関係コード Seikactl

ソース解説をすることはしません。最新版との同期もとれていません。

概要

同梱している Seikactl コマンドのソースコード。サンプルとして公開。

ソースコード

Seikactlメインコード

Program.cs

```
using System;
using System.Diagnostics;
using System.IO;
using System.Threading;

namespace Seikactl
{
    class Program
    {
        static int Main(string[] args)
        {
            Opts opt = new Opts(args);
            WCFClient scc =new WCFClient();

            if (opt.isActive)
            {
                if (opt.isASWaitBoot)
                {
                    if (0 < opt.ASWaitTime)
                    {
                        int times = 0;
                        for (times = 0; times < opt.ASWaitTime;
times++)
                        {
                            try
                            {
                                scc.Version();
                                break;
                            }
                            catch (Exception)
                            {
```

```
                                Thread.Sleep(1000);
                            }
                        }
                        if (times == opt.ASWaitTime)
                        {
                            Console.WriteLine("not alive");
                            return 8;
                        }
                    }
                    else
                    {
                        Console.WriteLine(@"value {0} is invalid.",
opt.ASWaitTime);
                        return 8;
                    }
                }

                if (opt.isProdScan)
                {
                    try
                    {
                        scc.ProductScan();
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("fail. {0}", e.Message);
                        return 8;
                    }

                }
                if (opt.isBootHttpSrv)
                {
                    try
                    {
                        scc.BootHttpService();
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("fail. {0}", e.Message);
                        return 8;
                    }
                }
                if (opt.isASStartup)
                {
                    try
                    {
                        var ps = new Process();
                        ps.StartInfo.UseShellExecute = false;
                        ps.StartInfo.WorkingDirectory =
```

```
Path.GetFullPath(opt.ASBootPath);
                        ps.StartInfo.FileName = Path.Combine(
opt.ASBootPath, "AssistantSeika.exe");
                        ps.Start();
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("fail. {0}", e.Message);
                        return 8;
                    }
                }
                if (opt.isASShutdown)
                {
                    try
                    {
                        scc.Shutdown();
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("fail. {0}", e.Message);
                        return 8;
                    }
                }
            }

            return 0;

        }
    }
}
```

オプション解析

Opt.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

namespace Seikactl
{
    class Opts
    {
        public bool isActive { get; }
        public bool isProdScan { get; }
        public bool isBootHttpSrv { get; }
        public bool isASWaitBoot { get; }
```

```
        public int ASWaitTime { get; }
        public bool isASShutdown { get; }
        public bool isASStartup { get; }
        public string ASBootPath { get; }

        public Opts(string[] args)
        {
            isActive = false;
            isProdScan = false;
            isBootHttpSrv = false;
            isASWaitBoot = false;
            isASStartup = false;
            isASShutdown = false;
            ASWaitTime = 180;

            if (args.Length == 0)
            {
                Console.WriteLine("Seikactl 20230319/u");
                Console.WriteLine("");
                Console.WriteLine("usage: Seikactl < boot PATH |
waitboot [N] | prodscan | boothttp | shutdown >");
                Console.WriteLine(@"  boot     : Start AssistantSeika
from PATH.");
                Console.WriteLine(@"  waitboot : Wait N seconds for
AssistantSeika to start. default Wait 180 seconds.");
                Console.WriteLine(@"  prodscan : Run a ""product
scan"". Note: Do not run continuously. ");
                Console.WriteLine(@"  boothttp : Run or reboot ""HTTP
Service"".");
                Console.WriteLine(@"  shutdown : Shutdown
AssistantSeika.");
                Console.WriteLine("");
            }
            else
            {
                try
                {
                    switch (args[0])
                    {
                        case "prodscan":
                            isProdScan = true;
                            break;

                        case "boothttp":
                            isBootHttpSrv = true;
                            break;

                        case "waitboot":
                            isASWaitBoot = true;
```

```
                            if (args.Length == 2)
                            {
                                ASWaitTime = int.Parse(args[1]);
                            }
                            break;

                        case "boot":
                            isASStartup = true;
                            if (args.Length == 2)
                            {
                                ASBootPath = args[1];
                            }
                            else
                            {
                                Console.WriteLine("need PATH.");
                            }
                            break;

                        case "shutdown":
                            isASShutdown = true;
                            break;

                        default:
                            Console.WriteLine("unknown {0}", args[0]);
                            break;
                    }


                    if (isProdScan || isBootHttpSrv || isASWaitBoot ||
isASShutdown || isASStartup)
                    {
                        isActive = true;
                    }
                }
                catch (Exception e)
                {
                    Console.WriteLine("err:{0}", e.Message);
                    isActive = false;
                }
            }
        }
    }
}
```

WCFクライアントコード

WCFClient.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Threading;

namespace Seikactl
{
    public class WCFClient
    {
        NetNamedPipeBinding Binding = new NetNamedPipeBinding();
        TimeSpan keeptime = new TimeSpan(24, 00, 00);

        string BaseAddr =
"net.pipe://localhost/EchoSeika/CentralGate/ApiEntry";

        public WCFClient()
        {
        }

        ~WCFClient()
        {
        }

        private ChannelFactory<IScAPIs> CreateChannelFactory()
        {
            var ans = new ChannelFactory<IScAPIs>(Binding, new
EndpointAddress(BaseAddr));

            while (ans.State != CommunicationState.Created)
            {
                Thread.Sleep(10);
            }

            return ans;
        }

        private IScAPIs CreateChannel(ChannelFactory<IScAPIs>
ChannelSc)
        {
            var ans = ChannelSc.CreateChannel();
            (ans as IContextChannel).OperationTimeout = keeptime;

            while (ChannelSc.State != CommunicationState.Opened)
            {
                Thread.Sleep(10);
            }
```

```
            return ans;
        }

        public string Version()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.Version();
            cf.Close();
            return ans;
        }

        public void ProductScan()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            api.ProductScan();
            cf.Close();
        }

        public void BootHttpService()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            api.BootHttpService();
            cf.Close();
        }

        public Dictionary<int, string> AvatorList()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.AvatorList();
            cf.Close();
            return ans;
        }

        public Dictionary<int, Dictionary<string, string>>
AvatorList2()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.AvatorList2();
            cf.Close();
            return ans;
        }
        public Dictionary<int, Dictionary<string, string>>
AvatorListDetail2()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
```

```
            var list1 = api.AvatorListDetail2();
            cf.Close();

            var ans = new Dictionary<int, Dictionary<string,
string>>();
            foreach(var item1 in list1)
            {
                if (item1.Value["isalias"] == "False")
                {
                    ans.Add(item1.Key, new Dictionary<string,
string>());

                    foreach (var item2 in item1.Value)
                    {
                        ans[item1.Key].Add(item2.Key, item2.Value);
                    }
                }
            }

            return ans;
        }

        public Dictionary<string, Dictionary<string, Dictionary<string,
decimal>>> GetDefaultParams2(int cid)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.GetDefaultParams2(cid);
            cf.Close();
            return ans;
        }

        public Dictionary<string, Dictionary<string, Dictionary<string,
decimal>>> GetCurrentParams2(int cid)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.GetCurrentParams2(cid);
            cf.Close();
            return ans;
        }

        public void ResetParams2(int cid)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            api.ResetParams2(cid);
            cf.Close();
        }
```

```
        public double Talk(int cid, string talktext, string filepath,
Dictionary<string, decimal> effects, Dictionary<string, decimal>
emotions)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.Talk(cid, talktext, filepath == "" ? "" :
MakeFullPath(filepath), effects, emotions);
            cf.Close();
            return ans;
        }
        public double Talk2(int cid, string[] talktexts, string
filepath, Dictionary<string, decimal> effects, Dictionary<string,
decimal> emotions)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            var ans = api.Talk2(cid, talktexts, filepath == "" ? "" :
MakeFullPath(filepath), effects, emotions);
            cf.Close();
            return ans;
        }

        public void TalkAsync(int cid, string talktext,
Dictionary<string, decimal> effects, Dictionary<string, decimal>
emotions)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            api.TalkAsync(cid, talktext, effects, emotions);
            cf.Close();
        }

        public void TalkAsync2(int cid, string[] talktexts,
Dictionary<string, decimal> effects, Dictionary<string, decimal>
emotions)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            api.TalkAsync2(cid, talktexts, effects, emotions);
            cf.Close();
        }

        public int CheckCid(int cid)
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);
            int ans = api.CheckCid(cid);
            cf.Close();
            return ans;
```

```
        }

        public void Shutdown()
        {
            var cf = CreateChannelFactory();
            var api = CreateChannel(cf);

            api.Shutdown();

            cf.Close();
        }

        private string MakeFullPath(string filepath)
        {
            return Path.GetFullPath(filepath);
        }

    }

    [ServiceContract(SessionMode = SessionMode.Required)]
    public interface IScAPIs
    {
        [OperationContract]
        string Version();

        [OperationContract]
        void ProductScan();

        [OperationContract]
        void BootHttpService();

        [OperationContract]
        Dictionary<int, string> AvatorList();

        [OperationContract]
        Dictionary<int, Dictionary<string, string>> AvatorList2();

        [OperationContract]
        Dictionary<int, Dictionary<string, string>>
AvatorListDetail2();

        [OperationContract]
        Dictionary<string, Dictionary<string, Dictionary<string,
decimal>>> GetDefaultParams2(int cid);

        [OperationContract]
        Dictionary<string, Dictionary<string, Dictionary<string,
decimal>>> GetCurrentParams2(int cid);
```

```
        [OperationContract]
        void ResetParams2(int cid);

        [OperationContract]
        double Talk(int cid, string talktext, string filepath,
Dictionary<string, decimal> effects, Dictionary<string, decimal>
emotions);

        [OperationContract]
        double Talk2(int cid, string[] talktexts, string filepath,
Dictionary<string, decimal> effects, Dictionary<string, decimal>
emotions);

        [OperationContract]
        void TalkAsync(int cid, string talktext, Dictionary<string,
decimal> effects, Dictionary<string, decimal> emotions);

        [OperationContract]
        void TalkAsync2(int cid, string[] talktexts, Dictionary<string,
decimal> effects, Dictionary<string, decimal> emotions);

        [OperationContract]
        double Save(int cid, string talktext, string filepath,
Dictionary<string, decimal> effects, Dictionary<string, decimal>
emotions);

        [OperationContract]
        int CheckCid(int cid);

        [OperationContract]
        void Shutdown();

    }

}
```

技術資料, Windows, wcf