Lab Manager SOAP API Reference

vCenter Lab Manager 4.0

EN-000175-00

vmware®

You can find the most up-to-date technical documentation on the VMware 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

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

To use the information in this guide, you should be familiar with the following items:

- 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 version of this book and other books, go to http://www.vmware.com/support/pubs.

Online and Telephone Support

To 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.

Support Offerings

To 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 vCenter 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 the 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), parameter types and the SOAP endpoint for the service.

This chapter includes the following topics:

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 about 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 15.

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 perform the following tasks:

- 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 23.

Lab Manager Data Objects

The Lab Manager SOAP API uses objects to interact with the data in your organization. Objects 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 use the Lab Manager data objects to perform operations such as query, clone, capture, and deploy on Lab Manager data. See "Lab Manager API Data Types" on page 19.

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 2005 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 2005 or later.

SOAP API 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 uses SSL exposes all SOAP API methods.

When you use the Web service URL to access 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 user credentials with each Lab Manager Web service method call. The required credentials are a Lab Manager user account, password, organization, and workspace name. You must provide the name of the organization and/or workspace that contains the objects on which you want to perform operations. The Lab Manager server authenticates these credentials.

NOTE You can use non-administrator credentials.

Getting Started with the Lab Manager SOAP API 2

You can use 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 includes the following topics:

Requirements for Developing an Application

Before you can start developing an application, review these requirements:

- VMware assumes that you are familiar with basic programming concepts and already have a programming development environment set up on your computer.
- You must have an instance of Lab Manager installed, configured, and running on your network.
- 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.
- Copy the code listing displayed in the next section, and paste it into your Microsoft Visual Studio 2005 environment.

Obtaining and Importing the WSDL File

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 this file, open Internet Explorer 5.5 or later and navigate to https://<hostname>/LabManager/SOAP/LabManager.asmx?WSDL for your Lab Manager server.

The WSDL file 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 to Your Development Environment

After you obtain the WSDL file, import it to 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. For instructions about other development platforms, see the product documentation for your platform.

Use Microsoft Visual Studio with the Lab Manager WSDL File

Microsoft Visual Studio programming languages access the Lab Manager SOAP API through objects that serve as proxies for their server-side counterparts. When managed code accesses XML Web services, 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 an Add a Web Reference wizard 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.

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

To add a Web reference by using Microsoft Visual Studio 2005

1 In Windows, select **Start** > **Microsoft Visual Studio .NET 2005**.

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

3 In the **URL** text box, type **`https://<hostname>/LabManager/SOAP/LabManager.asmx`** to obtain the service description of the Lab Manager Web service.

4 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.

---

5 Click **Yes**.

6 (Optional) If alert from the Visual Studio environment appears, 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 file.

7 Select the text in the Web reference name text box and type **`LabManager`**, the namespace used for this Web reference.

`LabManager` is one word, without spaces.

8 Click **Add Reference**.

9 (Optional) If a certificate warning message appears, click **Yes**.

10 Click **Yes**.

Visual Studio retrieves the service description and generates the LabManager proxy class 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 samples listed in "Simple and Advanced C# Code Samples" on page 13.

Simple and Advanced C# Code Samples

Review the setup requirements on "Requirements for Developing an Application" on page 11.

You can test basic API programming connectivity between your development workstation and your Lab Manager Web service by using the "Simple C# Example Console Application." If you are using a programming language other than C# and a Web services development environment other than Microsoft Visual Studio 2005, see the appropriate documentation for more information.

NOTE VMware assumes you are familiar with basic programming concepts and already have a programming development environment set up on your computer.

This code sample performs several simple tasks. The first two tasks are 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. To connect to the You must set up the certificate policy to accept all certificates.
- Makes a call to get a configuration object based on the object's name.
- Displays all configuration fields in the console.

Simple C# Example Console Application

Copy this sample code and paste it into your Microsoft Visual Studio 2005 environment.

```
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}
```

Advanced C# Sample Integrating Lab Manager and Quality Center

This C# .NET sample in this section is a more extensive, and more practical, sample of using the Lab Manager SOAP API. This code sample shows the integration of the Lab Manager SOAP API calls with a 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.
- Uses Mercury Quality Center to run a series of predefined tests on the deployed configuration.
- 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 methods:

- **CheckoutDeployConfiguration()** – Obtains the configuration from the library and deploys it to the Lab Manager workspace.
- **RunQCTestset()** – 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()** – 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.

Copy this sample code and paste it into your Microsoft Visual Studio 2005 environment.

```
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 Data Types 3

This chapter provides detailed information on the API data types in Lab Manager.

This chapter includes the following topics:

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 data 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 you write your client application, follow the data typing rules defined for your programming language and development environment. Your development tool maps the typed data in your programming language with these SOAP data types.

The Lab Manager data types are defined in the Lab Manager WSDL file. See "Lab Manager SOAP API Data Types" on page 20 for more information.

Table 3-2. Lab Manager SOAP API Data Types

Data Type	Description
AuthenticationHeader	Contains the user name, password, organization, and workspace 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 type passes the user name, password, organization, and workspace name of the caller in all Lab Manager SOAP API methods.

Supported API Calls

This data type supports all API calls.

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.
workspacename	string	Lab Manager workspace name

C# Sample Code

```
/**
** 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();
}
```

Configuration

This data type 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 that 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.
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 configuration. 2 = Captured library configuration. 3 = Gold Master captured library configuration. 4 = Archived library configuration.
autoDeleteInMiliSeconds	double	Time after which the configuration is deleted. If you type 0, the configuration is never deleted.
bucketName	string	Displays the workspace name if the configuration exists in a workspace; displays the organization name if the configuration exists in the library.
mustBeFenced	SOAPUtils.SOAPMustBeFenced	NotSpecified for either fenced or unfenced. True = for fenced only. False = for unfenced only.
autoDeleteDateTime	dateTime	Time after which a configuration is deleted.

Machine

This data type 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 if not deployed.
id	int	Machine identifier.

Table 3-5. Machine Fields (Continued)

Field	Data Type	Description
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.
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

Table 4-1 lists the vCenter Lab Manager SOAP API methods. For more detailed information about the methods and how to call them using C# .NET code samples, click the individual links.

Table 4-1. Lab Manager SOAP API Methods

Method	Description
"ConfigurationCapture" on page 24	Captures a workspace configuration and saves it to a specified Lab Manager datastore.
"ConfigurationCheckout" on page 25	Checks out a configuration from the configuration library and moves it to the workspace.
"ConfigurationClone" on page 26	Clones a configuration active in the workspace and saves it to storage.
"ConfigurationDelete" on page 27	Deletes a configuration from the workspace.
"ConfigurationDeploy" on page 28	Deploys a configuration in the workspace.
"ConfigurationPerformAction" on page 29	Performs an action on a configuration.
"ConfigurationSetPublicPrivate" on page 30	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 31	Undeploys a configuration in the workspace and discards its state.
"GetConfiguration" on page 32	Returns a Configuration object matching a configuration identifier.
"GetConfigurationByName" on page 33	Returns Configuration objects matching a configuration name. Configuration names are not guaranteed to be unique.
"GetCurrentOrganizationName" on page 34	Returns the current organization's name.
"GetCurrentWorkspaceName" on page 34	Returns the current workspace name.
"GetMachine" on page 35	Returns a Machine object matching a machine identifier.
"GetMachineByName" on page 36	Returns a Machine object matching a machine name.
"GetSingleConfigurationByName" on page 37	Returns a single Configuration object matching a configuration name.
"ListConfigurations" on page 38	Returns an array of Configuration objects in the workspace or configuration library.
"ListMachines" on page 39	Returns an array of Machine objects corresponding to the numeric identifier of a configuration.
"LiveLink" on page 40	Creates a URL to a configuration that can be emailed and clicked on to recreate the configuration.
"MachinePerformAction" on page 41	Performs an action on a machine.
"SetCurrentOrganizationByName" on page 42	Sets the organization to be used for each successive log in.
"SetCurrentWorkspaceByName" on page 43	Sets the workspace to be used for each successive log in.

ConfigurationCapture

This method captures a workspace configuration and saves it.

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.

C# Sample Code

```
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

This method checks out a configuration from the 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	The checked-out configuration name.

Response

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

C# Sample Code

```
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

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	The new workspace configuration name.

Response

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

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.
fenceMode	int	1 = Nonfenced 2 = FenceBlockInAndOut 3 = FenceAllowOutOnly 4 = FenceAllowInAndOut

Response

No response.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.

C# Sample Code

```
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

This call takes the name of a configuration and returns an array of configurations matching that name. Workspace configuration names are 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.

C# Sample Code

```
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

This call returns the current organization's name.

Syntax

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

Response

Field	Data Type	Description
organization name	string	Returns the current organization name.

C# Sample Code

```
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

This call returns the current workspace name.

Syntax

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

Response

Field	Data Type	Description
workspace name	string	Returns the current workspace name.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.

C# Sample Code

```
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

This call takes a configuration name, searches for it in both the configuration library and workspace and returns its corresponding Configuration object.

Syntax

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

Arguments

Field	Data Type	Description
name	string	Configuration name.

Response

Field	Data Type	Description
configuration	Configuration	Configuration object.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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.

C# Sample Code

```
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

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(1, 3);
```

Arguments

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

Response

No response.

C# Sample Code

```
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 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();
}
```

SetCurrentOrganizationByName

Sets the organization to be used for each successive log in. This method works when the organization name is empty in the authorization header.

Syntax

```
SetCurrentOrganizationByName(“MyOrganization”);
```

Arguments

Field	Data Type	Description
orgName	string	organization of which you are a member

C# Sample Code

```
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

Sets the workspace to be used for each successive log in. This method works when the workspace name is empty in the authorization header.

Syntax

```
SetCurrentWorkspaceByName(“MyOrganization“, “MyWorkspace”);
```

Arguments

Field	Data Type	Description
orgName	string	organization of which you are a member
workspaceName	sting	workspace of which you are a member

C# Sample Code

```
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();
}
```

Index