SOAP API Guide

Lab Manager 3.0

Lab Manager SOAP API Guide
Revision: 20080804
Item: EN-000067-00

You can find the most up-to-date technical documentation on our Web site at:

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

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

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

Contents

About This Book 7

1 Introducing VMware Lab Manager SOAP API 9
- Integrating Lab Manager with Automated Testing Tools 10
- Supported Operations 10
- Lab Manager Data Objects 10
- Standards Compliance and Compatible Development Platforms 11
- Security 11
- User Authentication 11

2 Getting Started with the Lab Manager SOAP API 13
- Requirements 13
- Obtaining and Importing the WSDL 14
  - Importing the WSDL File into Your Development Platform 14
  - Instructions for Using Microsoft Visual Studio with Lab Manager WSDL 14
- Simple and Advanced Code Samples 16
  - Simple C# Console Application 17
  - Advanced C# Sample: Integrating Lab Manager and Quality Center 19

3 Lab Manager API Data Types 27
- Primitive XML Data Types 27
- Lab Manager Data Types 28
- AuthenticationHeader 28
  - Supported API Calls 28
  - Fields 28
  - Sample Usage: C# 29
- Configuration 29
- Machine 30

4 Lab Manager API Method Reference 33
ConfigurationCapture 34
Syntax 34
Arguments 34
Response 35
Sample Code: C# 35
ConfigurationCheckout 36
Syntax 36
Arguments 36
Response 36
Sample Code: C# 36
ConfigurationClone 37
Syntax 37
Arguments 37
Response 38
Sample Code: C# 38
ConfigurationDelete 38
Syntax 38
Arguments 39
Response 39
Sample Code: C# 39
ConfigurationDeploy 40
Syntax 40
Arguments 40
Response 40
Sample Code: C# 40
ConfigurationPerformAction 41
Syntax 42
Arguments 42
Response 42
Sample Code: C# 42
ConfigurationSetPublicPrivate 43
Syntax 43
Arguments 43
Response 43
Sample Code: C# 43
ConfigurationUndeploy 44
Syntax 44
Arguments 44
Response 44
Sample Code: C# 44

GetConfiguration 45
Syntax 45
Response 45
Sample Code: C# 46
GetConfigurationByName 47
Syntax 47
Arguments 47
Response 47
Sample Code: C# 47
GetMachine 48
Syntax 48
Arguments 48
Response 49
Sample Code: C# 49
GetMachineByName 50
Syntax 50
Arguments 50
Response 50
Sample Code: C# 50
GetSingleConfigurationByName 51
Syntax 51
Arguments 51
Response 51
Sample Code: C# 51
ListConfigurations 52
Syntax 52
Arguments 52
Response 52
Sample Code: C# 53
ListMachines 54
Syntax 54
Arguments 54
Response 54
Sample Code: C# 54
LiveLink 55
Syntax 55
Arguments 55
Response 55
Sample Code: C# 56

MachinePerformAction 56
Syntax 57
Arguments 57
Response 57
Sample Code: C# 57

Index 59

About This Book

Use the *Lab Manager SOAP API Guide* to develop applications that use Lab Manager Web service data, automate tasks, or integrate VMware® Lab Manager with other software testing tools.

Intended Audience

This guide is intended for developers who want to use Lab Manager data for customized testing solutions, or integrate Lab Manager and other software testing tools in their environment. For example, using the Lab Manager SOAP API lets you integrate Lab Manager with automated software testing tools.

This guide assumes you have some familiarity with:

- Virtual machine technology
- Distributed, multitiered systems concepts
- Development and testing practices
- Windows or Linux operating systems
- Web Services, SOAP, and XML

Document Feedback

VMware welcomes your suggestions for improving our documentation. If you have comments, send your feedback to:

docfeedback@vmware.com

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current versions of this book and other books, go to:

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

Online and Telephone Support

Use online support to submit technical support requests, view your product and contract information, and register your products. Go to:

http://www.vmware.com/support

Customers with appropriate support contracts should use telephone support for the fastest response on priority 1 issues. Go to:

http://www.vmware.com/support/phone_support.html

Support Offerings

Find out how VMware support offerings can help meet your business needs. Go to:

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

VMware Professional Services

VMware Education Services courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. Courses are available onsite, in the classroom, and live online. For onsite pilot programs and implementation best practices, VMware Consulting Services provides offerings to help you assess, plan, build, and manage your virtual environment. To access information about education classes, certification programs, and consulting services, go to:

http://www.vmware.com/services/

Introducing VMware Lab Manager SOAP API

1

The Lab Manager SOAP application programming interface (API) provides programmatic access to the Lab Manager system. By using the secure API, you can connect to Lab Manager Server to automate or perform various operations.

The Lab Manager SOAP API uses XML-based technologies, including SOAP, as the communication protocol and Web Services Description Language (WSDL) as the interface description language. The Lab Manager WSDL file details the available methods of the service (called "operations" in Web Services vernacular) and parameter types, as well as the SOAP endpoint for the service.

This chapter includes the following topics:

- "Integrating Lab Manager with Automated Testing Tools" on page 10
- "Supported Operations" on page 10
- "Lab Manager Data Objects" on page 10
- "Standards Compliance and Compatible Development Platforms" on page 11
- "Security" on page 11
- "User Authentication" on page 11

Integrating Lab Manager with Automated Testing Tools

The Lab Manager SOAP API allows you to interact with Lab Manager data using the language and platform of your choice. The examples in this guide use the C# programming language and the Microsoft .NET framework, but other programming languages and development environments are also supported. If you are using a language other than C#, see the documentation on your development environment for comparable information about developing Web service applications.

In addition to extending or customizing Lab Manager by using the SOAP API, you can also integrate Lab Manager with automated testing systems. You can see an example of this integration in "Advanced C# Sample: Integrating Lab Manager and Quality Center" on page 19.

For more information about Lab Manager solutions, developer resources, and community resources, go to http://www.vmware.com.

Supported Operations

Using your preferred Web-enabled development environment, you can construct Web service client applications using standard Web service protocols to programmatically:

- Query for virtual machine and configuration information.
- Perform actions on machines and configurations.
- Capture, checkout, clone, delete, and deploy configurations.
- Create a LiveLink configuration URL that you can email to other team members.

For detailed information about supported Web service operations, see "Lab Manager API Method Reference" on page 33.

Lab Manager Data Objects

The Lab Manager SOAP API interacts with the data in your organization using objects, which are programmatic representations of the Lab Manager data. Object properties represent fields in those data entities. For example, a Lab Manager configuration is represented by a Configuration object, which has fields that represent the configuration name, configuration numeric identifier, deployment status, shared state, and more.

This document describes how to perform operations such as query, clone, capture, and deploy on Lab Manager data using the Lab Manager data objects. See "Lab Manager API Data Types" on page 27.

Standards Compliance and Compatible Development Platforms

The Lab Manager SOAP API complies with SOAP 1.1, WSDL 1.1, and other standards identified in the WS-I Basic Profile Version 1.1. The Lab Manager SOAP API works with current SOAP development environments that adhere to the Basic Profile Version 1.1 standards. The examples in this document use the Microsoft Visual Studio .NET 2003 development environment and the C# programming language.

NOTE Implementation differences in certain development platforms might prevent access to some or all of the features in the Lab Manager SOAP API.

If you are using Visual Studio for .NET development, VMware recommends that you use Visual Studio 2003 or higher.

Security

Client applications that access the Lab Manager data in your organization are subject to the same security protections that are used in the Lab Manager Web console. Lab Manager exposes all SOAP API methods using SSL.

When accessing the SOAP API with the Web service URL, you might see an SSL certificate warning. Accept the certificate to use the API or replace the certificate with a valid signed certificate.

User Authentication

Client applications must provide valid credentials—a Lab Manager user account, password, and organization name—with each Lab Manager Web service method call. The Lab Manager Server authenticates these credentials.

Getting Started with the Lab Manager SOAP API

2

You can review introductory information about using the Lab Manager SOAP API to develop an XML Web service client. An XML Web service client is any component or application that references and uses an XML Web service. This does not require a client-based application. In many cases, XML Web service clients might be other Web applications, such as Web Forms or even other XML Web services.

This chapter covers these topics:

- "Requirements" on page 13
- "Obtaining and Importing the WSDL" on page 14
- "Simple and Advanced Code Samples" on page 16

Requirements

The instructions in this chapter assume that an instance of Lab Manager is installed, configured, and running on your network. Before you can start developing an application, review these requirements:

- You must know the address of the Lab Manager server instance, starting with its fully qualified host name or IP address. For example:

  ```
  https://hostname.company.com/LabManager
  ```

- You must have an account on the target Lab Manager server.

Assuming you have an appropriate account on the Lab Manager Server, you can continue with "Obtaining and Importing the WSDL" on page 14.

Obtaining and Importing the WSDL

As with any standards-based SOAP API implementation, the Lab Manager API definition is available on the Web service as an XML-formatted WSDL file.

To obtain the WSDL, launch Internet Explorer 5.5 or higher and navigate to this URL for your Lab Manager Server:

```
https://<hostname>/LabManager/SOAP/LabManager.asmx?WSDL
```

The WSDL defines all the Lab Manager API calls and objects.

For more information on WSDL, go to http://www.w3.org/TR/wsdl.

Importing the WSDL File into Your Development Platform

After you obtain the WSDL, import it into your development environment and generate the necessary objects for use in building client Web service applications. The process depends on your development environment, programming language, and associated tools. For example, the Microsoft Visual Studio development environment handles the tasks automatically.

The next section provides an example of obtaining the WSDL and importing it in the Microsoft Visual Studio 2003. For instructions about other development platforms, see the product documentation for your platform.

Instructions for Using Microsoft Visual Studio with Lab Manager WSDL

Microsoft Visual Studio programming languages access the Lab Manager SOAP API through objects that serve as proxies for their server-side counterparts.

When accessing XML Web services in managed code, a proxy class and the .NET Framework handle all of the infrastructure coding.

Before you can use the Lab Manager SOAP API with Visual Studio, you must first generate the proxy class object from the WSDL file.

Visual Studio provides a wizard (“Add a Web Reference”) to connect to a Web service and generate the necessary artifacts. You can add a Web reference to an existing application or create a new application in Visual Studio.

The instructions below are specific to Microsoft Visual Studio 2003.

To add a Web reference

1 From the Windows Start menu, launch Microsoft Visual Studio .NET 2003.

The Visual Studio environment opens.

2 Select **New Project** to create a new project, or select **Open** to open an existing project.

3 In Visual Studio, choose **Add Web Reference** from the **Project** menu.

4 In the **URL** text box, type the URL to obtain the service description of the Lab Manager Web service:

```
https://<hostname>/LabManager/SOAP/LabManager.asmx
```

5 Click **Go**.

The certificate exchange between the Lab Manager server and the development environment client begins. A security alert displays the details of the certificate sent from the server.

NOTE The security alert messages are generated when the Lab Manager server uses the default, self-signed certificates. You can replace these certificates on the Lab Manager server with certificates purchased from Verisign, Thawte, and other certificate authorities.

---

6 Click **Yes**.

An alert from the Visual Studio environment might appear. Click **Yes**. The Microsoft Visual Studio environment connects to the Web service endpoint and displays the operations described in the Lab Manager Web service WSDL.

7 Select the text in the Web reference name text box and rename the Web reference to LabManagerSoap, the namespace used for this Web reference.

"LabManagerSoap" is one word, without spaces.

8 Click **Add Reference**.

A certificate warning message might appear. Click **Yes**.

9 Click **Yes** again.

See the "Adding and Removing Web References" topic in the Visual Studio documentation.

Visual Studio retrieves the service description and generates a proxy class (LabManagerSoap) that serves as an interface to the Lab Manager Web service from your application. At the end of the process, the class is added to the Web References folder of the project. (Click **Solution Explorer** to see LabManagerSoap listed in the Web References folder.)

With this basic setup task completed, you can build client applications that use the Lab Manager SOAP API. The fastest way to become familiar with the API is by reviewing the code sample listed in "Simple and Advanced Code Samples" on page 16.

Simple and Advanced Code Samples

This section contains two code samples—one simple and one more complex—written in C# using the Microsoft Visual Studio 2003 IDE.

Assuming that you have a Lab Manager instance running, your programming environment is set up, and you have the appropriate permissions on the Lab Manager server (see "Requirements" on page 13), you can test basic API programming connectivity between your development workstation and your Lab Manager Web service by using the "Simple C# Console Application." Copy the code listing displayed in the next section, and paste it into your Microsoft Visual Studio 2003 environment.

NOTE VMware assumes you are familiar with basic programming concepts and already have a programming development environment set up on your computer. If you are using a programming language other than C# and a Web services development environment other than Microsoft Visual Studio 2003, see the appropriate documentation for more information.

The code performs several simple tasks. The first two tasks (binding to the Web service and providing credentials) are typically required of any application that makes calls to a Lab Manager Web service:

- Binds to the Lab Manager SOAP API.
- Sets up the user name and password for making a SOAP call.
- Sets up the ServicePointManager certificate policy to accept the SSL certificate. You must set up the certificate policy to accept all certificates to connect to the API.
- Makes a call to get a configuration object based on name.
- Displays all configuration fields in the console.

Simple C# Console Application

```
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.VMwareLabManagerSOAPinterface binding =
                new LabManagerSoap.VMwareLabManagerSOAPinterface();
                //
                //** Enter the URL for your system here
                //
                binding.Url
                              ="https://10.6.1.248/LabManager/SOAP/LabManag
                              er.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, and
                              organization name here
                //
                binding.AuthenticationHeaderValue.username = "jaya";
                binding.AuthenticationHeaderValue.password = "Lab Manager";
                binding.AuthenticationHeaderValue.organizationname =
                              "MyOrg";
```

```
                //**
                //** 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 all 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}
```

Advanced C# Sample: Integrating Lab Manager and Quality Center

The C# .NET example in this section is a more extensive—and more practical—example of using the Lab Manager SOAP API. This sample shows the integration of the Lab Manager SOAP API calls with Mercury Interactive Corporation Quality Center product. The sample code performs these tasks:

- Makes Lab Manager API (Lab Manager SOAP API) calls to check out a configuration from the Library and deploy it.
- Runs a series of predefined tests on the deployed configuration using Mercury Quality Center.
- Makes Lab Manager SOAP API calls to capture the configuration and undeploy it from the Workspace.

These tasks are accomplished in the sample code using these three methods:

- **CheckoutDeployConfiguration()** method – Obtains the configuration from the Library and deploys it to the Lab Manager Workspace.
- **RunQCTestset()** method – Runs a series of predefined Mercury Interactive Quality Center tests. (For more information about the predefined tests, see the Mercury Interactive Quality Center documentation.)
- **CaptureUndeployConfiguration()** method – Undeploys the configuration and captures it to the Library.

In addition, the GetLMAPI() method creates a new binding to the Lab Manager API and sets up authentication parameters. This method configures the certificate policy for the .NET service point manager to accept any certificate programmatically. GetLMAPI() returns an instance of the Lab Manager binding.

```
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\artifac
                                ts";
                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.VMwareLabManagerSOAPinterface 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.VMwareLabManagerSOAPinterface 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.VMwareLabManagerSOAPinterface GetLMAPI()
    {
        //
        //** Bind to SOAP interface
        //
        LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
        //
        //**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.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 Data Types

3

This chapter covers these topics about Lab Manager API data types:

- "Primitive XML Data Types" on page 27
- "Lab Manager Data Types" on page 28
- "AuthenticationHeader" on page 28
- "Configuration" on page 29
- "Machine" on page 30

Primitive XML Data Types

Lab Manager SOAP API data types are based on the primitive XML data types shown in Table 3-1. These primitive types are the building blocks for the Lab Manager data types used in making Lab Manager API calls.

Table 3-1. Primitive XML Data Types in the Lab Manager SOAP API

Value	Description
xsd:Boolean	A logical value, including true, false, 0, and 1.
xsd:date	Date values.
xsd:dateTime	Date/time values (timestamps).
xsd:double	Numeric value that corresponds to the IEEE double-precision 64-bit floating point type defined in the standard IEEE 754-1985.
xsd:int	Numeric value from -2147483648 to 2147483647.
xsd:string	Any character data.

Lab Manager Data Types

When writing your client application, follow the data typing rules defined for your programming language and development environment.Your development tool handles the mapping of typed data in your programming language with these SOAP data types.

The Lab Manager data types are defined in the Lab Manager WSDL file. For each type, this chapter lists its properties and description.

Table 3-2. Lab Manager SOAP API Data Types

Data Type	Description
AuthenticationHeader	Contains the user name, password, and organization name of the caller. This data type is part of every SOAP header in Lab Manager Web Service methods.
Configuration	Configuration object.
Machine	Machine object.

AuthenticationHeader

This data structure passes the user name, password, and organization name of the caller in all Lab Manager SOAP API methods.

Supported API Calls

All

Fields

Table 3-3. AuthenticationHeader Fields

Field	Data Type	Description
organizationname	string	Lab Manager organization name.
password	string	Lab Manager account password.
username	string	Lab Manager account user name.

Sample Usage: C#

```
/**
** 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.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    binding.AuthenticationHeaderValue =new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "hedley";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";

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

Configuration

This data structure exists for each configuration in the Lab Manager configuration library or Workspace. A configuration is a group of virtual machines (and its operating systems, applications, and data) which Lab Manager controls as a single unit.

An integer ID field uniquely identifies a configuration. Configuration names are not guaranteed to be unique.

Table 3-4. Configuration Fields

Field	Data Type	Description
dateCreated	dateTime	Configuration creation date.
description	string	Configuration description.
fenceMode	int	1 = Not fenced. 2 = Fenced – Block traffic in and out. 3 = Fenced – Allow traffic out only. 4 = Fenced – Allow traffic in and out.
id	int	Configuration identifier.

Table 3-4. Configuration Fields (Continued)

Field	Data Type	Description
isDeployed	boolean	True if deployed; false if not deployed.
isPublic	boolean	True if others can view and access; false if not.
name	string	Configuration name.
owner	string	Owner user name.
type	int	Configuration type: 1 =Workspace configurations, 2 = Library configurations.

Machine

This data structure exists for each virtual machine in the configuration library or Workspace of Lab Manager. An integer ID field uniquely identifies a machine. Machine names are not guaranteed to be unique except within a configuration.

Table 3-5. Machine Fields

Field	Data Type	Description
configID	int	ID of the configuration to which the virtual machine belongs.
datastoreNameResidesOn	string	Name of the datastore on which the virtual machine is stored.
description	string	Machine description.
externalIP	string	Temporary IP address when inside the fence.
hostNameDeployedOn	string	Name of the ESX host on which the virtual machine is deployed. The field is empty is the virtual machine is not deployed.
id	int	Machine identifier.
internalIP	string	Permanent assigned IP address.
isDeployed	boolean	True if deployed
macAddress	string	MAC address assigned to the primary NIC.
memory	int	Memory size in MB.
name	string	Machine name.

Table 3-5. Machine Fields (Continued)

Field	Data Type	Description
ownerFullName	string	Full name of the virtual machine owner.
status	int	1=Off 2 =On 3=Suspended 4=Stuck 128=Invalid.

Lab Manager API Method Reference

4

This section contains information about Lab Manager Web service methods and how to call them using C# .NET code samples.

Table 4-1. Lab Manager SOAP API Methods

Method	Description
"ConfigurationCapture" on page 34	Captures a Workspace configuration and saves it to a specified Lab Manager datastore.
"ConfigurationCheckout" on page 36	Checks out a configuration from the configuration library and moves it to the Workspace.
"ConfigurationClone" on page 37	Clones a configuration active in the Workspace and saves it to storage.
"ConfigurationDelete" on page 38	Deletes a configuration from the Workspace.
"ConfigurationDeploy" on page 40	Deploys a configuration in the Workspace.
"ConfigurationPerformAction" on page 41	Performs an action on a configuration.
"ConfigurationSetPublicPrivate" on page 43	Sets the configuration state to public or private. Public configurations are accessible for others to use. Private configurations are only available for the owner.
"ConfigurationUndeploy" on page 44	Undeploys a configuration in the Workspace and discards its state.
"GetConfiguration" on page 45	Returns a Configuration object matching a configuration identifier.

Table 4-1. Lab Manager SOAP API Methods (Continued)

Method	Description
"GetConfigurationByName" on page 47	Returns Configuration objects matching a configuration name. (Configuration names are not guaranteed to be unique.)
"GetMachine" on page 48	Returns a Machine object matching a machine identifier.
"GetMachineByName" on page 50	Returns a Machine object matching a machine name.
"GetSingleConfigurationByName" on page 51	Returns a single Configuration object matching a configuration name.
"ListConfigurations" on page 52	Returns an array of Configuration objects in the Workspace or configuration library.
"ListMachines" on page 54	Returns an array of Configuration objects corresponding to the numeric identifier of a configuration.
"LiveLink" on page 55	Creates a URL to a configuration that can be emailed and clicked on to recreate the configuration.
"MachinePerformAction" on page 56	Performs an action on a machine.

ConfigurationCapture

This method captures a Workspace configuration and saves it to a specified Lab Manager datastore with a name.

Syntax

```
int newConfigId = ConfigurationCapture(10,
    "Config10Capture");
```

Arguments

Field	Data Type	Description
configurationID	int	Configuration identifier.
newLibraryName	string	Capture name.

Response

Field	Data Type	Description
configurationID	int	Configuration identifier of the new capture.

Sample Code: C#

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

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

This method checks out a configuration from the configuration library and moves it to the Workspace under a different name.

Syntax

```
int result = ConfigurationCheckout(7, “Config7May10”);
```

Arguments

Field	Data Type	Description
configurationID	int	Numeric identifier of the configuration in the configuration library.
workspaceName	string	Workspace name of the checked-out configuration.

Response

Field	Data Type	Description
workspaceID	int	Numeric identifier of the configuration in the Workspace.

Sample Code: C#

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

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

This method clones a Workspace configuration, saves it in a datastore, and makes it visible in the Workspace under the new name.

Syntax

```
int result = ConfigurationClone(6, “Config6Clone”);
```

Arguments

Field	Data Type	Description
configurationId	int	Numeric identifier of the configuration in the configuration library.
newWorkspaceName	string	Workspace name of the clone.

Response

Field	Data Type	Description
workspaceId	int	Numeric identifier of the new Workspace configuration.

Sample Code: C#

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

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

This method deletes a configuration from the Workspace. You cannot delete a deployed configuration.

Syntax

```
ConfigurationDelete(6);
```

Arguments

Field	Data Type	Description
configurationID	int	Numeric identifier of the Workspace configuration.

Response

No response.

Sample Code: C#

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

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

This method allows you to deploy an undeployed configuration which resides in the Workspace.

Syntax

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

Arguments

Field	Data Type	Description
configurationID	int	Numeric identifier of the configuration in the Workspace.
isCached	boolean	Always set a false value.
isFenced	int	1 = Not fenced. 2 = Fenced – Block traffic in and out. 3 = Fenced – Allow traffic out only. 4 = Fenced – Allow traffic in and out. When you fence a configuration, only the primary NIC is fenced.

Response

No response.

Sample Code: C#

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

    //** Create login
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
```

```
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    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, true, 1);
    }
}
catch (Exception e)
{
    Console.WriteLine("Error: " + e.Message);
    Console.ReadLine();
}
```

ConfigurationPerformAction

This method performs one of the following configuration actions as indicated by the action identifier:

- 1 – Power on. Turns on a configuration.
- 2 – Power off. Turns off a configuration. Nothing is saved.
- 3 – Suspend. Freezes the CPU and state of a configuration.
- 4 – Resume. Resumes a suspended configuration.
- 5 – Reset. Reboots a configuration.
- 6 – Snapshot. Saves a configuration state at a specific point in time.
- 7 – Revert. Returns the configuration to a snapshot state.
- 8 – Shutdown. Shuts down a configuration before turning it off.

Syntax

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

Arguments

Field	Data Type	Description
action	int	Action to take on the configuration.
configurationID	int	Configuration identifier.

Response

No response.

Sample Code: C#

```
try
{
    LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    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/*Resum
                        e*/);
        }
}
```

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

ConfigurationSetPublicPrivate

Use this call to set the state of a configuration to public or private. If the configuration state is public, all Lab Manager users in all organizations can access this configuration (read only). If the configuration is private, only its owner and administrators can view it.

Syntax

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

Arguments

Field	Data Type	Description
configurationID	int	Configuration identifier.
isPublic	boolean	true = public, false = private.

Response

No response.

Sample Code: C#

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

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

Undeploys a configuration in the Workspace and discards its state.

Syntax

```
ConfigurationUndeploy(10);
```

Arguments

Field	Data Type	Description
configurationID	int	Configuration numeric identifier.

Response

No response.

Sample Code: C#

```
try
{
    LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    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";
    //
    //* 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

This method returns an object of type Configuration matching the configuration ID passed.

Syntax

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

Field	Data Type	Description
configurationID	int	Configuration identifier.

Response

Field	Data Type	Description
configuration	Configuration	Configuration object matching configuration id passed.

Sample Code: C#

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

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

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

    //** Write to the console all 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

This call takes the name of a configuration and returns an array of configurations matching that name. Configuration names are not unique. More than one configuration with a given name can exist. If configurations with that name do not exist, an empty array is returned.

Syntax

```
Configuration [] config = GetConfigurationByName(“Config9”);
```

Arguments

Field	Data Type	Description
name	string	Configuration name.

Response

Field	Data Type	Description
configuration[]	Configuration	Array of Configuration objects with the same name.

Sample Code: C#

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

GetMachine

This call takes the numeric identifier of a machine and returns its corresponding Machine object.

Syntax

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

Arguments

Field	Data Type	Description
machineID	int	Machine identifier.

Response

Field	Data Type	Description
machine	Machine	Machine object matching the machine identifier.

Sample Code: C#

```
try
{
    LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    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

This call takes a configuration identifier and a machine name and returns the matching Machine object.

Syntax

```
Machine mach = GetMachineByName(10, “Config9VM1”);
```

Arguments

Field	Data Type	Description
configurationId	int	Configuration identifier.
name	string	Machine name.

Response

Field	Data Type	Description
machine	Machine	Machine Object.

Sample Code: C#

```
try
{
    LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    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

This call takes a configuration name, searches for it in both the configuration library and Workspace and returns its corresponding Configuration object. An error is returned if more than one configuration exists with that name.

Syntax

```
Configuration config = GetSingleConfigurationByName(“Config9”);
```

Arguments

Field	Data Type	Description
name	string	Configuration name.

Response

Field	Data Type	Description
configuration	Configuration	Configuration object.

Sample Code: C#

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

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

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

    //** Write to the console all 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

This method returns an array of type Configuration. Depending on configuration type requested, one object is returned for each configuration in the configuration library or each configuration in the Workspace.

Syntax

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

Arguments

Field	Data Type	Description
configurationType	int	1= Workspace configurations, 2=Library configurations.

Response

Field	Data Type	Description
configurations[]	Configuration Array	Array of Configuration objects.

Sample Code: C#

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

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

This method returns an array of type Machine. The method returns one Machine object for each virtual machine in a configuration.

Syntax

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

Arguments

Field	Data Type	Description
configurationID	int	Configuration numeric identifier.

Response

Field	Data Type	Description
machine[]	Machine array	Array of Machine objects.

Sample Code: C#

```
try
{
    LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    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";
    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

This method allows you to create a LiveLink URL to a Library configuration.

Syntax

```
string url = LiveLink(“LiveLinkWin2K”));
```

Arguments

Field	Data Type	Description
configurationName	string	The name of a Library configuration.

Response

Field	Data Type	Description
URL	string	A string containing the configuration URL in the Library. The URL can be sent in an email to recreate the configuration when clicked.

Sample Code: C#

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

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

This method performs one of the following machine actions as indicated by the action identifier:

- 1 – Power on. Turns on a machine.
- 2 – Power off. Turns off a machine. Nothing is saved.
- 3 – Suspend. Freezes a machine CPU and state.
- 4 – Resume. Resumes a suspended machine.

- 5 – Reset. Reboots a machine.
- 6 – Snapshot. Save a machine state at a specific point in time.
- 7 – Revert. Returns a machine to a snapshot state.
- 8 – Shutdown. Shuts down a machine before turning off.

Syntax

```
MachinePerformAction(3);
```

Arguments

Field	Data Type	Description
action	int	Action to take on the machine.
machineID	int	Machine identifier.

Response

No response.

Sample Code: C#

```
try
{
    LabManagerSoap.VMwareLabManagerSOAPinterface binding = new
        LabManagerSoap.VMwareLabManagerSOAPinterface();
    binding.AuthenticationHeaderValue = new
        LabManagerSoap.AuthenticationHeader();
    binding.AuthenticationHeaderValue.username = "jaya";
    binding.AuthenticationHeaderValue.password = "Lab Manager";
    binding.AuthenticationHeaderValue.organizationname = "MyOrg";
    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 status—if 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();
        }
```

Index

C

CaptureUndeployConfiguration **24**
CertificateAccepter() **25**
CheckoutDeployConfiguration **23**
Code
    simple and advanced samples **16**

D

data types
    AuthenticationHeader **28**
    Configuration **29**
    for Lab Manager **28**
    Machine **30**
    primitive XML **27**
development environment, supported **10**

G

GetLMAPI() **24**

L

Lab Manager SOAP API
    defining **9**
languages, supported **10**

O

operations, supported **10**

R

RunQCTestset() **21**

S

security, using SSL **11**
SOAP API methods
    ConfigurationCapture **34**
    ConfigurationCheckout **36**
    ConfigurationClone **37**
    ConfigurationDelete **38**
    ConfigurationDeploy **40**
    ConfigurationPerformAction **41**
    ConfigurationSetPublicPrivate **43**
    ConfigurationUndeploy **44**
    GetConfiguration **45**
    GetConfigurationByName **47**
    GetMachine **48**
    GetMachineByName **50**
    GetSingleConfigurationByName **51**
    ListConfigurations **52**
    ListMachines **54**
    LiveLink **55**
    MachinePerformAction **56**
SOAP API, obtaining the WSDL **14**

U

users, authenticating **11**