MeshAgent / modules / win-info.js

krayon007 Updated internal documentation f93610e · 2 years ago History

```
/*
Copyright 2019-2020 Intel Corporation

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/


//
// This module fetches various Windows System information, such as pending reboot statu
// volume defrag state, installed applications, windows update status, etc
//


var promise = require('promise');

//
// This function queries WMI to fetch Windows Update Status
//
function qfe()
{
    var child = require('child_process').execFile(process.env['windir'] + '\\System32\\
    child.stdout.str = ''; child.stdout.on('data', function (c) { this.str += c.toStrin
    child.stderr.str = ''; child.stderr.on('data', function (c) { this.str += c.toStrin
    child.waitExit();

    var lines = child.stdout.str.trim().split('\r\n');
```


MeshAgent / modules / win-info.js Top

Code Blame 260 lines (236 loc) · 9.16 KB Raw

```
function qfe()
    for (i = 1; i < lines.length; ++i)
    {
        var obj = {};
        tokens = lines[i].split(',');
        for (key = 0; key < keys.length; ++key)
        {
            if (tokens[key]) { obj[keys[key]] = tokens[key]; }
        }
        result.push(obj);
    }
    return (result);
}

// This function uses Windows Powershell to fetch metadata about the currently configur
function av()
{
```

```
    var child = require('child_process').execFile(process.env['windir'] + '\\System32\\
    if (child == null) { return ([]); }

    child.descriptorMetadata = 'process-manager';
    child.stdout.str = ''; child.stdout.on('data', function (c) { this.str += c.toStrin
    child.stderr.str = ''; child.stderr.on('data', function (c) { this.str += c.toStrin

    child.stdin.write('[reflection.Assembly]::LoadWithPartialName("system.core")\r\n');
    child.stdin.write('Get-WmiObject -Namespace "root/SecurityCenter2" -Class AntiVirus
    child.stdin.write('ForEach-Object -Process { ');
    child.stdin.write('$Bytes = [System.Text.Encoding]::UTF8.GetBytes($_.displayName);
    child.stdin.write('$EncodedText =[Convert]::ToBase64String($Bytes); ');
    child.stdin.write('Write-Host ("{0},{1}" -f $_.productState,$EncodedText); }\r\n');
    child.stdin.write('exit\r\n');
    child.waitExit();

    if (child.stdout.str == '') { return ([]); }

    var lines = child.stdout.str.trim().split('\r\n');
    var result = [];
    for (i = 0; i < lines.length; ++i)
    {
        var keys = lines[i].split(',');
        if(keys.length == 2)
        {
            var status = {};
            status.product = Buffer.from(keys[1], 'base64').toString();
            status.updated = (parseInt(keys[0]) & 0x10) == 0;
            status.enabled = (parseInt(keys[0]) & 0x1000) == 0x1000;
            result.push(status);
        }
    }
    return (result);
}

//
// This function uses the defrag system utility to query defrag state of the specified
//
// Note: options.volume must be specified
function defrag(options)
{
    var ret = new promise(function (res, rej) { this._res = res; this._rej = rej; });
    var path = '';

    switch(require('os').arch())
    {
        case 'x64':
            if (require('_GenericMarshal').PointerSize == 4)
            {
                // 32 Bit App on 64 Bit Windows
                ret._rej('Cannot defrag volume on 64 bit Windows from 32 bit applicatio
                return (ret);
            }
            else
            {
                // 64 Bit App
                path = process.env['windir'] + '\\System32\\defrag.exe';
            }
            break;
        case 'ia32':
            // 32 Bit App on 32 Bit Windows
            path = process.env['windir'] + '\\System32\\defrag.exe';
            break;
        default:
            ret._rej(require('os').arch() + ' not supported');
            return (ret);
            break;
    }

    ret.child = require('child_process').execFile(process.env['windir'] + '\\System32\\
    ret.child.promise = ret;
    ret.child.promise.options = options;
    ret.child.stdout.str = ''; ret.child.stdout.on('data', function (c) { this.str += c
    ret.child.stderr.str = ''; ret.child.stderr.on('data', function (c) { this.str += c
```

```
132         ret.child.on('exit', function (code)
133         {
134             var lines = this.stdout.str.trim().split('\r\n');
135             var obj = { volume: this.promise.options.volume };
136             for (var i in lines)
137             {
138                 var token = lines[i].split('=');
139                 if(token.length == 2)
140                 {
141                     switch(token[0].trim().toLowerCase())
142                     {
143                         case 'volume size':
144                             obj['size'] = token[1];
145                             break;
146                         case 'free space':
147                             obj['free'] = token[1];
148                             break;
149                         case 'total fragmented space':
150                             obj['fragmented'] = token[1];
151                             break;
152                         case 'largest free space size':
153                             obj['largestFragment'] = token[1];
154                             break;
155                     }
156                 }
157             }
158             this.promise._res(obj);
159         });
160         return (ret);
161     }
162
163     // Helper/Shortcut for registry query
164     function regQuery(H, Path, Key)
165     {
166         try
167         {
168             return(require('win-registry').QueryKey(H, Path, Key));
169         }
170         catch(e)
171         {
172             return (null);
```

```
187         {
188             ret = 'Windows Update';
189         }
190         else if ((tmp=regQuery(HKEY.LocalMachine, 'SYSTEM\\CurrentControlSet\\Control\\Sess
191         {
192             ret = 'File Rename';
193         }
194         else if (regQuery(HKEY.LocalMachine, 'SYSTEM\\CurrentControlSet\\Control\\ComputerN
195         {
196             ret = 'System Rename';
197         }
198         return (ret);
199     }
200
201     //
202     // Returns a promise that fetches the list of installed applications
203     //
204     function installedApps()
205     {
206         var promise = require('promise');
```

```
        var ret = new promise(function (a, r) { this._resolve = a; this._reject = r; });

        var code = "\
        var reg = require('win-registry');\
        var result = [];\
        var val, tmp;\
        var items = reg.QueryKey(reg.HKEY.LocalMachine, 'SOFTWARE\\\\Microsoft\\\\Windows\\
        for (var key in items.subkeys)\
        {\
            val = {};\
            try\
            {\
                val.name = reg.QueryKey(reg.HKEY.LocalMachine, 'SOFTWARE\\\\Microsoft\\\\Wi
            }\
            catch(e)\
            {\
                continue;\
            }\
            try\
            {\
                val.version = reg.QueryKey(reg.HKEY.LocalMachine, 'SOFTWARE\\\\Microsoft\\\
                if (val.version == '') { delete val.version; }\
            }\
            catch(e)\
            {\
            }\
            try\
            {\
                val.location = reg.QueryKey(reg.HKEY.LocalMachine, 'SOFTWARE\\\\Microsoft\\
                if (val.location == '') { delete val.location; }\
            }\
            catch(e)\
            {\
            }\
            result.push(val);\
        }\
        console.log(JSON.stringify(result,'', 1));process.exit();";

        ret.child = require('child_process').execFile(process.execPath, [process.execPath.s
        ret.child.promise = ret;
        ret.child.stdout.str = ''; ret.child.stdout.on('data', function (c) { this.str += c
        ret.child.on('exit', function (c) { this.promise._resolve(JSON.parse(this.stdout.st
        return (ret);
    }

    if (process.platform == 'win32')
    {
        module.exports = { qfe: qfe, av: av, defrag: defrag, pendingReboot: pendingReboot,
    }
    else
    {
        var not_supported = function () { throw (process.platform + ' not supported'); };
        module.exports = { qfe: not_supported, av: not_supported, defrag: not_supported, pe
    }
```