ise-node-ip}
Authorization: Basic xxxxxxxxxxxxxxxxxxx
```

删除 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 204 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

部署单一 SGMappingGroup

下表列出部署单一 SGMappingGroup API 调用的主要特征：

表 7-96　部署单一 SGMappingGroup API 调用的主要特征

说明	部署指定的 SGMappingGroup
摘要	PUT /ers/config/sgmappinggroup/{mapping-group-id}/deploy
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	202、400、401、403、404、500、503

部署单一 SGMappingGroup API 调用的请求示例

```
PUT /ers/config/sgmappinggroup/{mapping-id}/deploy
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
```

部署单一 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 202 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

部署所有 SGMappingGroup

下表列出部署所有 SGMappingGroup API 调用的主要特征：

表 7-97　部署所有 SGMappingGroup API 调用的主要特征

说明	部署所有 SGMappingGroup
摘要	PUT /ers/config/sgmappinggroup/deployall
请求头	Authorization: Basic、Host
查询字符串	N/A
请求消息正文	N/A
响应头	N/A
响应消息正文	N/A
响应状态	202、400、401、403、404、500、503

部署所有 SGMappingGroup API 调用的请求示例

```
PUT /ers/config/sgmappinggroup/deployall
HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
```

部署所有 SGMappingGroup API 调用的响应示例

```
HTTP/1.1 202 OK
Date: Thu, 12 Jul 2012 23:59:59 GMT
```

获取部署状态

下表列出获取部署状态 API 调用的主要特征：

表 7-98　　获取部署状态 API 调用的主要特征

说明	获取部署状态
摘要	GET /ers/config/sgmappinggroup/deploy/status
请求头	Accept、Authorization
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	具有以下属性的 OperationResponse： DeviceCount CompleteCount
响应状态	200、400、401、403、404、500

获取部署状态 API 调用的请求示例

```
PUT /ers/config/sgmappinggroup/deploy/status HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.ers.operationresult.1.0+xml
```

获取部署状态 API 调用的响应示例

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2014 23:59:59 GMT
{
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:operationResult xmlns:ns2="ers.ise.cisco.com">
    <attributesList>
        <attribute name="DeviceCount" value="137"/>
<attribute name ="CompleteCount" value ="19"/>
    </attributesList>
</ns2:operationResult>
}
```

适用于 SXP 本地绑定的外部 RESTful 服务 API

以下 API 适用于 SXP 本地绑定：

- 创建 SXP 本地绑定
- 更新 SXP 本地绑定
- 删除 SXP 本地绑定
- 创建 SXP 对等体
- 更新 SXP 对等体
- 删除 SXP 对等体
- 创建 SXP 连接
- 更新 SXP 连接
- 删除 SXP 连接

创建 SXP 本地绑定

下表列出创建 SXP 本地绑定 API 调用的主要特征：

表 7-99　　创建 SXP 本地绑定 API 调用的主要特征

说明	创建 SXP 本地绑定
摘要	POST /ers/config/sxplocalbindings
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	ERSSxpLocalBinding
响应头	Content-Length、Content-Type、Location
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SXP 本地绑定 API 调用的请求示例

```
Method: POST
URI: https://<ISE-ADMIN-NODE>/ers/config/sxplocalbindings
HTTP Content-Type header.
application/vnd.com.cisco.ise.sxp.sxplocalbindings.1.0+xml; charset=utf-8 Bulk Support:
Operation 'Create' can be used within Bulk Request.
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sxplocalbindings xmlns:ns2="ers.ise.cisco.com">
<groupid>Group Id</groupid>
<ipAddressOrHost>ipAddressOrHost</ipAddressOrHost>
<sgtid>Sgt Id</sgtid>
</sxplocalbindings>
```

创建 SXP 本地绑定 API 调用的响应示例

```
HTTP Status: 201 (Created)
Content:
N/A
```

更新 SXP 本地绑定

下表列出更新 SXP 本地绑定 API 调用的主要特征：

表 7-100　更新 SXP 本地绑定 API 调用的主要特征

说明	更新 SXP 本地绑定
摘要	PUT /ers/config/sxplocalbindings/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	ERSSxpLocalBinding
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，404，415，500

更新 SXP 本地绑定 API 调用的请求示例

```
Method: PUT
URI: https://<ISE-ADMIN-NODE>/ers/config/sxplocalbindings/{id}
HTTP Content-Type header.
application/vnd.com.cisco.ise.sxp.sxplocalbindings.1.0+xml; charset=utf-8
Bulk Support: Operation 'Update' can be used within Bulk Request.
Request Content:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sxplocalbindings xmlns:ns2="ers.ise.cisco.com.">
<groupid>Group Id</groupid>
<ipAddressOrHost>ipAddressOrHost</ipAddressOrHost>
<sgtid>Sgt Id</sgtid>
</sxplocalbindings>
```

更新 SXP 本地绑定 API 调用的响应示例

```
HTTP Status: 200 (OK)
Content:

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
<updatedField field="field1">
<newvalue>val_new</newvalue>
<oldValue>val_old</oldValue>
</updatedField>
<updatedField field="some other field">
<newvalue>val_new</newvalue>
<oldValue>val_old</oldValue>
</updatedField>
</ns2:updatedFields>
```

删除 SXP 本地绑定

下表列出删除 SXP 本地绑定 API 调用的主要特征：

表 7-101　删除 SXP 本地绑定 API 调用的主要特征

说明	删除 SXP 本地绑定
摘要	DELETE /ers/config/sxplocalbindings/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除 SXP 本地绑定 API 调用的请求示例

```
Method: DELETE
URI: https://<ISE-ADMIN-NODE>/ers/config/sxplocalbindings/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.sxp.sxplocalbindings.1.0+xml
Bulk Support: Operation 'Delete' can be used within Bulk Request.
Request Content:
N/A
```

删除 SXP 本地绑定 API 调用的响应示例

```
HTTP Status: 204 (No Content)
Content:
N/A
```

创建 SXP 对等体

下表列出创建 SXP 对等体 API 调用的主要特征：

表 7-102　创建 SXP 对等体 API 调用的主要特征

说明	创建 SXP 对等网络
摘要	POST /ers/config/sxppeers
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	ERSSxpPeer
响应头	Content-Length、Content-Type、Location
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SXP 对等体 API 调用的请求示例

```
Method: POST
URI: https://<ISE-ADMIN-NODE>/ers/config/sxppeers
HTTP Content-Type header:
application/vnd.com.cisco.ise.sxp.sxppeers.1.0+xml;charset=utf-8
Bulk Support: Operation 'Create' can be used within Bulk Request.
Request Content:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sxppeers xmlns:ns2="ers.ise.cisco.com">
<peerIpAddress>IP address of the peer</peerIpAddress>
<peerName>Peer Name</peerName>
</sxppeers>
```

创建 SXP 对等体 API 调用的响应示例

```
HTTP Status: 201 (Created)
Content:
N/A
```

更新 SXP 对等体

下表列出更新 SXP 对等体 API 调用的主要特征：

表 7-103　　更新 SXP 对等体 API 调用的主要特征

说明	更新指定的 SXP 对等体
摘要	PUT /ers/config/sxppeers/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	ERSSxpPeer
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，404，415，500

更新 SXP 对等体 API 调用的请求示例

```
Method: PUT
URI: https://<ISE-ADMIN-NODE>/ers/config/sxppeers/{id}
HTTP Content-Type header:
application/vnd.com.cisco.ise.sxp.sxppeers.1.0+xml; charset=utf-8
Bulk Support: Operation 'Update' can be used within Bulk Request.
Request Content:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sxppeers xmlns:ns2="ers.ise.cisco.com">
<peeripAddress>IP address of the peer</peerIpAddress>
<peerName>Peer Name</peerName>
</sxppeers>
```

更新 SXP 对等体 API 调用的响应示例

```
HTTP Status: 200 (OK)
Content:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
<updatedField field="field1">
<newvalue>val_new</newvalue>
<oldValue>val_old</oldValue>
</updatedField>
<updatedField field="some other field">
<newvalue>val_new</newvalue>
<oldValue>val_old</oldValue>
</updatedField>
</ns2:updatedFields>
```

删除 SXP 对等体

下表列出删除 SXP 对等体 API 调用的主要特征：

表 7-104　　删除 SXP 对等体 API 调用的主要特征

说明	删除指定的 SXP 对等体
摘要	DELETE /ers/config/sxppeers/{id}
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除 SXP 对等体 API 调用的请求示例

```
Method: DELETE
URI: https://<ISE-ADMIN-NODE>/ers/config/sxppeers/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.sxp.sxplocalbindings.1.0+xml
Bulk Support: Operation 'Delete' can be used within Bulk Request.
Request Content:
N/A
```

删除 SXP 对等体 API 调用的响应示例

```
HTTP Status: 204 (No Content)
Content:
N/A
```

创建 SXP 连接

下表列出创建 SXP 连接 API 调用的主要特征：

表 7-105　创建 SXP 连接 API 调用的主要特征

说明	创建 SXP 连接
摘要	POST /ers/config/sxpconnections
请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	ERSSxpConnection
响应头	Content-Length、Content-Type、Location
响应消息正文	N/A
响应状态	201，400，401，403，415，500

创建 SXP 连接 API 调用的请求示例

```
Method: POST
URI: https://<ISE-ADMIN-NODE>/ers/config/sxpconnections
HTTP Content-Type header:
application/vnd.com.cisco.ise.sxp.sxpconnections.1.0+xml;charset=utf-8
Bulk Support: Operation 'Create' can be used within Bulk Request.
Request Content:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sxpconnections description="Brief description about network peer"
xmlns:ns2="ers.ise.cisco.com">
<password><encryptedValue>HUQX51QePOvewsQw+OGoxSKLZwn7Mpmj</encryptedValue>
</password>
<peerId>Peer Id</peerId>
<sourceId>Source Id</sourceId>
<sxpMode>LISTENER</sxpMode>
<sxpVersion>VERSION_2</sxpVersion>
<topPort>64999</topPort>
</sxpconnections>
```

创建 SXP 连接 API 调用的响应示例

```
HTTP Status: 201 (Created)
Content:
N/A
```

更新 SXP 连接

下表列出更新 SXP 连接 API 调用的主要特征：

表 7-106　更新 SXP 连接 API 调用的主要特征

说明	更新指定的 SXP 对等连接
摘要	PUT /ers/config/sxpconnections/{id}
请求头	Accept、Authorization、Host

表 7-106　更新 SXP 连接 API 调用的主要特征（续）

查询字符串	N/A
请求消息正文	ERSSxpConnection
响应头	Content-Length、Content-Type
响应消息正文	更新字段的列表
响应状态	200，400，401，403，404，415，500

更新 SXP 连接 API 调用的请求示例

```
Method: PUT
URI: https://<ISE-ADMIN-NODE>/ers/config/sxpconnections/{id}
HTTP Content-Type header:
application/vnd.com.cisco.ise.sxp.sxpconnections.1.0+xml;charset=utf-8
Bulk Support: Operation 'Update' can be used within Bulk Request.
Request Content:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sxpconnections description="Brief description about network peer"
xmlns:ns2="ers.ise.cisco.com">
<groupId>Group Id to which the peer belongs</groupId>
<password><encryptedValue>HUQX51QePOvewsQw+OGoxSKLZwn7Mpmj</encryptedValue>
</password>
<peerId>Peer Id</peerId>
<sourceId>Source Id</sourceId>
<sxpMode>LISTENER</sxpMode>
<sxpVersion>VERSION_2</sxpVersion>
<topPort>64999</topPort>
</sxpconnections>
```

更新 SXP 连接 API 调用的响应示例

```
HTTP Status: 200 (OK)
Content:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
<updatedField field="field1">
<newvalue>val_new</newvalue>
<oldValue>val_old</oldValue>
</updatedField>
<updatedField field="some other field">
<newvalue>val_new</newvalue>
<oldValue>val_old</oldValue>
</updatedField>
</ns2:updatedFields>
```

删除 SXP 连接

下表列出删除 SXP 连接 API 调用的主要特征：

表 7-107　删除 SXP 连接 API 调用的主要特征

说明	删除指定的 SXP 对等连接
摘要	DELETE /ers/config/sxpconnections/{id}

表 7-107　删除 SXP 连接 API 调用的主要特征（续）

请求头	Accept、Authorization、Host
查询字符串	N/A
请求消息正文	N/A
响应头	Content-Length、Content-Type
响应消息正文	N/A
响应状态	200，204，400，401，403，404，415，500

删除 SXP 连接 API 调用的请求示例

```
Method: DELETE
URI: https://<ISE-ADMIN-NODE>/ers/config/sxpconnections/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.sxp.sxpconnections.1.0+xml
Bulk Support: Operation 'Delete' can be used within Bulk Request.
Request Content:
N/A
```

删除 SXP 连接 API 调用的响应示例

```
HTTP Status: 204 (No Content)
Content:
N/A
```

REST API 客户端

通过外部 RESTful 服务 API，您可以对 Cisco ISE 资源执行 CRUD（创建、读取、更新、删除）操作。要使用与 Cisco ISE 服务器通信并在其上执行操作的外部 RESTful 服务 API 构建和测试应用，您可以使用任何行业标准 REST API 客户端，如 Google Chrome 的 POSTMAN 插件。

POSTMAN 根据 REST 架构和基本原则进行设计，使您可以利用 Google Chrome Web 浏览器发送并检索标准 HTTP 和 HTTPS 请求和响应。您可以使用以下标准 HTTP 方法对 Cisco ISE 资源执行 CRUD 操作：

- GET
- POST
- PUT
- DELETE

通过 ERS API，您可以在各种 API 调用中使用这些 HTTP 请求，从而使您可以在 Cisco ISE 服务器上执行操作。有关使用这些 HTTP 请求的操作的完整列表，请参阅 <ERS API 操作 >。

备注　要下载 POSTMAN 插件，请访问 https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en。有关使用 POSTMAN 插件的详细信息，请访问 https://github.com/a85/POSTMan-Chrome-Extension/wiki。

GET 方法

请求指定资源的表示。使用 GET 的请求只会检索数据，不会产生任何其他影响。

备注 本节介绍如何使用 POSTMAN 插件发起 ERS API 调用。此 API 调用使用 GET HTTP 方法和 ERS API 的其他部分（本节不做介绍）。有关各个 ERS API 部分（如特征、请求和响应）的详细信息，请参阅外部 RESTful 服务 API 操作。

使用 GET HTTPS 方法的 ERS API 调用的请求正文包括以下三个构成要素：

- URI
- Accept 标头
- Authorization 标头

URI

GET 方法向 Cisco ISE 服务器发送 URI，HTTP 回复为原始结果数据。典型的 URI 必须遵守以下格式：

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resouce Name>*

  其中，*<Cisco ISE Server Address>* 表示 Cisco ISE 服务器的服务器地址，*<port>* 表示端口 9060，*<namespace>* 表示 ISE 资源所属的命名空间，*<Cisco ISE Resource Name>* 表示 Cisco ISE 资源的名称。

以下示例显示的 URI 用于请求 *interaluser* ISE 资源的数据：

- *https://10.56.13.196:9060/ers/config/internaluser*。

备注 URI 不是请求正文；它只是一个 URL。此 URL 使用 GET 方法发送到服务器。

Accept 标头

Accept 标头必须遵守以下格式：

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

其中，*<resource-namespace>* 表示 ISE 资源所属的命名空间，*<resource-type>* 表示 ISE 资源的类型，*<major-version>* 表示 ISE 部署的主要版本号，*<minor-version>* 表示 ISE 部署的次要版本号。

以下示例显示典型的 Accept 标头：

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization 标头

Authorization 标头包含嵌入到 GET 请求中的加密授权密钥。指定授权凭证后，您必须生成加密密钥，此加密密钥之后将嵌入到请求正文中。

备注 有关生成加密密钥的详细信息，请参阅使用 POSTMAN 提出 GET 请求（第 7-93 页）。

使用 POSTMAN 提出 GET 请求

操作步骤

步骤 1　在 Google Chrome 浏览器中打开 POSTMAN 插件。

步骤 2　使用左侧窗格中的选项创建新集合。

注　有关使用 POSTMAN 插件的详细信息，请访问 https://github.com/a85/POSTMan-Chrome-Extension/wiki。

步骤 3　从下拉菜单中，选择 **GET**。

步骤 4　在 URL 栏，输入 URI。

URI 指定您尝试与之通信的 Cisco ISE 服务器和您尝试访问的 ISE 资源。有关 URI 格式的详细信息，请参阅 URI（第 7-92 页）。

步骤 5　点击 **Basic Auth** 选项卡。

通过此选项，您可以指定显示的用户访问凭证。

步骤 6　在 Username 和 Password 字段指定您的访问凭证，然后点击 **Refresh Headers**。

POSTMAN 会显示 Authorization 标头与加密密钥。

步骤 7　通过指定以下值添加 Accept 标头：application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

注　有关 Accept 标头的详细信息，请参阅 Accept 标头（第 7-92 页）。

步骤 8　点击 **Send**。

POSTMAN 插件会显示 200 OK 状态响应，表示请求成功。此请求也会返回您在 URL 指定的资源的详细信息。

POST 方法

请求服务器接受请求中包含的实体作为 URI 所标识 Web 资源的新附属资源。

备注　本节介绍如何使用 POSTMAN 插件发起 ERS API 调用。此 API 调用使用 POST HTTP 方法和 ERS API 的其他部分（本节不做介绍）。有关各个 ERS API 部分（如特征、请求和响应）的详细信息，请参阅外部 RESTful 服务 API 操作。

使用 POST HTTP 方法的 ERS API 调用的请求正文包含以下三个构成要素：

- URI
- Content-Type 标头
- Authorization 标头

URI

POST 方法向 Cisco ISE 服务器发送 URI。典型的 URI 必须遵守以下格式：

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resouce Name>*

  其中，*<Cisco ISE Server Address>* 表示 Cisco ISE 服务器的服务器地址，*<port>* 表示端口 9060，*<namespace>* 表示 ISE 资源所属的命名空间，*<Cisco ISE Resource Name>* 表示 Cisco ISE 资源的名称。

以下示例显示的 URI 用于请求 *interaluser* ISE 资源的数据：

- *https://10.56.13.196:9060/ers/config/internaluser*。

备注　URI 不是请求正文；它只是一个 URL。此 URL 使用 POST 方法发送到服务器。

Content-Type 标头

Content-Type 标头必须遵守以下格式：

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

  其中，*<resource-namespace>* 表示 ISE 资源所属的命名空间，*<resource-type>* 表示 ISE 资源的类型，*<major-version>* 表示 ISE 部署的主要版本号，*<minor-version>* 表示 ISE 部署的次要版本号。

以下示例显示典型的 Accept 标头：

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization 标头

Authorization 标头包含嵌入到 POST 请求中的加密授权密钥。指定授权凭证后，您必须生成加密密钥，此加密密钥之后将嵌入到请求正文中。

备注　有关生成加密密钥的详细信息，请参阅使用 POSTMAN 提出 POST 请求（第 7-94 页）。

使用 POSTMAN 提出 POST 请求

操作步骤

步骤 1　在 Google Chrome 浏览器中打开 POSTMAN 插件。

步骤 2　使用左侧窗格中的选项创建新集合。

注　有关使用 POSTMAN 插件的详细信息，请访问 https://github.com/a85/POSTMan-Chrome-Extension/wiki。

步骤 3　从下拉菜单中，选择 **POST**。

步骤 4　在 URI 栏，输入 URI。

URI 指定您尝试与之通信的 Cisco ISE 服务器和您尝试访问的 ISE 资源。有关 URI 格式的详细信息，请参阅 URI（第 7-94 页）。

步骤 5　点击 **Basic Auth** 选项卡。

通过此选项，您可以指定显示的用户访问凭证。

步骤 6　在 Username 和 Password 字段指定您的访问凭证，然后点击 **Refresh Headers**。

POSTMAN 会显示 Authorization 标头与加密密钥。

步骤 7　通过指定以下值添加 Content-Type 标头：application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

注　有关 Accept 标头的详细信息，请参阅 Content-Type 标头（第 7-94 页）。

步骤 8　从显示在 raw 按钮旁边的下拉菜单，选择 **XML**。

步骤 9　点击 **raw**。

步骤 10　POSTMAN 插件会打开一个编辑窗格，您可以利用此窗格指定 POST 请求的正文。

步骤 11　在编辑窗格中输入 POST 请求的消息正文。

备注　此消息正文必须包含您尝试在 ISE 服务器上创建的 ISE 资源对应的详细信息。例如，创建内部 interaluser 时，您必须指定详细信息，如 interaluser 的名称、interaluser 的说明、密码等。有关使用 POST 请求的 ERS API 的消息正文的详细信息，以及需要指定的 ISE 资源的详细信息，请参阅外部 RESTful 服务 API 操作。

步骤 12　点击 **Send**。

POSTMAN 插件会显示 201 CREATED 状态响应，表示请求成功。您可以转到 ISE GUI 以验证您添加的 ISE 资源是否显示在 ISE GUI 中。

PUT 方法

请求包含的实体存储在提供的 URI 下。如果 URI 指向已经存在的现有资源，系统将修改此资源；如果 URI 指向的不是现有资源，服务器可以使用此 URI 创建资源。

备注　本节介绍如何使用 POSTMAN 插件发起 ERS API 调用。此 API 调用使用 PUT HTTP 方法和 ERS API 的其他部分（本节不做介绍）。有关各个 ERS API 部分（如特征、请求和响应）的详细信息，请参阅外部 RESTful 服务 API 操作。

使用 POST HTTP 方法的 ERS API 调用的请求正文包含以下三个构成要素：

- URI
- Content-Type 标头
- Authorization 标头

URI

PUT 方法向 Cisco ISE 服务器发送 URI。典型的 URI 必须遵守以下格式：

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resouce Name>*

  其中，*<Cisco ISE Server Address>* 表示 Cisco ISE 服务器的服务器地址，*<port>* 表示端口 9060，*<namespace>* 表示 ISE 资源所属的命名空间，*<Cisco ISE Resource Name>* 表示 Cisco ISE 资源的名称。

以下示例显示的 URI 用于请求 *interaluser* ISE 资源的数据：

- *https://10.56.13.196:9060/ers/config/internaluser*。

备注　URI 不是请求正文；它只是一个 URL。此 URL 使用 PUT 方法发送到服务器。

Content-Type 标头

Content-Type 标头必须遵守以下格式：

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

  其中，*<resource-namespace>* 表示 ISE 资源所属的命名空间，*<resource-type>* 表示 ISE 资源的类型，*<major-version>* 表示 ISE 部署的主要版本号，*<minor-version>* 表示 ISE 部署的次要版本号。

以下示例显示典型的 Accept 标头：

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization 标头

Authorization 标头包含嵌入到 PUT 请求中的加密授权密钥。指定授权凭证后，您必须生成加密密钥，此加密密钥之后将嵌入到请求正文中。

备注　有关生成加密密钥的详细信息，请参阅使用 POSTMAN 提出 PUT 请求（第 7-96 页）。

使用 POSTMAN 提出 PUT 请求

操作步骤

步骤 1　在 Google Chrome 浏览器中打开 POSTMAN 插件。

步骤 2　使用左侧窗格中的选项创建新集合。

注　有关使用 POSTMAN 插件的详细信息，请访问 https://github.com/a85/POSTMan-Chrome-Extension/wiki。

步骤 3　从下拉菜单中，选择 **PUT**。

步骤 4　在 URI 栏，输入 URI。

URI 指定您尝试与之通信的 Cisco ISE 服务器和您尝试访问的 ISE 资源。有关 URI 格式的详细信息，请参阅 URI（第 7-96 页）。

步骤 5　点击 **Basic Auth** 选项卡。

通过此选项，您可以指定显示的用户访问凭证。

步骤 6　在 Username 和 Password 字段指定您的访问凭证，然后点击 **Refresh Headers**。

POSTMAN 会显示 Authorization 标头与加密密钥。

步骤 7　通过指定以下值添加 Content-Type 标头：application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

注　有关 Accept 标头的详细信息，请参阅 Content-Type 标头（第 7-96 页）。

步骤 8　从显示在 raw 按钮旁边的下拉菜单，选择 **XML**。

步骤 9　点击 **raw**。

步骤 10　POSTMAN 插件会打开一个编辑窗格，您可以利用此窗格指定 POST 请求的正文。

步骤 11　在编辑窗格中输入 POST 请求的消息正文。

备注　此消息正文必须包含您尝试在 ISE 服务器上更新的 ISE 资源对应的详细信息。例如，更新 interaluser 时，必须指定详细信息，如 interaluser 的名称、interaluser 的说明、密码等。有关使用 POST 请求的 ERS API 的消息正文的详细信息，以及需要指定的 ISE 资源的详细信息，请参阅外部 RESTful 服务 API 操作。

步骤 12　点击 **Send**。

POSTMAN 插件会显示 201 CREATED 状态响应，表示请求成功。您可以转到 ISE GUI 以验证您添加的 ISE 资源是否显示在 ISE GUI 中。

Delete 方法

删除指定资源。

备注　本节介绍如何使用 POSTMAN 插件发起 ERS API 调用。此 API 调用使用 DELETE HTTP 方法和 ERS API 的其他部分（本节不做介绍）。有关各个 ERS API 部分（如特征、请求和响应）的详细信息，请参阅外部 RESTful 服务 API 操作。

使用 DELETE HTTP 方法的 ERS API 调用的请求正文包含以下三个构成要素：

- URI
- Accept 标头
- Authorization 标头

URI

DELETE 方法向 Cisco ISE 服务器发送 URI。典型的 URI 必须遵守以下格式：

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resouce Name>*

  其中，*<Cisco ISE Server Address>* 表示 Cisco ISE 服务器的服务器地址，*<port>* 表示端口 9060，*<namespace>* 表示 ISE 资源所属的命名空间，*<Cisco ISE Resource Name>* 表示 Cisco ISE 资源的名称。

以下示例显示的 URI 用于请求 *interaluser* ISE 资源的数据：

- *https://10.56.13.196:9060/ers/config/internaluser*。

备注　URI 不是请求正文；它只是一个 URL。此 URL 使用 GET 方法发送到服务器。

Accept 标头

Accept 标头必须遵守以下格式：

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

其中，*<resource-namespace>* 表示 ISE 资源所属的命名空间，*<resource-type>* 表示 ISE 资源的类型，*<major-version>* 表示 ISE 部署的主要版本号，*<minor-version>* 表示 ISE 部署的次要版本号。

以下示例显示典型的 Accept 标头：

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization 标头

Authorization 标头包含嵌入到 DELETE 请求中的加密授权密钥。指定授权凭证后，您必须生成加密密钥，此加密密钥之后将嵌入到请求正文中。

备注　有关生成加密密钥的详细信息，请参阅使用 POSTMAN 提出 DELETE 请求（第 7-98 页）。

使用 POSTMAN 提出 DELETE 请求

操作步骤

步骤 1　在 Google Chrome 浏览器中打开 POSTMAN 插件。

步骤 2　使用左侧窗格中的选项创建新集合。

注　有关使用 POSTMAN 插件的详细信息，请访问 https://github.com/a85/POSTMan-Chrome-Extension/wiki。

步骤 3　从下拉菜单中，选择 **DELETE**。

步骤 4　在 URL 栏，输入 URI。

URI 指定您尝试与之通信的 Cisco ISE 服务器和您尝试访问的 ISE 资源。有关 URI 格式的详细信息，请参阅 URI（第 7-98 页）。

步骤 5　点击 **Basic Auth** 选项卡。

通过此选项，您可以指定显示的用户访问凭证。

步骤 6　在 Username 和 Password 字段指定您的访问凭证，然后点击 **Refresh Headers**。

POSTMAN 会显示 Authorization 标头与加密密钥。

步骤 7　通过指定以下值添加 Accept 标头：application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

注　有关 Accept 标头的详细信息，请参阅 Accept 标头（第 7-98 页）。

步骤 8　点击 **Send**。

POSTMAN 插件会显示 200 OK 状态响应，表示请求成功。指定的 ISE 资源将从 ISE 服务器删除。

Cisco ISE 故障原因报告

本附录提供可用于访问 Cisco ISE 故障原因报告的过程。Cisco ISE 故障原因报告允许您查看故障原因列表。

简介

Cisco ISE 故障原因报告是在提供有关所有的信息故障原因可能遇到的 Cisco ISE 用户界面的选项。您可以使用此检查返回作为输出将故障原因映射呼叫，当使用故障排除 API 时的 Cisco ISE 查询的。

Cisco ISE 故障原因报告让您访问的故障原因的完整列表适用于监控 ESS 节点运行的 Cisco Cisco ISE 软件定义的。以下程序允许您查看或编辑定义的故障原因列表。您必须登录到该目标 ESS 监控节点的 Cisco ISE 用户界面查看和访问故障原因。有关登录的详细信息，请参阅验证监控节点，第 1-2 页。

查看故障原因

步骤 1 选择 **Operations > Reports > Authnetication Summary** 报告。

步骤 2 在 Navigation 窗格中，展开**监控**并选择**故障原因编辑器**。

步骤 3 从提供的过滤器列表选择故障原因。

步骤 4 提供您要查找的故障原因。

步骤 5 点击 Run。

故障原因列表在右侧面板中显示。

步骤 6 单击任何故障原因获得详细数据报表在新窗口中。