Web WakeUp Local services API

Local services

Web WakeUp provides an interface containing functions that allow you to find and wake a named computer to check its current status. The information provided here is for advanced users with a prerequisite knowledge of web service development.

You can open the `authorisationservices.asmx` and `localservices.asmx` pages from a web browser but only if you are logged on and browsing from the server where the pages are hosted.

Local services methods are available for use once an instance of the Web WakeUp API class has been created. The two ways of using the Web WakeUp API are:

Creating a service reference

In Microsoft Visual Studio, you can add a service reference to the web service URL

```
http://host/WebWakeUp/WebServices/LocalServices.asmx
```

This generates a proxy class, typically with the name `LocalServicesSoapClient`. In code, the proxy class can be used to access Web WakeUp API methods. For example, the following code fragment illustrates the FindMachines() method being invoked:

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var resultArr = client.FindMachines("MachineName");
 //do what you want to do with the results array.
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

You must call `client.Close()` after use

Creating a web reference

In Microsoft Visual Studio, there is an advanced option of the service reference called a web reference. For Web WakeUp you can add a web reference to the web service URL

```
http://host/WebServices/LocalServices.asmx
```

This generates a proxy class to the web service, typically with name `LocalServicesWse`. In code, the class can then be used to access Web WakeUp API methods. For example, the following code fragment shows a reference to `LocalServices.asmx` that has been created with the name `lssc` and the FindMachines() method being invoked on the new soap client that is created via the reference:

```
var client = new lssc.LocalServicesSoapClient();
String[] result = client.FindMachines("MachineName");
finally
{
 client.Close(); //MUST CALL THIS
}
```

As Web WakeUp uses windows authorization, the basic HTTP binding in the application's `app.config` file will also need the following security settings added:

On this page:

```
...
<security mode="TransportCredentialOnly">
    <transport clientCredentialType="Ntlm"/>
</security>
...
```

The following methods are available in the Web WakeUp API.

AddRegisteredMachine

`bool AddRegisteredMachine(string UserName, string MachineName)`

Purpose

Adds a computer to a user's list of registered computers

Inputs

- `UserName` – the user name for an authorized user in the format `DomainName\UserName`
- `MachineName` – the name for the computer to be added in the format `DomainName\MachineName`

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.AddRegisteredMachine( "ACME\Joe",
                                           "ACME\ACMEdvwks0012");
 if (result)
     Console.write("added ACMEdvwks0012 for Joe");
 else
     Console.write("could not add ACMEdvwks0012");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

ClearRegisteredMachine

`bool ClearRegisteredMachine(string UserName)`

Purpose

Clears the default computer registered to a user

Inputs

- `UserName` – the user name for an authorized user in the format `DomainName\UserName`

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.ClearRegisteredMachine( "ACME\Joe");
 if (result)
    Console.write("registered computer cleared");
 else
    Console.write("could not clear computer");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

DeleteRegisteredMachine

`bool DeleteRegisteredMachine(string UserName, string MachineName)`

Purpose

Deletes a computer from a user's list of registered computers using its name.

Inputs

- `UserName` – the user name for an authorized user in the format `DomainName\UserName`
- `MachineName` – the name for the computer to be deleted in the format `DomainName\MachineName`

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.DeleteRegisteredMachine("ACME\Joe",
                                             "ACME\ACMEdvwks0012");
 if (result)
     Console.write("removed ACMEdvwks0012 from Joe");
 else
     Console.write("could not remove ACMEdvwks0012");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

FindMachineExact

`bool FindMachineExact(string MachineName)`

Purpose

Finds a computer using its exact netBIOS name. This method is used to find a computer when the exact netBIOS name for the target computer is known.

Inputs

- `MachineName` – the netBIOS name for the computer to be found. Can also be in the format `DomainName\MachineName`. Anything but a complete match will result in failure.

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.FindMachineExact("ACME\ACMEdvwks0012");
 if (result)
     Console.write("found ACMEdvwks0012");
 else
     Console.write("cannot find ACMEdvwks0012");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

FindMachines

`string[] FindMachines(string SearchText)`

Purpose

Finds a number of computers matching a search string.

Inputs

- `SearchText` – the search text matched against the netBIOS name for the computers to be found. Can also be in the form `DomainName\SearchText`

Returns

- Array of strings containing the computers matching the search text. The number of items in this array is limited by the maximum number of returned items allowed.

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var resultArr = client.FindMachines("ACME\ACME");
 //process the results array
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

FindRegisteredMachine

`string FindRegisteredMachine(string UserName)`

Purpose

Finds the default computer registered to a user.

Inputs

- `UserName` – the user name for an authorized user in the format `DomainName\UserName`

Returns

- String containing the name of the computer registered to the user

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.FindRegisteredMachine( "ACME\Joe" );
 if (result == "")
     Console.write("no registered computer for Joe");
 else
     Console.write( result );
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

GetDisplayPages

`String[] GetDisplayPages()`

Purpose

Gets an array of the Web WakeUp pages configured for display by the administrator.

Inputs

- null

Returns

- A string array containing all the pages enabled by the Web WakeUp administrator

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.GetDisplayPages( );
 //process results array
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

GetMaxItemCount

`int GetMaxItemCount()`

Purpose

Gets the maximum number of items returned by Web WakeUp.

Inputs

- null

Returns

- Returns an integer that gives the maximum number of items that will be returned in the array returned by `FindMachines`

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var i = client.GetMaxItemCount();
 console.WriteLine("The max items value: {0}", i);
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

GetRegisteredMachines

`String[] GetRegisteredMachines(string UserName)`

Purpose

Inputs

- `UserName` – the user name for an authorized user in the format `DomainName\UserName`

Returns

- A string array containing all the computers registered for a user

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.GetRegisteredMachines( "ACME\Joe");
 //process results array
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

GetStatus

`MachineInfo GetStatus(string MachineName)`

Purpose

Determines the target machine's state using the 1E WakeUp infrastructure. The WakeUp Server is contacted, and will attempt to determine the machine's state using either an ICMP ping or a proprietary TCP message via the primary WakeUp Agent in the target machine's subnet.

`GetStatus()` will synchronously return the machine's most recent recorded state from a cache, but will asynchronously refresh this cache. The first time you call it, it may return an Unknown state if the machine has not been queried recently hence you may want to call it multiple times with at least a 10 second between each call and repeat until the machine's actual state (On or Off) is returned or until a reasonable number of retry attempts are exhausted.

In practice, a machine's status is normally returned in less than 30 seconds. If, after this period, `GetStatus()` still returns Unknown, it may indicate that the target machine cannot be reached within the 1E WakeUp infrastructure.

Making calls to `GetStatus()` in rapid succession is resource-intensive and will not reduce the time taken to determine the machine's state.

Inputs

- `MachineName` - the netBIOS name of the machine whose status is to be retrieved. You may specify the machine with or without its domain name prefix (for example: MYMACHINE or MYDOMAIN\MYMACHINE)

Returns

- `MachineInfo` - a structure containing details of the machine's status. The structure has the following fields:
- `_status` - A enumeration with the values: 0=Unknown, 1=On, 2=Off
- `IPAddress` - The IP address of the target machine
- `MACAddress` - The MAC address of the target machine

The `IPAddress` and `MACAddress` fields may not be present if the status of the machine could not be determined. If the machine has multiple network adapters, the information returned will apply to the network adapter which has the highest priority mapping as determined by the WakeUp Server boundary configuration in NightWatchman Management Center.

Example

C#

```
var
client = new lssc.LocalServices();

try
{
var result = client.GetStatus(@"acme\ACMEdvwks0012");
var status = result._status;
if (status == lssc.State.On)
{
        Console.WriteLine("The machine is on");
        Console.WriteLine("Its IP address is {0}, MAC address is {1}", result.IPAddress, result.MACAddress);
}
else if (status == lssc.State.Off)
{
        Console.WriteLine("The machine is off");
}
else // (status == lssc.State.Unknown)
{
        Console.WriteLine("The status of the machine is unknown");
}
}
catch (Exception ex)
{
// Handle the error case
}
finally
{
client.Close(); // Must call this
}
```

Ping

`bool Ping(string netBIOSName)`

Purpose

Checks to see if a named computer is awake using information retrieved from NightWatchman Management Center.

Inputs

- `netBIOSName` – the netBIOS name for the computer to check

Returns

- `Boolean` – success or failure

Example

C#

```
try
{
 var result = client.Ping("ACMEdvwks0012");
 if (result)
     Console.write("response for ACMEdvwks0012");
 else
     Console.write("no response for ACMEdvwks0012");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

RegisterMachine

`bool RegisterMachine(string UserName, string MachineName)`

Purpose

Adds the default registered computer for a user using its name.

Inputs

- `UserName` – the user name for an authorized user in the format `DomainName\UserName`
- `MachineName` – the netBIOS name for the computer to be set

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result =
 client.RegisterMachine("ACME\Joe",
                        "ACME\ACMEdvwks0012");
 if (result)
     Console.write("ACMEdvwks0012 default for Joe");
 else
     Console.write("could not set ACMEdvwks0012");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

WakeByName

`bool WakeByName(string netBIOSName)`

Purpose

Attempts to wake a computer using its netBIOS name. If you intend to make a high volume of calls to wake computers in a short space of time you should use the `WakeMachines` method, which supports waking a list of computers, and not `WakeByName`.

Inputs

- `netBIOSName` – the netBIOS name for the computer to wake up

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.WakeByName("ACMEdvwks0012");
 if (result)
     Console.write("sent wake up to ACMEdvwks0012");
 else
     Console.write("no wakeup sent to ACMEdvwks0012");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

WakeMachines

`bool WakeMachines(string MachineNames)`

Purpose

Attempts to wake a number of computers on a list. The `MachineNames` list is a comma separated list of computers in the form "domainA\name1, domainB\name2, ..." There is a limit on the number of computers that can be awoken in a single call governed by the maximum length of the input argument.

WakeUp has an absolute upper limit of 10,000 computers that can be awoken in a single operation, but given the default value for the `WakeMachines` input argument length of 65536 characters and an average `domain\computer` name length of around *26* characters the default limit on `WakeMachines` is around *2500* computers.

To increase the default input argument size, change some bindings in the `NWM.ServiceHost.exe.config` file, see setting the maximum number of computers returned.

Inputs

- `MachineNames` – a list of computer names to wake up

Returns

- `Boolean` – success or failure

Example

C#

```
var client = new lssc.LocalServicesSoapClient();
try
{
 var result = client.WakeMachines("ACMEdvwks0012,
                                  ACMEdvwks0013,
                                  ACMEdvwks0014");
 if (result)
     Console.write("sent wake up to computers");
 else
     Console.write("no wakeup sent to computers");
}
catch(Exception ex)
{
 //handle the error case
}
finally
{
 client.Close(); //MUST CALL THIS
}
```

The methods listed above may all be used in third-party applications. Any other methods in the Web WakeUp API are for internal use only and are not supported in third-party applications.