Lab Manager SOAP API 参考

vCenter Lab Manager 4.0

CN-000175-00

vmware®

您可以在 VMware 网站上找到最新的技术文档，网址为：

http://www.vmware.com/cn/support/

VMware 网站还提供最新的产品更新。

如果对本文档有任何意见或建议，请将反馈信息提交至以下地址：

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

目录

关于本文档

使用 VMware® 《vCenter Lab Manager SOAP API 指南》，可以开发使用 Lab Manager Web 服务数据、自动执行任务或将 Lab Manager 与其他软件测试工具集成的应用程序。

目标读者

本指南适用于要将 Lab Manager 数据用于自定义测试解决方案或将 Lab Manager 和其环境中的其他软件测试工具集成的开发人员。例如，您可以使用 Lab Manager SOAP API 将 Lab Manager 与自动软件测试工具集成。

要使用本指南中的信息，您应该熟悉以下几项：

- 虚拟机技术
- 分布式、多层系统概念
- 开发和测试实践
- Windows 或 Linux 操作系统
- Web 服务、 SOAP 和 XML

文档反馈

VMware 欢迎您提出宝贵建议，以便改进我们的文档。如有意见或建议，请将反馈发送到 docfeedback@vmware.com。

技术支持和教育资源

下列各节介绍提供的技术支持资源。请通过以下网站访问本文档和其他文档的最新版本：http://www.vmware.com/cn/support/pubs。

在线支持和电话支持

要通过在线支持提交技术支持请求、查看产品和合同信息以及注册您的产品，请访问 http://www.vmware.com/cn/support。

客户只要拥有相应的支持合同，就可以通过电话支持，尽快获得对优先级高的问题的答复。请访问 http://www.vmware.com/cn/support/phone_support。

支持服务项目

要了解 VMware 支持服务如何帮助您满足业务需求，请访问 http://www.vmware.com/cn/support/services。

VMware 专业服务

VMware 教育服务课程提供了大量实践操作环境、案例研究示例，以及用作作业参考工具的课程材料。这些课程可以通过现场指导、教室授课的方式学习，也可以通过在线直播的方式学习。关于现场试点项目及实施的最佳实践， VMware 咨询服务可提供多种服务，协助您评估、计划、构建和管理虚拟环境。要了解有关教育课程、认证计划和咨询服务的信息，请访问 http://www.vmware.com/cn/services。

VMware vCenter Lab Manager SOAP API 简介 1

Lab Manager SOAP 应用程序编程接口 (API) 可提供对 Lab Manager 系统的编程访问。使用安全 API，您可以连接到 Lab Manager 服务器以自动化或执行各种操作。

Lab Manager SOAP API 使用基于 XML 的技术（包括 SOAP）作为通信协议，使用网络服务描述语言 (WSDL) 作为界面描述语言。Lab Manager WSDL 文件详细介绍了服务（在 Web 服务中称为操作）提供的方法、服务使用的参数类型和 SOAP 端点。

请注意 配置、虚拟机或模板对象的名称不应该区分大小写。

本章包含下列主题：

将 Lab Manager 与自动测试工具集成

借助 Lab Manager SOAP API，您可以使用所选语言和平台与 Lab Manager 数据交互。本指南中的示例使用的是 C# 编程语言和 Microsoft.NET framework，此外，还支持其他编程语言和开发环境。如果您使用的不是 C# 语言，请参见有关开发环境的文档，了解有关开发 Web 服务应用程序的信息。

使用 SOAP API 除了可以扩展或自定义 Lab Manager 之外，还可以将 Lab Manager 与自动测试系统集成。您可以在 “将 Lab Manager 和 Quality Center 集成的高级 C# 示例”（第 15 页）中查看这种集成的示例。

有关 Lab Manager 解决方案、开发人员资源和社区资源的详细信息，请访问 http://www.vmware.com。

支持的操作

使用首选的启用 Web 的开发环境，您可以使用标准的 Web 服务协议构建 Web 服务客户端应用程序，以便以编程方式执行下列任务：

- 查询虚拟机信息和配置信息。
- 对虚拟机和配置执行操作。
- 捕获、签出、克隆、删除和部署配置。
- 创建您可以通过电子邮件发送给其他团队成员的 LiveLink 配置 URL。

有关支持的 Web 服务操作的详细信息，请参见 “Lab Manager API 方法引用”（第 23 页）。

Lab Manager 数据对象

Lab Manager SOAP API 使用对象与您组织中的数据交互。对象是 Lab Manager 数据的编程表示形式。对象属性表示这些数据实体中的字段。例如， Lab Manager 配置由配置对象表示，配置对象包含表示配置名称、配置数字标识符、部署状态、共享状态以及其他信息的字段。

本文档介绍了如何使用 Lab Manager 数据对象对 Lab Manager 数据执行查询、克隆、捕获和部署等操作。请参见 “Lab Manager API 数据类型”（第 19 页）。

符合标准的兼容性开发平台

Lab Manager SOAP API 符合 SOAP 1.1、WSDL 1.1 以及 WS-I Basic Profile Version 1.1（WS-I 基本配置文件 1.1 版）中确定的其他标准。 Lab Manager SOAP API 可以与符合 WS-I Basic Profile Version 1.1 标准的当前 SOAP 开发环境结合使用。本文档中的示例使用的是 Microsoft Visual Studio .NET 2005 开发环境和 C# 编程语言。

请注意 某些开发平台上存在实施差异，这可能阻止对 Lab Manager SOAP API 中的部分或所有功能的访问。

如果您使用 Visual Studio 进行 .NET 开发， VMware 建议您使用 Visual Studio 2005 或更高版本。

SOAP API 安全性

那些访问您组织中的 Lab Manager 数据的客户端应用程序，授予 Lab Manager Web 控制台上使用的相同安全保护的限制。 Lab Manager 使用 SSL 公开所有 SOAP API 方法。

当使用 Web 服务 URL 访问具有该 Web 服务 URL 的 SOAP API 时，可能会显示 SSL 证书警告。接受该证书以使用 API 或将该证书替换为有效的已签署证书。

用户身份验证

客户端应用程序必须为每个 Lab Manager Web 服务方法调用提供有效的用户凭据。所需的凭据为 Lab Manager 用户帐户、密码、组织和工作区名称。必须提供包含要对其执行操作的对象的组织和/或工作区的名称。 Lab Manager 服务器会对这些凭据进行验证。

请注意 您可以使用非管理员凭据。

Lab Manager SOAP API 入门 2

您可以使用 Lab Manager SOAP API 开发 XML Web 服务客户端。 XML Web 服务客户端指任何引用和使用 XML Web 服务的组件或应用程序。这不需要基于客户端的应用程序。在很多情况下， XML Web 服务客户端可以是其他 Web 应用程序，例如 Web 窗体，甚至是其他 XML Web 服务。

本章包含下列主题：

开发应用程序的要求

在开始开发应用程序之前，请了解下列要求：

- VMware 假定您熟悉基本编程概念，并且已经在计算机上设置了编程开发环境。
- 网络上必须安装、配置并运行 Lab Manager 实例。
- 必须了解 Lab Manager 服务器实例的地址，该地址以完全限定主机名或 IP 地址开头，例如 https://hostname.company.com/LabManager。
- 您必须在目标 Lab Manager 服务器上拥有一个帐户。
- 复制下一节中显示的代码列表，并将其粘贴到 Microsoft Visual Studio 2005 环境中。

获取并导入 WSDL 文件

与任何基于标准的 SOAP API 实施一样， Lab Manager API 定义在 Web 服务上以 XML 格式的 WSDL 文件提供。要获取此文件，请打开 Internet Explorer 5.5 或更高版本，并导航到 https://<hostname>/LabManager/SOAP/LabManager.asmx?WSDL 找到 Lab Manager 服务器。

WSDL 文件可定义所有 Lab Manager API 调用和对象。有关 WSDL 的详细信息，请访问 http://www.w3.org/TR/wsdl。

将 WSDL 文件导入开发环境

在获取 WSDL 文件之后，将其导入开发环境并生成用于构建客户端 Web 服务应用程序的必要对象。此过程取决于开发环境、编程语言和关联的工具。例如， Microsoft Visual Studio 开发环境会自动处理这些任务。有关其他开发平台的说明，请参见该平台的产品文档。

将 Microsoft Visual Studio 与 Lab Manager WSDL 文件结合使用

Microsoft Visual Studio 编程语言通过作为其对应的服务器端代理的对象访问 Lab Manager SOAP API。当托管代码访问 XML Web 服务时，代理类和 .NET Framework 将处理所有基础结构编码。

您必须首先从 WSDL 文件生成代理类对象，才可以将 Lab Manager SOAP API 与 Visual Studio 结合使用。Visual Studio 提供了 [Add a Web Reference] 向导，以连接到 Web 服务并生成必要的项目。您可以将 Web 引用添加到现有应用程序或在 Visual Studio 中创建新的应用程序。

请参见 Visual Studio 文档中的 “添加和移除 Web 引用”。

使用 Microsoft Visual Studio 2005 添加 Web 引用

1 在 Windows 中，选择 **[Start] > [Microsoft Visual Studio .NET 2005]**。

2 选择 **[New Project]** 以创建新的项目，或选择 **[Open]** 以打开现有项目。

3 在 **[URL]** 文本框中，键入 **`https://<hostname>/LabManager/SOAP/LabManager.asmx`** 以获取 Lab Manager Web 服务的服务描述。

4 单击 **[Go]**。

开始在 Lab Manager 服务器和开发环境客户端之间交换证书。将出现一个安全警示，显示从服务器发送的证书的详细信息。

请注意 当 Lab Manager 服务器使用默认的自签署证书时，将生成安全警示消息。您可以使用从 Verisign、 Thawte 和其他证书颁发机构购买的证书替换 Lab Manager 服务器上的证书。

---

5 单击 **[Yes]**。

6 （可选）如果 Visual Studio 环境中出现警示，请单击 **[Yes]**。

Microsoft Visual Studio 环境将连接到 Web 服务端点并显示 Lab Manager Web 服务 WSDL 文件中描述的操作。

7 选择 [Web reference name] 文本框中的文本并键入 **`LabManager`**，即用于此 Web 引用的命名空间。

`LabManager` 是一个词，中间没有空格。

8 单击 **[Add Reference]**。

9 （可选）如果出现证书警告消息，请单击 **[Yes]**。

10 单击 **[Yes]**。

Visual Studio 将检索服务描述并生成 LabManager 代理类作为应用程序与 Lab Manager Web 服务的接口。在过程结束时，该类将添加到项目的 Web References 文件夹中。（单击 **[Solution Explorer]** 可查看 Web References 文件夹中列出的 LabManagerSoap。）

完成此基本设置任务之后，即可以构建使用 Lab Manager SOAP API 的客户端应用程序。熟悉 API 的最快方法是查看 “简单和高级 C# 代码示例”（第 13 页）中列出的代码示例。

简单和高级 C# 代码示例

有关设置要求，请参见 “开发应用程序的要求”（第 11 页）。

您可以使用 “简单 C# 示例控制台应用程序”测试开发工作站和 Lab Manager Web 服务之间的基本 API 编程连接性。如果您使用的编程语言不是 C#，而且 Web 服务开发环境不是 Microsoft Visual Studio 2005，请参见相应的文档以了解详细信息。

请注意 VMware 假定您熟悉基本编程概念，并且已经在计算机上设置了编程开发环境。

此代码示例可执行多个简单任务。任何调用 Lab Manager Web 服务的应用程序都需要执行前两个任务。

- 绑定到 Lab Manager SOAP API。
- 设置用户名和密码以调用 SOAP。
- 将 ServicePointManager 证书策略设置为接受 SSL 证书。要进行连接，必须将证书策略设置为接受所有证书。
- 根据对象名称调用以获得配置对象。
- 显示控制台中的所有配置字段。

简单 C# 示例控制台应用程序

复制以下示例代码并将其粘贴到 Microsoft Visual Studio 2005 环境中。

```
using System;
using System.Net;

namespace LMConsoleApplication1
{
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                //
                //** Bind to the Lab Manager SOAP API
                //
                LabManagerSoap.LabManagerSOAPinterface binding =
                new LabManagerSoap.LabManagerSOAPinterface();
                //
                //** Enter the URL for your system here
                //
                binding.Url ="https://10.6.1.248/LabManager/SOAP/LabManager.asmx";
                binding.Timeout = 10 * 60 * 1000; // 10 minutes
                ServicePointManager.CertificatePolicy = new CertificateAccepter();

                //**
                //** Allocate AuthenticationHeader object to hold caller's
                //** user name and password
                //**
                binding.AuthenticationHeaderValue = new
                    LabManagerSoap.AuthenticationHeader();

                //
                //** Substitute a real user's user name, password, organization, and workspace
                                name here
                //
                binding.AuthenticationHeaderValue.username = "jaya";
                binding.AuthenticationHeaderValue.password = "Lab Manager";
```

```
                    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
                    binding.AuthenticationHeaderValue.workspacename = "My'Workspace";

                    //**
                    //** Call GetSingleConfigurationByName()
                    //** Get default configuration that comes with Lab Manager
                    //** installation and write all property values to console
                    //**
                    LabManagerSoap.Configuration defCfg=
                        binding.GetSingleConfigurationByName("Sample Configuration");
                    //
                    //** Print out configuration properties to the Console
                    //
                    Console.WriteLine("Name = " + defCfg.name);
                    Console.WriteLine("ID = " + defCfg.id.ToString());
                    Console.WriteLine("Description = "+ defCfg.description);
                    Console.WriteLine("isPublic = "+ defCfg.isPublic.ToString());
                    Console.WriteLine("isDeployed = "+ defCfg.isDeployed.ToString());
                    Console.WriteLine("fenceMode = "+ defCfg.fenceMode.ToString());
                    Console.WriteLine("type = " + defCfg.type.ToString());
                    Console.WriteLine("owner = " + defCfg.owner);
                    Console.WriteLine("dateCreated = " +
                        defCfg.dateCreated.ToString());
                    Console.ReadLine();
                }
                catch (Exception e)
                {
                    Console.WriteLine("Error:" + e.Message);
                    Console.ReadLine();
                }
            } //** end Main
        } //** end Class1

        /// <summary>
        /// This class is needed to automatically accept the SSL certificate
        /// the Lab Manager sends on each API call.
        /// </summary>

        public class CertificateAccepter :System.Net.ICertificatePolicy
        {
            public CertificateAccepter() {}

            public bool CheckValidationResult(
                System.Net.ServicePoint servicePoint,
                System.Security.Cryptography.X509Certificates.X509Certificate cert,
                System.Net.WebRequest webRequest, int iProblem)
            {
                return true;
            }
        }
    } //** end Namespace}
```

将 Lab Manager 和 Quality Center 集成的高级 C# 示例

本节中的 C# .NET 示例是一个范围更广、更实际的使用 Lab Manager SOAP API 的示例。此代码示例显示了 Lab Manager SOAP API 调用与 Mercury Interactive Corporation Quality Center 产品的集成。示例代码将执行下列任务：

- 调用 Lab Manager API (Lab Manager SOAP API) 以从库中签出配置并对其进行部署。
- 使用 Mercury Quality Center 在已部署的配置上运行一系列预定义的测试。
- 调用 Lab Manager SOAP API 以从工作区中捕获配置，并取消对该配置的部署。

在示例代码中使用下列方法完成上述任务：

- **CheckoutDeployConfiguration( ) –** 从库中获取配置并将其部署到 Lab Manager 工作区。
- **RunQCTestset() –** 运行一系列预定义的 Mercury Interactive Quality Center 测试。有关预定义测试的详细信息，请参见 Mercury Interactive Quality Center 文档。
- **CaptureUndeployConfiguration( ) –** 取消对该配置的部署并将其捕获到库中。

此外， GetLMAPI() 方法将创建与 Lab Manager API 的新绑定并设置身份验证参数。此方法以编程方式将 .NET 服务点管理器的证书策略配置为接受任何证书。 GetLMAPI() 将返回 Lab Manager 绑定的实例。

复制以下示例代码并将其粘贴到 Microsoft Visual Studio 2005 环境中。

```
using System;
using System.Configuration;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using TDAPIOLELib; //** From Mercury Quality Center

namespace MATRun
{
    /// <summary>
    /// Class1 comprises methods to check out a configuration from the Lab
    /// Manager Library and deploy it to the Workspace; execute several
    /// tests; and capture a configuration.
    /// </summary>
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]

        static void Main(string[] args)
        {
            NameValueCollection settings=ConfigurationSettings.AppSettings;
            string filename      = null;
            string buildlocation = null;
            string buildversion  = null;

            if ( args.Length > 0 )
            {
                buildlocation = args[0];
                buildversion  = args[1];
             }

            if ( buildlocation == null )
            {
                buildlocation =
                    @"\\fs.labmanger.com\public\build\outputdir\1423\artifacts";
                buildversion = "Lab Manager-2.0.4018";
            }
            filename =
                @"\\fs.labmanager.com\public\build\build-to-test.bat";
            StreamWriter f = new StreamWriter(filename);
```

```
    f.WriteLine(String.Format(@"xcopy {0}\setup.exe c:\ /Y",
        buildlocation));
    f.Close();
    Console.WriteLine(String.Format("Testing {0} at location {1}",
        buildversion, buildlocation));

    string config = CheckoutDeployConfiguration(buildversion);
    RunQCTestset();
    CaptureUndeployConfiguration(config);

} //** End Main() method

//
//** Initialize parameters
//
static string library_config = "ProofOfBuild-R2";
static string storage_server ="LM Server";
static string perform_capture ="Yes";
static string soap_server = "LM Server";

///<summary>
/// The RunQCTestset()method executes a series of predefined
/// tests using Mercury Interactive's Quality Center product.
///</summary>

static void RunQCTestset()
{
    string server   = "https://demo12.Lab Manager.com/qcbin";
    string domain   = "Lab Manager_SYSTEMS";
    string project  = "Snapshot_20";
    string username = "jaya";
    string password = "Lab Manager";
    string host     = "10.6.1.34";
    string chosenTestSet = "Install_Verify";

// ----
    TDConnection tdc = new TDConnection();
    tdc.InitConnection(server, domain, "");
    tdc.ConnectProjectEx(domain, project, username, password);
    if ( tdc.Connected)
    {
        TestSetFactory testSetFactory =(TestSetFactory)tdc.TestSetFactory;
        List testSetList;
        testSetList = testSetFactory.NewList("");
        foreach ( TestSet testSet in testSetList)
        {
            if ( testSet.Name.ToUpper() == chosenTestSet.ToUpper())
                {
                Console.WriteLine("Scheduling "+ testSet.Name);
                TSScheduler sched = (TSScheduler)
                testSet.StartExecution(host);
                sched.RunAllLocally = false;
                sched.Run(null);
                ExecutionStatus status = (ExecutionStatus)
                sched.ExecutionStatus;
                while ( status.Finished == false )
                {
                   System.Threading.Thread.Sleep(30);
                   status.RefreshExecStatusInfo(null, true);
                }

                     // results
                     TDAPIOLELib.TSTestFactory tsf;
                     tsf = (TSTestFactory) testSet.TSTestFactory;
                     TDAPIOLELib.List testlist;
                     testlist = tsf.NewList("");
                     foreach ( TSTest test in testlist)
                     {
```

```
                            TDAPIOLELib.Run r= (Run) test.LastRun;
                            if (r != null)
                            {
                              Console.WriteLine(test.Name + " " + r.Name + " " +
                                r.Status.ToString());
                            }
                          } //** end foreach
                          break;

                    } //** end if
                } //** end foreach
            } //** end if
        } //** end RunQCTestset

    ///<summary>
    ///The CheckoutDeployConfiguration() method obtains the configuration
    ///from the Lab Manager Library and deploys it to the Lab Manager
    ///Workspace.
    ///</summmary>

    static string CheckoutDeployConfiguration( string version)
    {
        //
        //** Check out a configuration and deploy it to the Workspace

        string srcconfig = "ProofOfBuild-R2"; //** Configuration name
        System.DateTime time = System.DateTime.Now;
        string configname = version+"-"+
            time.ToString().Replace(" ", "_").Replace("/","-");

        //
        //** Bind to Lab Manager SOAP Web service
        //
        LabManagerSoap.LabManagerSOAPinterface binding = GetLMAPI();

        //
        //** Get configuration information -- Configuration object
        //
        LabManagerSoap.Configuration config =
            binding.GetSingleConfigurationByName(srcconfig);
        Console.WriteLine("Checkout configuratioin "+ srcconfig);

        //
        //** Check configuration out of Configuration Library and
        //** name it(configname)
        //
        int newCheckoutID = binding.ConfigurationCheckout(config.id, configname);
        Console.WriteLine("Deploy configuratioin "+ srcconfig);

        //
        //** Deploy Configuration
        //** false = Do not run images from ESX host
        //** 1 = Fenced mode, traffic blocked in and out
        //
        binding.ConfigurationDeploy(newCheckoutID, false, 1);
        Console.WriteLine("Deploy is completed");
        return configname;
        }

    ///<summary>
    /// The CaptureUndeployConfiguration() method saves the configuration
    /// to the Lab Manager Library and undeploys it from the workspace.
    ///</summary>

    static void CaptureUndeployConfiguration(string configname)
    {
    //
    //** Bind to Lab Manager SOAP Web Service
```

```
            //
            LabManagerSoap.LabManagerSOAPinterface binding = GetLMAPI();
            LabManagerSoap.Configuration config =
                binding.GetSingleConfigurationByName(configname);
            if ( perform_capture.Equals("Yes") )
                {
                    Console.WriteLine("Capture configuration "+ configname);
                    int newConfigCaptureID = binding.ConfigurationCapture(config.id,
                        configname);
                }
            Console.WriteLine("Undeploy configuration "+ configname);
            binding.ConfigurationUndeploy(config.id);
            Console.WriteLine("Undeploy is completed");
            }

        /// <summary>
        ///The GetLMAPI() method creates a new binding to the Lab Manager API
        ///and sets up authentication and other basic parameters.This method
        ///returns a CertificateAcceptor object.
        /// </summary>

            static LabManagerSoap.LabManagerSOAPinterface GetLMAPI()
            {
                //
                //** Bind to SOAP interface
                //
                LabManagerSoap.LabManagerSOAPinterface binding = new
                LabManagerSoap.LabManagerSOAPinterface();
                //
                //**Allocate caller login object
                //
                binding.AuthenticationHeaderValue = new
                    LabManagerSoap.AuthenticationHeader();
                binding.Url = binding.Url.Replace("https://qa240.VMware.com",
                    "https:/demo44.VMware.com");
                binding.AuthenticationHeaderValue.username = "jaya";
                binding.AuthenticationHeaderValue.password = "vlm";
                binding.AuthenticationHeaderValue.organizationname = "MyOrg";
                binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
                binding.Timeout = 10 * 60 * 1000; // 10 minutes

                ServicePointManager.CertificatePolicy = new CertificateAccepter();
                return binding; //** return binding reference
                }
            }
            /// <summary>
            /// The CertificateAccepter class automatically accepts the SSL
            /// certificate sent by Lab Manager with each API call from a client
            /// application.
            /// </summary>

            public class CertificateAccepter :System.Net.ICertificatePolicy
            {
                public CertificateAccepter() {}

                public bool CheckValidationResult(
                    System.Net.ServicePoint servicePoint,
                    System.Security.Cryptography.X509Certificates.X509Certificate cert,
                    System.Net.WebRequest webRequest, int iProblem)
                {
                    return true;
                }
            }//end CertificateAccepter class declaration
        }//end namespace declaration
```

Lab Manager API 数据类型

3

本章提供了有关 Lab Manager 中 API 数据类型的详细信息。

本章包含下列主题：

原始 XML 数据类型

Lab Manager SOAP API 数据类型基于表 3-1 中显示的原始 XML 数据类型。这些原始数据类型是用于 Lab Manager API 调用的 Lab Manager 数据类型的构建基块。

表 3-1. Lab Manager SOAP API 中的原始 XML 数据类型

值	描述
xsd:Boolean	逻辑值，包括 true、 false、 0 和 1。
xsd:date	日期值。
xsd:dateTime	日期 / 时间值（时间戳）。
xsd:double	与标准 IEEE 754-1985 中定义的 IEEE 双精度 64 位浮点类型对应的数值。
xsd:int	从 -2147483648 到 2147483647 的数值。
xsd:string	任何字符数据。

Lab Manager 数据类型

当编写客户端应用程序时，请遵守为您的编程语言和开发环境定义的数据键入规则。您的开发工具使用具有这些 SOAP 数据类型的编程语言映射键入的数据。

Lab Manager 数据类型在 Lab Manager WSDL 文件中定义。有关详细信息，请参见 “Lab Manager SOAP API 数据类型”（第 19 页）。

表 3-2. Lab Manager SOAP API 数据类型

数据类型	描述
AuthenticationHeader	包含调用方的用户名、密码、组织和工作区名称。这种数据类型在 Lab Manager Web 服务方法中是每个 SOAP 标头的一部分。
Configuration	配置对象。
Machine	计算机对象。

AuthenticationHeader

这种数据类型使用所有 Lab Manager SOAP API 方法传递调用方的用户名、密码、组织和工作区名称。

支持的 API 调用

这种数据类型支持所有 API 调用。

字段

表 3-3. AuthenticationHeader 字段

字段	数据类型	描述
organizationname	字符串	Lab Manager 组织名称。
password	字符串	Lab Manager 帐户密码。
username	字符串	Lab Manager 帐户用户名。
workspacename	字符串	Lab Manager 工作区名称。

C# 示例代码

```
/**
** Visual Studio Console application in C#
** LMsoap = Web reference name for LM Web service
** Set up login code for all LM Web service method calls
**
**/
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue =new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "hedley";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";

    ServicePointManager.CertificatePolicy = new CertificateAccepter();
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

配置

Lab Manager 配置库或工作区中的每个配置都存在这种数据类型。配置是一个组，其中包含虚拟机及其操作系统、应用程序和作为一个单元受 Lab Manager 控制的数据。

一个整数 ID 字段唯一标识一项配置，但不保证配置名称是唯一的。

表 3-4. 配置字段

字段	数据类型	描述
dateCreated	日期时间	配置创建日期。
description	字符串	配置描述。
fenceMode	整数	1 = 未防护。 2 = 已防护。阻止传入和传出流量。 3 = 已防护。仅允许传出流量。 4 = 已防护。允许传入和传出流量。
id	整数	配置标识符。
isDeployed	布尔型	如果已部署，则为 True。如果未部署，则为 False。
isPublic	布尔型	如果其他人可以查看和访问，则为 True。否则，为 False。
name	字符串	配置名称。
owner	字符串	所有者用户名。
type	整数	配置类型： 1 = 工作区配置。 2 = 捕获的库配置。 3 = 金牌高手捕获的库配置。 4 = 已归档的库配置。
autoDeleteInMiliSeconds	双精度	经过多久删除配置。如果您键入 0，则永远不会删除配置。
bucketName	字符串	如果工作区中存在配置，则显示工作区名称；如果库中存在配置，则显示组织名称。
mustBeFenced	SOAPUtils.SOAPMustBeFenced	未指定 “已防护”还是 “未防护”。 True = 仅防护。 False = 仅不防护。
autoDeleteDateTime	日期时间	经过多久删除配置。

计算机

Lab Manager 的配置库或工作区中的每个虚拟机都存在这种数据类型。一个整数 ID 字段唯一标识一台计算机，但除非在配置范围内，否则无法保证计算机名称的唯一性。

表 3-5. 计算机字段

字段	数据类型	描述
configID	整数	虚拟机所属的配置的 ID。
DatastoreNameResidesOn	字符串	用于存储虚拟机的数据存储的名称。
description	字符串	计算机描述。
externalIP	字符串	当在防护范围内时的临时 IP 地址。
HostNameDeployedOn	字符串	在其上部署虚拟机的 ESX 主机的名称。如果未部署虚拟机，则该字段为空。
id	整数	计算机标识符。
internalIP	字符串	分配的永久 IP 地址。
isDeployed	布尔型	如果已部署，则为 True。

表 3-5. 计算机字段（续）

字段	数据类型	描述
macAddress	字符串	分配给主 NIC 的 MAC 地址。
memory	整数	内存大小，以 MB 为单位。
name	字符串	计算机名称。
OwnerFullName	字符串	虚拟机所有者的全名。
status	整数	1 = 关闭。 2 = 打开。 3 = 已挂起。 4 = 死机。 128 = 无效。

Lab Manager API 方法引用

4

表 4-1 列出了 vCenter Lab Manager SOAP API 方法。有关这些方法以及如何使用 C# .NET 代码示例调用这些方法的详细信息，请单击以下各个链接。

表 4-1. Lab Manager SOAP API 方法

方法	描述
“ConfigurationCapture”（第 24 页）	捕获工作区配置并将其保存到指定的 Lab Manager 数据存储。
“ConfigurationCheckout”（第 25 页）	从配置库签出一种配置并将其移动到工作区。
“ConfigurationClone”（第 26 页）	克隆工作区中的配置活动并将其保存到存储。
“ConfigurationDelete”（第 27 页）	从工作区中删除配置。
“ConfigurationDeploy”（第 28 页）	在工作区中部署配置。
“ConfigurationPerformAction”（第 29 页）	对配置执行操作。
“ConfigurationSetPublicPrivate”（第 30 页）	将配置状态设置为 “公共”或 “专用”。公共配置可供其他人使用。专用配置仅供所有者使用。
“ConfigurationUndeploy”（第 31 页）	在工作区中不对配置进行部署并放弃其状态。
“GetConfiguration”（第 32 页）	返回与配置标识符匹配的配置对象。
“GetConfigurationByName”（第 33 页）	返回与配置名称匹配的配置对象。不保证配置名称是唯一的。
“GetCurrentOrganizationName”（第 34 页）	返回当前组织的名称。
“GetCurrentWorkspaceName”（第 34 页）	返回当前工作区名称。
“GetMachine”（第 35 页）	返回与计算机标识符匹配的计算机对象。
“GetMachineByName”（第 36 页）	返回与计算机名称匹配的计算机对象。
“GetSingleConfigurationByName”（第 37 页）	返回与配置名称匹配的单个配置对象。
“ListConfigurations”（第 38 页）	返回工作区或配置库中的配置对象阵列。
“ListMachines”（第 39 页）	返回对应于配置的数字标识符的计算机对象阵列。
“LiveLink”（第 40 页）	创建一个配置的 URL，可通过电子邮件方式发送并单击它来重新创建该配置。
“MachinePerformAction”（第 41 页）	对计算机执行操作。
“SetCurrentOrganizationByName”（第 42 页）	将组织设置为可供每个连续登录的用户使用。
“SetCurrentWorkspaceByName”（第 43 页）	将工作区设置为可供每个连续登录的用户使用。

ConfigurationCapture

此方法可捕获工作区配置并保存该配置。

语法

```
int newConfigId = ConfigurationCapture(10,
    ″Config10Captureî);
```

参数

字段	数据类型	描述
configurationID	整数	配置标识符。
newLibraryName	字符串	捕获名称。

响应

字段	数据类型	描述
configurationID	整数	新捕获的配置标识符。

C# 示例代码

```
try
{
    //**
    //** LabManagerSoap is the name of the Web reference in Visual Studio
    //**
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config =
        binding.GetSingleConfigurationByName("Config26");

    //** Get configuration identifier and deployed status from object
    int configurationId = Config.id;
    bool deployed = Config.isDeployed;

    //** Capture configuration if it’s deployed
    if (deployed)
    {
        //** Save capture with date and time stamp
        string captureName=Config.name + DateTime.Now.ToString();
        string LMStorageServer = "LM Server";
        binding.ConfigurationCapture(configurationId, captureName);
    }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ConfigurationCheckout

此方法从库中签出一种配置并使用其他名称将其移动到工作区中。

语法

```
int result = ConfigurationCheckout(7, "Config7May10");
```

参数

字段	数据类型	描述
configurationID	整数	配置库中的配置的数字标识符。
workspaceName	字符串	已签出的配置名称。

响应

字段	数据类型	描述
configurationID	整数	工作区中的配置的数字标识符。

C# 示例代码

```
try
{
    //
    //** LMSoap is the name of the Web reference in Visual Studio.
    //
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //
    //** Create login
    //
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    binding.Url =
        "https://demo18.LabManager.com/LabManager/SOAP/LabManager.asmx";
    binding.Timeout = 10 * 60 * 1000; // 10 minutes
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //
    //** Get Configuration object
    //
    LabManagerSoap.Configuration Config =
        binding.GetSingleConfigurationByName("Win2K3Exchange");
    int configurationId = Config.id;

    //
    //** Timestamp library configuration name as new Workspace name
    //
    string checkoutName=Config.name + DateTime.Now.ToString();

    //
    //** Check out and move to Workspace
    //
    int newConfigID = binding.ConfigurationCheckout(Config.id,
        checkoutName);
    Console.WriteLine("New Config ID=" + newConfigID.ToString());
    Console.ReadLine();
```

```
}
    catch (Exception e)
{
    Console.WriteLine("Error="+e.Message);
    Console.ReadLine();
}
```

ConfigurationClone

此方法克隆工作区配置、将其保存在数据存储中并使其在工作区中显示为新名称。

语法

```
int result = ConfigurationClone(6, "Config6Clone");
```

参数

字段	数据类型	描述
configurationId	整数	配置库中的配置的数字标识符。
newWorkspaceName	字符串	新工作区配置名称。

响应

字段	数据类型	描述
configurationID	整数	新工作区配置的数字标识符。

C# 示例代码

```
try
{
    //**
    //** LabManagerSoap is the name of the Web reference in Visual Studio
    //**
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Clone Configuration
    int newConfigId = binding.ConfigurationClone(24, "ClonedConfig24");
    Console.WriteLine("New Config ID=" + newConfigId.ToString());
}
    catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ConfigurationDelete

此方法从工作区中删除配置。但您不能删除已部署的配置。

语法

```
ConfigurationDelete(6);
```

参数

字段	数据类型	描述
configurationID	整数	工作区配置的数字标识符。

响应

无响应。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config=binding.GetSingleConfigurationByName(
        "Config24");

    //** Get configuration identifier and deployed status from object
    int configurationId = Config.id;
    bool deployed = Config.isDeployed;

    //** Delete configuration if it isn't deployed
    if (!deployed)
        {
            binding.ConfigurationDelete(configurationId);
        }
    else
        {
            //**
            //** Must undeploy configuration before deleting it
            //**
            binding.ConfigurationUndeploy(configurationId);
            binding.ConfigurationDelete(configurationId);
        }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ConfigurationDeploy

此方法允许您部署驻留在工作区中的未经部署的配置。

语法

```
ConfigurationDeploy(6, false, 1);
```

参数

字段	数据类型	描述
configurationID	整数	工作区中的配置的数字标识符。
isCached	布尔型	始终将值设置为 false。
fenceMode	整数	1 = Nonfenced 2 = FenceBlockInAndOut 3 = FenceAllowOutOnly 4 = FenceAllowInAndOut

响应

无响应。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config =
        binding.GetSingleConfigurationByName("Config24");

    //** Get configuration identifier and deployed status from object
    int configurationId = Config.id;
    bool deployed = Config.isDeployed;

    //** Deploy configuration if it isn't already.
    if (!deployed)
    {
        //** Deploy in fenced mode and run from ESX hosts
        binding.ConfigurationDeploy(configurationId, false, 1);
    }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ConfigurationPerformAction

此方法按照操作标识符指示执行下列配置操作之一：

- 1 = 启动。打开配置。
- 2 = 关机。关闭配置。不保存任何内容。
- 3 = 挂起。冻结 CPU 和配置状态。
- 4 = 恢复。恢复已挂起的配置。
- 5 = 重置。重新引导配置。
- 6 = 快照。保存特定时刻的配置状态。
- 7 = 恢复。将配置返回到快照状态。
- 8 = 关机。在关机之前关闭配置。

语法

```
ConfigurationPerformAction(int configurationID, int action);
```

参数

字段	数据类型	描述
action	整数	要对配置执行的操作。
configurationID	整数	配置标识符。

响应

无响应。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    int configurationType = 1; //** Get workspace configuration

    //**
    //** Get array of all configurations
    //**
    LabManagerSoap.Configuration [] configurations =
        binding.ListConfigurations(configurationType);

    //**
    //** Loop through all configurations.
    //**
    for (int j=0; j < configurations.Length; j++)
        {
            binding.ConfigurationPerformAction(configurations[j].id,4/*Resume*/);
        }
}
```

```
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ConfigurationSetPublicPrivate

使用此调用将配置状态设置为“公共”或“专用”。如果配置状态为“公共”，则所有组织内的所有 Lab Manager 用户都可访问此配置（只读）。如果配置状态为 “专用”，则只有所有者和管理员可查看该配置。

语法

```
ConfigurationSetPublicPrivate(10, false);
```

参数

字段	数据类型	描述
configurationID	整数	配置标识符。
isPublic	布尔型	true = 公共， false = 专用。

响应

无响应。

C# 示例代码

```
try
{
    //
    //** LabManagerSoap is the name of the Web reference in Visual Studio
    //
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config =
        binding.GetSingleConfigurationByName("Config24");

    //** Get configuration identifier and shared status from object
    bool shared = Config.isPublic;

    //** Make configuration public if it isn't already.
    if (!shared)
    {
        binding.ConfigurationSetPublicPrivate(Config.id, true);
    }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ConfigurationUndeploy

在工作区中取消部署配置并放弃其状态。

语法

```
ConfigurationUndeploy(10);
```

参数

字段	数据类型	描述
configurationID	整数	配置数字标识符。

响应

无响应。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.Url =
        "https://demo18.LabManager.com/LabManager/SOAP/LabManager.asmx";
    binding.Timeout = 10 * 60 * 1000; // 10 minutes
    ServicePointManager.CertificatePolicy = new CertificateAccepter();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    //
    //* Get configurations in Workspace, not Library
    //
    int configurationType= 1;
    LabManagerSoap.Configuration[] configurations =
        binding.ListConfigurations(configurationType);
    //
    //** Undeploy all deployed configurations I own
    //
    for (int i=0; i < configurations.Length; i++)
        {
        if (configurations[i].owner.Equals("jaya"))
            {
            binding.ConfigurationUndeploy(configurations[i].id);
            }
        }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

GetConfiguration

此方法可返回与已通过验证的配置 ID 匹配的配置类型对象。

语法

```
Configuration config = GetConfiguration(10);
```

字段	数据类型	描述
configurationID	整数	配置标识符。

响应

字段	数据类型	描述
Configuration	配置	与已通过验证的配置 ID 匹配的配置对象。

C# 示例代码

```
try
{
    //** LabManagerSoap is the name of the Web reference in Visual Studio
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config =
        binding.GetConfiguration(26);

    //** Write to the console configuration properties
    Console.WriteLine("Config name = " + Config.name);
    Console.WriteLine("Config id = " + Config.id.ToString());
    Console.WriteLine("Config description = " + Config.description);
    Console.WriteLine("Config isPublic = " + Config.isPublic.ToString());
    Console.WriteLine("Config isDeployed = " + Config.isDeployed.ToString());
    Console.WriteLine("Config fenceMode = " + Config.fenceMode.ToString());
    Console.WriteLine("Config type = " + Config.type.ToString());
    Console.WriteLine("Config owner = " + Config.owner);
    Console.WriteLine("Config dateCreated = " +
        Config.dateCreated.ToString());
    Console.ReadLine();
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

GetConfigurationByName

该调用获取配置名称并返回与该名称相匹配的配置阵列。工作区配置名称是唯一的。可以存在多个使用给定名称的配置。如果不存在使用给定名称的配置，将返回一个空阵列。

语法

```
Configuration [] config = GetConfigurationByName("Config9");
```

参数

字段	数据类型	描述
name	字符串	配置名称。

响应

字段	数据类型	描述
configuration[]	配置	使用相同名称的配置对象的阵列。

C# 示例代码

```
try
{
    //
    //** LabManagerSoap is the name of the Web reference in Visual Studio
    //
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //
    //** Get Configuration objects
    //
    LabManagerSoap.Configuration [] Configs =
        binding.GetConfigurationByName("Config24Capture");

    //
    //** Write to the console all configurations and their properties.
    //
    for (int i=0; i < Configs.Length; i++)
        {
        Console.WriteLine("Config name = " + Configs[i].name);
        Console.WriteLine("id = " + Configs[i].id.ToString());
        Console.WriteLine("description = " + Configs[i].description);
        Console.WriteLine("isPublic = " +
        Configs[i].isPublic.ToString());
        Console.WriteLine("isDeployed = " +
        Configs[i].isDeployed.ToString());
        Console.WriteLine("fenceMode = " +
        Configs[i].fenceMode.ToString());
        Console.WriteLine("type = " + Configs[i].type.ToString());
        Console.WriteLine("owner = " + Configs[i].owner);
        Console.WriteLine("dateCreated = " +
        Configs[i].dateCreated.ToString());
        Console.WriteLine();
        Console.ReadLine();
```

```
        }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

GetCurrentOrganizationName

该调用返回当前组织的名称。

语法

```
string organizationName = GetCurrentOrganizationName();
```

响应

字段	数据类型	描述
organization name	字符串	返回当前组织名称。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    string organizationName = binding.GetCurrentOrganizationName();

    Console.WriteLine("Current organization I am logged in:" + organizationName);

    Console.ReadLine();
}
    catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();

}
```

GetCurrentWorkspaceName

该调用返回当前工作区名称。

语法

```
string workspaceName = GetCurrentWorkspaceName();
```

响应

字段	数据类型	描述
workspace name	字符串	返回当前工作区名称。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname;
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    string workspaceName = binding.GetCurrentWorkspaceName();

    Console.WriteLine("Current workspace I am logged in:" + workspaceName);

    Console.ReadLine();
}
    catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();

}
```

GetMachine

该调用获取计算机的数字标识符并返回与其对应的计算机对象。

语法

```
Machine mach = GetMachine(10);
```

参数

字段	数据类型	描述
machineID	整数	计算机标识符。

响应

字段	数据类型	描述
machine	计算机	与计算机标识符匹配的计算机对象。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    LabManagerSoap.Machine machine = binding.GetMachine(35);

    //** Write to the console all machines in configuration.
    Console.WriteLine("Machine = " + machine.name);
    Console.WriteLine("id = " + machine.id.ToString());
```

```
    Console.WriteLine("description = " + machine.description);
    Console.WriteLine("internalIP = " + machine.internalIP);
    Console.WriteLine("externalIP = " + machine.externalIP);
    Console.WriteLine("status = " + machine.status.ToString());
    Console.WriteLine("isDeployed = " + machine.isDeployed.ToString());
    Console.ReadLine();
}
    catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

GetMachineByName

该调用获取配置标识符和计算机名称并返回匹配的计算机对象。

语法

```
Machine mach = GetMachineByName(10, "Config9VM1");
```

参数

字段	数据类型	描述
configurationId	整数	配置标识符。
name	字符串	计算机名称。

响应

字段	数据类型	描述
machine	计算机	计算机对象。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    LabManagerSoap.Machine machine = binding.GetMachineByName(10,
        "Config9VM1");

    //** Write to the console all machines fields
    Console.WriteLine("Machine = " + machine.name);
    Console.WriteLine("id = " + machine.id.ToString());
    Console.WriteLine("description = " + machine.description);
    Console.WriteLine("internalIP = " + machine.internalIP);
    Console.WriteLine("externalIP = " + machine.externalIP);
    Console.WriteLine("status = " + machine.status.ToString());
    Console.WriteLine("isDeployed = " + machine.isDeployed.ToString());
    Console.ReadLine();
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

GetSingleConfigurationByName

该调用获取配置名称、在配置库和工作区中搜索该名称并返回与其对应的配置对象。

语法

```
Configuration config = GetSingleConfigurationByName("Config9");
```

参数

字段	数据类型	描述
name	字符串	配置名称。

响应

字段	数据类型	描述
Configuration	配置	配置对象。

C# 示例代码

```
try
{
    //** LabManagerSoap is the name of the Web reference in Visual Studio
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config =
        binding.GetSingleConfigurationByName("Config24Capture");

    //** Write to the console configuration properties.
    Console.WriteLine("Config name = " + Config.name);
    Console.WriteLine("Config id = " + Config.id.ToString());
    Console.WriteLine("Config description = " + Config.description);
    Console.WriteLine("Config isPublic = " + Config.isPublic.ToString());
    Console.WriteLine("Config isDeployed = " + Config.isDeployed.ToString());
    Console.WriteLine("Config fenceMode = " + Config.fenceMode.ToString());
    Console.WriteLine("Config type = " + Config.type.ToString());
    Console.WriteLine("Config owner = " + Config.owner);
    Console.WriteLine("Config dateCreated = " +
        Config.dateCreated.ToString());
    Console.ReadLine();
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ListConfigurations

此方法返回配置类型阵列。根据请求的配置类型，为配置库中的每个配置或工作区中的每个配置返回一个对象。

语法

```
Configuration [] config = ListConfigurations(1);
```

参数

字段	数据类型	描述
configurationType	整数	1= 工作区配置， 2= 库配置。

响应

字段	数据类型	描述
configurations[]	配置阵列	配置对象阵列。

C# 示例代码

```
try
{
    //**
    //** LabManagerSoap is the name of the Web reference in Visual Studio.
    //**
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configurations in Workspace.
    int configurationType = 1; //** 1=Workspace
    LabManagerSoap.Configuration [] WSconfigurations =
        binding.ListConfigurations(configurationType);

    //** Write to the console all configurations
    for (int i=0; i < WSconfigurations.Length; i++)
    {
        Console.WriteLine("Configuration name = " +
        WSconfigurations[i].name);
        Console.WriteLine("id = " + WSconfigurations[i].id.ToString());
        Console.WriteLine("description = " +WSconfigurations[i].description);
        Console.WriteLine("isPublic = " +
            WSconfigurations[i].isPublic.ToString());
        Console.WriteLine("isDeployed = "+
            WSconfigurations[i].isDeployed.ToString());
        Console.WriteLine("fenceMode = " +
            WSconfigurations[i].fenceMode.ToString());
        Console.WriteLine("type = " + WSconfigurations[i].type.ToString());
        Console.WriteLine("owner = " + WSconfigurations[i].owner);
        Console.WriteLine("dateCreated = " +
            WSconfigurations[i].dateCreated.ToString());
        Console.WriteLine();
    }
    Console.ReadLine();
```

```
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

ListMachines

此方法返回计算机类型阵列。此方法为配置中的每个虚拟机返回一个计算机对象。

语法

```
Machine [] machines = ListMachines(1);
```

参数

字段	数据类型	描述
configurationID	整数	配置数字标识符。

响应

字段	数据类型	描述
machine[]	计算机阵列	计算机对象阵列。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.Url =
        "https://demo18.LabManager.com/LabManager/SOAP/LabManager.asmx";
    binding.Timeout = 10 * 60 * 1000; // 10 minutes
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    int configurationType = 1;

    //** Get workspace configuration
    LabManagerSoap.Configuration [] configurations =
        binding.ListConfigurations(configurationType);
    for (int j=0; j < configurations.Length; j++)
        {
        Console.WriteLine("Configuration = " +
            configurations[j].name.ToString());
        LabManagerSoap.Machine [] machines =
            binding.ListMachines(configurations[j].id);
        //** Write to the console all machines in configuration
        for (int i=0; i < machines.Length; i++)
            {
            Console.WriteLine("Machine = " + machines[i].name);
            Console.WriteLine("id = " + machines[i].id.ToString());
            Console.WriteLine("description = " + machines[i].description);
            Console.WriteLine("internalIP = " + machines[i].internalIP);
            Console.WriteLine("externalIP = " + machines[i].externalIP);
            Console.WriteLine("status = " + machines[i].status.ToString());
            Console.WriteLine("isDeployed = " +
```

```
            machines[i].isDeployed.ToString());
            Console.WriteLine();
            }
        Console.ReadLine();
        }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

LiveLink

此方法允许您创建库配置的 LiveLink URL。

语法

```
string url = LiveLink("LiveLinkWin2K"));
```

参数

字段	数据类型	描述
configurationName	字符串	库配置名称。

响应

字段	数据类型	描述
URL	字符串	包含库中的配置 URL 的字符串。可以通过电子邮件发送该 URL 并单击它来重新创建该配置。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
    LabManagerSoap.LabManagerSOAPinterface();

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    binding.Url =
        "https://demo18.LabManager.com/LabManager/SOAP/LabManager.asmx";
    binding.Timeout = 10 * 60 * 1000; // 10 minutes
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    //** Get Configuration object
    LabManagerSoap.Configuration Config =
        binding.GetSingleConfigurationByName("Win2kBEA");

    //** If configuration is deployed, livelink it
    if (Config.isDeployed)
        {
        string captureName= "Win2kBEA" + DateTime.Now.ToString();
        string url = binding.LiveLink(Config.name);
        Console.WriteLine("LiveLink URL="+url);
        Console.ReadLine();
        }
```

```
}
    catch (Exception e)
{
    Console.WriteLine("Error="+e.Message);
    Console.ReadLine();
}
```

MachinePerformAction

此方法按照操作标识符指示执行下列计算机操作之一：

- 1 = 启动。打开计算机。
- 2 = 关机。关闭计算机。不保存任何内容。
- 3 = 挂起。冻结计算机 CPU 和状态。
- 4 = 恢复。恢复已挂起的计算机。
- 5 = 重置。重新引导计算机。
- 6 = 快照。保存特定时刻的计算机状态。
- 7 = 恢复。将计算机返回到快照状态。
- 8 = 关机。在关机之前关闭计算机。

语法

```
MachinePerformAction(1, 3);
```

参数

字段	数据类型	描述
action	整数	要对计算机执行的操作。
machineID	整数	计算机标识符。

响应

无响应。

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    binding.AuthenticationHeaderValue.workspacename = "MyWorkspace";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    int configurationType = 1; //** Get workspace configuration

    //**
    //** Get array of all configurations
    //**
    LabManagerSoap.Configuration [] configurations =
        binding.ListConfigurations(configurationType);

    //**
    //** Loop through all configurations.
```

```
        //**
        for (int j=0; j < configurations.Length; j++)
            {
            //**
            //** Get array of all machines in configurations
            //**
            LabManagerSoap.Machine [] machines =
                binding.ListMachines(configurations[j].id);
                //**
                //** Loop through all machines
                //**
                for (int i=0; i < machines.Length; i++)
                {
                    //**
                    //** Check statusóif machine is suspended, then resume it
                    //**
                    if (machines[i].status == 3)
                    {
                        binding.MachinePerformAction(machines[i].id, 4);
                    }
                }
            }
    }
    catch (Exception e)
    {
        Console.WriteLine("Error:" + e.Message);
        Console.ReadLine();
    }
```

SetCurrentOrganizationByName

将该组织设置为可供每个连续登录的用户使用。此方法在授权标头中的组织名称为空时起作用。

语法

```
SetCurrentOrganizationByName("MyOrganization");
```

参数

字段	数据类型	描述
orgName	字符串	您所属的组织

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    // This field must be empty for SetCurrentOrganization to work
    binding.AuthenticationHeaderValue.organizationname = "";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    SetCurrentOrganizationByName("MyOrganization");
    string organizationName = GetCurrentOrganizationName();

    Console.WriteLine("Current organization I am logged in:" + organizationName);

    Console.ReadLine();
}
    catch (Exception e)
```

```
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

SetCurrentWorkspaceByName

将该工作区设置为可供每个连续登录的用户使用。此方法在授权标头中的工作区名称为空时起作用。

语法

```
SetCurrentWorkspaceByName("MyOrganization", "MyWorkspace");
```

参数

字段	数据类型	描述
orgName	字符串	您所属的组织
workspaceName	字符串	您所属的工作区

C# 示例代码

```
try
{
    LabManagerSoap.LabManagerSOAPinterface binding = new
        LabManagerSoap.LabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    // The following authorization header fields must be empty or absent
    // for the SetCurrentOrganization and SetCurrentWorkspace to work
    binding.AuthenticationHeaderValue.organizationname = "";
    binding.AuthenticationHeaderValue.workspacename = "";
    ServicePointManager.CertificatePolicy = new CertificateAccepter();

    SetCurrentOrganizationByName("MyOrganization");
    SetCurrentWorkspaceByName("MyOrganization", "MyWorkspace");

    string organizationName = GetCurrentOrganizationName();
    string workspaceName = GetCurrentWorkspaceName();

    Console.WriteLine("Current organization I am logged in:" + organizationName);
    Console.WriteLine("Current workspace I am logged in:" + workspaceName);

    Console.ReadLine();
}
    catch (Exception e)
{
    Console.WriteLine("Error:" + e.Message);
    Console.ReadLine();
}
```

索引