関係コード３

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

音声合成製品制御コードサンプル。

ソースコード

VOICEROID2用制御コード

ScDeviceDriver.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Codeer.Friendly;
using Codeer.Friendly.Dynamic;
using Codeer.Friendly.Windows;
using Codeer.Friendly.Windows.Grasp;
using Codeer.Friendly.Windows.NativeStandardControls;
using RM.Friendly.WPFStandardControls;
using ScDriver;

namespace ScDriver.VOICEROID2
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private readonly string DrvName =
"Voiceroid2.Driver@echoseika.hgotoh.jp";
        private readonly string DrvVersion = "20200430/c";
        private readonly string DrvProdName = "VOICEROID2";
        private readonly int CidBase = 2000;

        public SemaphoreSlim Semaphore = new SemaphoreSlim(1, 1);

        private WindowsAppFriend _app = null;
        private WindowControl uiTreeTop = null;
        private WPFTabControl VoicePresetTab = null;
        private WPFTabControl TuneTab = null;
        private WPFTextBox    TalkTextBox = null;
```

```
        private WPFButtonBase PlayButton = null;
        private WPFButtonBase SaveButton = null;
        private WPFListView AvatorListView_std = null;
        private WPFListView AvatorListView_usr = null;

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
            ScDrvProdName = DrvProdName;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            Process p = GetVoiceroidEditorProcess();

            if (p != null)
            {
                try
                {
                    _app = new WindowsAppFriend(p);
                    uiTreeTop = WindowControl.FromZTop(_app);

                    //判明しているGUI要素特定
                    var tabs =
uiTreeTop.GetFromTypeFullName("AI.Framework.Wpf.Controls.TitledTabContr
ol");
                    VoicePresetTab = new WPFTabControl(tabs[0]);  //ボイス
（プリセット）のタブコントロール
                    TuneTab = new WPFTabControl(tabs[1]);  //チューニング
のタブコントロール

                    var editUis =
uiTreeTop.GetFromTypeFullName("AI.Talk.Editor.TextEditView")[0].Logical
Tree();
                    TalkTextBox = new WPFTextBox(editUis[4]);     //テキ
ストボックス
                    PlayButton = new WPFButtonBase(editUis[6]);  //再生ボ
タン
                    SaveButton = new WPFButtonBase(editUis[24]); //音声保
存ボタン

                    //標準タブにいる各話者毎のGUI要素データを取得
                    TuneTab.EmulateChangeSelectedIndex(1);
                    VoicePresetTab.EmulateChangeSelectedIndex(0);
                    AvatorListView_std = new
WPFListView(uiTreeTop.GetFromTypeFullName("System.Windows.Controls.List
View")[0]);
                    ScanPreset(AvatorListView_std, 0, 0);
```

```
                    //ユーザータブにいる各話者プリセット毎のGUI要素データを取得
                    TuneTab.EmulateChangeSelectedIndex(1);
                    VoicePresetTab.EmulateChangeSelectedIndex(1);
                    AvatorListView_usr = new
WPFListView(uiTreeTop.GetFromTypeFullName("System.Windows.Controls.List
View")[1]);
                    ScanPreset(AvatorListView_usr,
AvatorListView_std.ItemCount, 1);
                }
                catch (Exception e)
                {
                    ThrowException(string.Format(@"{0} {1}", e.Message,
e.StackTrace));
                }
            }

            IsAlive = AvatorParams.Count != 0;
        }

        private Process GetVoiceroidEditorProcess()
        {
            string winTitle1 = "VOICEROID2";
            string winTitle2 = winTitle1 + "*";

            int RetryCount = 3;
            int RetryWaitms = 500;
            Process p = null;

            for (int i = 0; i < 3; i++)
            {
                Process[] ps = Process.GetProcesses();

                foreach (Process pitem in ps)
                {
                    if ((pitem.MainWindowHandle != IntPtr.Zero) &&
                        ((pitem.MainWindowTitle.Equals(winTitle1)) ||
(pitem.MainWindowTitle.Equals(winTitle2))))
                    {
                        p = pitem;
                        break;
                    }
                }
                if (p != null) break;
                if (i < (RetryCount - 1)) Thread.Sleep(RetryWaitms);
            }

            return p;
        }

        public void Dispose()
```

```
        {
            Dispose(true);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string talkText)
        {
            Semaphore.Wait();

            Stopwatch sw = new Stopwatch();
            bool iconFind = false;
            int avatorIndex = ConvertAvatorIndex(cid);

            //最後の確認ダイアログを殺し切れていなかった時のための処理
            var infoOldDlgs = WindowControl.GetFromWindowText(_app, "情
報");
            try
            {
                if ((infoOldDlgs.Length != 0) &&
(infoOldDlgs[0].WindowClassName == "#32770"))
                {
                    //OKボタンを押す
                    NativeButton btn = new
NativeButton(infoOldDlgs[0].IdentifyFromWindowClass("Button"));
                    btn.EmulateClick(new Async());
                    Thread.Sleep(10);
                }
            }
            catch (Exception)
            {
                //
            }

            if (PlayButton == null) return 0.0;
            if (SaveButton == null) return 0.0;
            if (TalkTextBox == null) return 0.0;

            TuneTab.EmulateChangeSelectedIndex(1);

            //話者切り替え
            AvatorSelect(avatorIndex);

            //音声保存ボタンを使った再生終了判定を止めて、再生ボタンのアイコンの状
態で判定する方法に変更する。
            var items01 =
```

```
PlayButton.LogicalTree(TreeRunDirection.Descendants).ByType("System.Windows.Controls.Image");
            dynamic playButtonImage1 = items01[0].Dynamic(); //再生アイコン。このアイコンが有効時？の判定はまだないな…
            dynamic playButtonImage2 = items01[1].Dynamic(); //停止アイコン。処理ではこのアイコンのプロパティを持ている

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            //再生中なので再生終了を待つ
            //※再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
            iconFind = playButtonImage2.IsVisible;
            if (iconFind)
            {
                while (iconFind)
                {
                    Thread.Sleep(10);
                    iconFind = playButtonImage2.IsVisible;
                }
            }

            TalkTextBox.EmulateChangeText(talkText);
            Thread.Sleep(10);

            sw.Start();

            PlayButton.EmulateClick();

            //再生開始を待つ
            //再生ボタンのアイコンが停止アイコンに切り替わるのを待つ
            iconFind = playButtonImage2.IsVisible;
            if (!iconFind)
            {
                while (!iconFind)
                {
                    Thread.Sleep(10);
                    iconFind = playButtonImage2.IsVisible;
                }
            }

            //再生終了を待つ
            //再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
            iconFind = playButtonImage2.IsVisible;
            if (iconFind)
            {
                while (iconFind)
                {
                    Thread.Sleep(10);
                    iconFind = playButtonImage2.IsVisible;
                }
```

```
            }

            sw.Stop();
            Semaphore.Release();

            return sw.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override void PlayAsync(int cid, string talkText)
        {
            Task.Run(() =>
            {
                Semaphore.Wait();

                bool iconFind = false;
                int avatorIndex = ConvertAvatorIndex(cid);

                //最後の確認ダイアログを殺し切れていなかった時のための処理
                var infoOldDlgs = WindowControl.GetFromWindowText(_app,
"情報");
                try
                {
                    if ((infoOldDlgs.Length != 0) &&
(infoOldDlgs[0].WindowClassName == "#32770"))
                    {
                        //OKボタンを押す
                        NativeButton btn = new
NativeButton(infoOldDlgs[0].IdentifyFromWindowClass("Button"));
                        btn.EmulateClick(new Async());
                        Thread.Sleep(10);
                    }
                }
                catch (Exception)
                {
                    //
                }

                if (PlayButton == null) return;
                if (SaveButton == null) return;
                if (TalkTextBox == null) return;

                TuneTab.EmulateChangeSelectedIndex(1);
```

```
                //話者切り替え
                AvatorSelect(avatorIndex);

                //音声保存ボタンを使った再生終了判定を止めて、再生ボタンのアイコンの状態で判定する方法に変更する。
                var items01 = PlayButton.LogicalTree(TreeRunDirection.Descendants).ByType("System.Windows.Controls.Image");
                dynamic playButtonImage1 = items01[0].Dynamic(); //再生アイコン。このアイコンが有効時？の判定はまだないな…
                dynamic playButtonImage2 = items01[1].Dynamic(); //停止アイコン。処理ではこのアイコンのプロパティを持ている

                ApplyEffectParameters(avatorIndex);
                ApplyEmotionParameters(avatorIndex);

                //再生中なので再生終了を待つ
                //※再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
                iconFind = playButtonImage2.IsVisible;
                if (iconFind)
                {
                    while (iconFind)
                    {
                        Thread.Sleep(10);
                        iconFind = playButtonImage2.IsVisible;
                    }
                }

                TalkTextBox.EmulateChangeText(talkText);
                Thread.Sleep(10);

                PlayButton.EmulateClick();

                //再生開始を待つ
                //再生ボタンのアイコンが停止アイコンに切り替わるのを待つ
                iconFind = playButtonImage2.IsVisible;
                if (!iconFind)
                {
                    while (!iconFind)
                    {
                        Thread.Sleep(10);
                        iconFind = playButtonImage2.IsVisible;
                    }
                }

                //再生終了を待つ
                //再生ボタンのアイコンが再生アイコンに切り替わるのを待つ
                iconFind = playButtonImage2.IsVisible;
                if (iconFind)
                {
                    while (iconFind)
```

```
                    {
                        Thread.Sleep(10);
                        iconFind = playButtonImage2.IsVisible;
                    }
                }

                Semaphore.Release();
            });
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string talkText, string
saveFilename)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            //最後の確認ダイアログを殺し切れていなかった時のための処理
            var infoOldDlgs = WindowControl.GetFromWindowText(_app, "情
報");
            try
            {
                if ((infoOldDlgs.Length != 0) &&
(infoOldDlgs[0].WindowClassName == "#32770"))
                {
                    //OKボタンを押す
                    NativeButton btn = new
NativeButton(infoOldDlgs[0].IdentifyFromWindowClass("Button"));
                    btn.EmulateClick(new Async());
                    Thread.Sleep(10);
                }
            }
            catch (Exception)
            {
                //
            }

            if (PlayButton == null) return 0.0;
            if (SaveButton == null) return 0.0;
            if (TalkTextBox == null) return 0.0;

            TuneTab.EmulateChangeSelectedIndex(1);

            //話者切り替え
```

```
            AvatorSelect(avatorIndex);

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            if (!SaveButton.IsEnabled)
            {
                while (!SaveButton.IsEnabled)
                {
                    Thread.Sleep(10);
                }
            }

            TalkTextBox.EmulateChangeText(talkText);
            Thread.Sleep(10);

            //VOICEROID2 Editorの音声保存ボタンを押す
            SaveButton.EmulateClick(new Async());

            bool finish_savefileSetup = false;
            bool skip_saveOptionDlg = false;
            while (finish_savefileSetup == false)
            {
                //音声保存の設定ダイアログ処理
                var saveDlgs = WindowControl.GetFromWindowText(_app, "音
声保存");
                try
                {
                    if ((!skip_saveOptionDlg) && (saveDlgs.Length !=
0))
                    {
                        //OKボタンを押す
                        WPFButtonBase btn = new
WPFButtonBase(saveDlgs[0].LogicalTree(TreeRunDirection.Descendants).ByT
ype("System.Windows.Controls.Button")[0]);
                        btn.EmulateClick(new Async());
                        skip_saveOptionDlg = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                //名前を付けて保存ダイアログで名前を設定
                var fileDlgs = WindowControl.GetFromWindowText(_app, "名
前を付けて保存");
                try
                {
                    if ((fileDlgs.Length != 0) &&
(fileDlgs[0].WindowClassName == "#32770"))
```

```
                    {
                        //NativeEdit fileName = new
NativeEdit(fileDlgs[0].GetFromWindowClass("Edit")[0]);
                        //NativeButton btn = new
NativeButton(fileDlgs[0].GetFromWindowClass("Button")[0]);

                        // https://github.com/mikoto2000/TTSController
UI特定の記述を参照
                        NativeButton btn = new
NativeButton(fileDlgs[0].IdentifyFromDialogId(1));
                        NativeEdit saveNameText = new
NativeEdit(fileDlgs[0].IdentifyFromZIndex(11, 0, 4, 0, 0));

                        //ファイル名を設定
                        saveNameText.EmulateChangeText(saveFilename);
                        Thread.Sleep(100);

                        //OKボタンを押す
                        btn.EmulateClick(new Async());
                        finish_savefileSetup = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                Thread.Sleep(10);
            }

            bool finish_fileSave = false;
            while (finish_fileSave == false)
            {
                //上書き確認ダイアログの処理2
                var overwriteDlgs2 =
WindowControl.GetFromWindowText(_app, "ファイル保存");
                try
                {
                    if ((overwriteDlgs2.Length != 0) &&
(overwriteDlgs2[0].WindowClassName == "#32770"))
                    {
                        //上書きボタンを押す
                        NativeButton btn = new
NativeButton(overwriteDlgs2[0].GetFromWindowClass("Button")[0]);
                        btn.EmulateClick(new Async());
                    }
                }
                catch (Exception)
                {
```

```
                    //
                }

                //最後の確認ダイアログの処理
                var infoDlgs = WindowControl.GetFromWindowText(_app, "情報");
                try
                {
                    if ((infoDlgs.Length != 0) && (infoDlgs[0].WindowClassName == "#32770"))
                    {
                        //OKボタンを押す
                        NativeButton btn = new NativeButton(infoDlgs[0].IdentifyFromWindowClass("Button"));
                        btn.EmulateClick(new Async());
                        finish_fileSave = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                Thread.Sleep(10);
            }

            return 0.0;
        }

        /// <summary>
        ///感情パラメタをデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEmotion(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIndex] as AvatorParam;

            foreach (KeyValuePair<string, EffectValueInfo> item in avator.VoiceEmotions_default)
            {
                avator.VoiceEmotions[item.Key].value = item.Value.value;
            }

            ApplyEmotionParameters(avatorIndex);
        }

        /// <summary>
        ///音声効果をデフォルト値に戻す
```

```
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEffect(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIndex] as
AvatorParam;

            foreach (var effect in avator.VoiceEffects_default)
            {
                avator.VoiceEffects[effect.Key].value =
effect.Value.value;
            }

            ApplyEffectParameters(avatorIndex);
        }

        public override void Dispose(bool disposing)
        {
            if (Disposed) return;

            if (disposing)
            {
                foreach (var item in AvatorParams)
                {
                    ResetVoiceEffect(item.Key + CidBase);
                    ResetVoiceEmotion(item.Key + CidBase);
                }

                AvatorParams.Clear();
                _app.Dispose();
            }

            Disposed = true;
        }

        private void ScanPreset(WPFListView avatorListView, int
idxBase, int presetTabIndex)
        {
            for (int avatorIdx = 0; avatorIdx <
avatorListView.ItemCount; avatorIdx++)
            {
                VOICEROID2.AvatorParam avator = new
VOICEROID2.AvatorParam();

                avator.AvatorUI = new VOICEROID2.AvatorUIParam();
                avator.VoiceEmotions = new Dictionary<string,
EffectValueInfo>();
                avator.VoiceEmotions_default = new Dictionary<string,
```

```
EffectValueInfo>();
                avator.AvatorUI.EmotionSliderIndexs = new
Dictionary<string, int>();
                avator.AvatorUI.EmotionSliders = new Dictionary<int,
WPFSlider>();

                avatorListView.EmulateChangeSelectedIndex(avatorIdx);
                TuneTab.EmulateChangeSelectedIndex(1); //ボイスタブ

                //プリセット名取得（話者名）
                var params1 =
TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("AI.Framework.W
pf.Controls.TextBoxEx")[0];
                WPFTextBox nameTextBox = new WPFTextBox(params1);

                avator.AvatorName = nameTextBox.Text;

                //スライダーの配列を取得(共通)
                try
                {
                    TuneTab.EmulateChangeSelectedIndex(1); //チューニング:
ボイスタブ
                    var params2 =
TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("System.Windows
.Controls.Slider");
                    avator.AvatorUI.VolumeSlider = new
WPFSlider(params2[0]);
                    avator.AvatorUI.SpeedSlider = new
WPFSlider(params2[1]);
                    avator.AvatorUI.PitchSlider = new
WPFSlider(params2[2]);
                    avator.AvatorUI.IntonationSlider = new
WPFSlider(params2[3]);
                    avator.AvatorUI.ShortPauseSlider = new
WPFSlider(params2[4]);
                    avator.AvatorUI.LongPauseSlider = new
WPFSlider(params2[5]);
                    avator.AvatorUI.WithEmotionParams = false;

                    avator.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                    {
                        {EnumVoiceEffect.volume,     new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Value),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.speed,      new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Value),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.pitch,      new
```

```
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.PitchSlider.Value),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.intonation, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.shortpause, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.longpause,  new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Value
), Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Maximum), 0.01m)}
                    };
                    avator.VoiceEffects = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                    {
                        {EnumVoiceEffect.volume,     new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Value),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.VolumeSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.speed,      new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Value),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.SpeedSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.pitch,      new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.PitchSlider.Value),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.PitchSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.intonation, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.IntonationSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.shortpause, new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Valu
e), Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.ShortPauseSlider.Maximum), 0.01m)},
                        {EnumVoiceEffect.longpause,  new
EffectValueInfo(Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Value
), Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Minimum),
Convert.ToDecimal(avator.AvatorUI.LongPauseSlider.Maximum), 0.01m)}
                    };
                }
                catch (Exception ep2)
                {
                    ThrowException(string.Format("ep2 fail. unknown
gui(LinearFader capture).{0}", ep2.Message));
```

```
                }

                //スライダーの配列を取得(スタイル)
                try
                {
                    var params3 =
TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("System.Windows
.Controls.ListBox").Single();

                    WPFListBox sliders = new WPFListBox(params3);

                    if ((sliders != null) && (sliders.ItemCount != 0))
                    {
                        for (int sidx = 0; sidx < sliders.ItemCount;
sidx++)
                        {
                            var sitem = sliders.GetItem(sidx);
                            var textblocks =
sitem.VisualTree().ByType("System.Windows.Controls.TextBlock");
                            WPFSlider slider = new
WPFSlider(sitem.VisualTree().ByType("AI.Framework.Wpf.Controls.LinearFa
der").Single());
                            WPFTextBlock emoname = new
WPFTextBlock(textblocks[textblocks.Count > 2 ? (textblocks.Count - 2) :
0]);

avator.VoiceEmotions_default.Add(emoname.Text, new
EffectValueInfo(Convert.ToDecimal(slider.Value),
Convert.ToDecimal(slider.Minimum), Convert.ToDecimal(slider.Maximum),
0.01m));
                            avator.VoiceEmotions.Add(emoname.Text, new
EffectValueInfo(Convert.ToDecimal(slider.Value),
Convert.ToDecimal(slider.Minimum), Convert.ToDecimal(slider.Maximum),
0.01m));
avator.AvatorUI.EmotionSliderIndexs.Add(emoname.Text, sidx);
                            avator.AvatorUI.EmotionSliders.Add(sidx,
slider); //将来のため保持しているだけ。
                        }

                        avator.AvatorUI.WithEmotionParams = true;
                    }
                }
                catch (Exception ep3)
                {
                    ThrowException(string.Format("ep3 fail. unknown
gui(Style LinearFader capture).{0}", ep3.Message));
                }

                avator.AvatorIndex = idxBase + avatorIdx;
                avator.AvatorUI.PresetTabIndex = presetTabIndex;
                avator.AvatorUI.IndexOnPresetTab = avatorIdx;
```

```
                AvatorParams.Add(idxBase + avatorIdx, avator);
            }
        }

        private void AvatorSelect(int avatorIndex)
        {
            VOICEROID2.AvatorParam avator = AvatorParams[avatorIndex]
as VOICEROID2.AvatorParam;

            if (AvatorListView_std == null) return;
            if (AvatorListView_usr == null) return;

VoicePresetTab.EmulateChangeSelectedIndex(avator.AvatorUI.PresetTabInde
x);
            switch (avator.AvatorUI.PresetTabIndex)
            {
                case 0:
AvatorListView_std.EmulateChangeSelectedIndex(avator.AvatorUI.IndexOnPr
esetTab);
                    break;

                case 1:
AvatorListView_usr.EmulateChangeSelectedIndex(avator.AvatorUI.IndexOnPr
esetTab);
                    break;
            }
        }

        private void ApplyEmotionParameters(int avatorIndex)
        {
            VOICEROID2.AvatorParam avator = AvatorParams[avatorIndex]
as VOICEROID2.AvatorParam;

            //スタイルを持っている話者なら処理する
            if (avator.AvatorUI.WithEmotionParams)
            {
                TuneTab.EmulateChangeSelectedIndex(1); //ボイスタブ
                WPFListBox emoList = new
WPFListBox(TuneTab.VisualTree(TreeRunDirection.Descendants).ByType("Sys
tem.Windows.Controls.ListBox").Single());

                foreach (KeyValuePair<string, EffectValueInfo> item in
avator.VoiceEmotions)
                {
                    double p =
Convert.ToDouble(avator.VoiceEmotions[item.Key].value);
                    int eidx =
avator.AvatorUI.EmotionSliderIndexs[item.Key];
```

```
                    WPFSlider slider = new
WPFSlider(emoList.GetItem(eidx).VisualTree().ByType("AI.Framework.Wpf.C
ontrols.LinearFader").Single());
                    slider["Value"](p);
                }
            }
        }

        private void ApplyEffectParameters(int avatorIndex)
        {
            WPFSlider slider = null;
            VOICEROID2.AvatorParam avator = AvatorParams[avatorIndex]
as VOICEROID2.AvatorParam;

            TuneTab.EmulateChangeSelectedIndex(1);

            foreach (KeyValuePair<EnumVoiceEffect, EffectValueInfo>
item in avator.VoiceEffects)
            {
                switch (item.Key)
                {
                    case EnumVoiceEffect.volume:
                        slider = avator.AvatorUI.VolumeSlider;
                        break;

                    case EnumVoiceEffect.speed:
                        slider = avator.AvatorUI.SpeedSlider;
                        break;

                    case EnumVoiceEffect.pitch:
                        slider = avator.AvatorUI.PitchSlider;
                        break;

                    case EnumVoiceEffect.intonation:
                        slider = avator.AvatorUI.IntonationSlider;
                        break;

                    case EnumVoiceEffect.shortpause:
                        slider = avator.AvatorUI.ShortPauseSlider;
                        break;

                    case EnumVoiceEffect.longpause:
                        slider = avator.AvatorUI.LongPauseSlider;
                        break;
                }

                double p =
Convert.ToDouble(avator.VoiceEffects[item.Key].value);

                if (slider != null) slider.EmulateChangeValue(p);
            }
```

```
        }

        private decimal GetSliderValue(int avatorIndex, EnumVoiceEffect
ef)
        {
            decimal ans = 0.00m;
            WPFSlider slider = null;
            VOICEROID2.AvatorParam avator = AvatorParams[avatorIndex]
as VOICEROID2.AvatorParam;

            AvatorSelect(avatorIndex);
            TuneTab.EmulateChangeSelectedIndex(1);

            switch (ef)
            {
                case EnumVoiceEffect.volume:
                    slider = avator.AvatorUI.VolumeSlider;
                    break;

                case EnumVoiceEffect.speed:
                    slider = avator.AvatorUI.SpeedSlider;
                    break;

                case EnumVoiceEffect.pitch:
                    slider = avator.AvatorUI.PitchSlider;
                    break;

                case EnumVoiceEffect.intonation:
                    slider = avator.AvatorUI.IntonationSlider;
                    break;

                case EnumVoiceEffect.shortpause:
                    slider = avator.AvatorUI.ShortPauseSlider;
                    break;

                case EnumVoiceEffect.longpause:
                    slider = avator.AvatorUI.LongPauseSlider;
                    break;
            }

            ans = Convert.ToDecimal(slider.Value);

            return ans;
        }

        private decimal GetSliderValue(int avatorIndex, string emotion)
        {
            VOICEROID2.AvatorParam avator = AvatorParams[avatorIndex]
as VOICEROID2.AvatorParam;
```

```
            AvatorSelect(avatorIndex);
            TuneTab.EmulateChangeSelectedIndex(1);

            if
(!avator.AvatorUI.EmotionSliderIndexs.ContainsKey(emotion))
            {
                ThrowException("Effect Slider not found");
            }

            WPFSlider slider =
avator.AvatorUI.EmotionSliders[avator.AvatorUI.EmotionSliderIndexs[emot
ion]];

            return Convert.ToDecimal(slider.Value);
        }

    }
}
```

VOICEROID+EX用制御コード

ScDeviceDriver.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Codeer.Friendly;
using Codeer.Friendly.Windows;
using Codeer.Friendly.Windows.Grasp;
using Codeer.Friendly.Windows.NativeStandardControls;
using Ong.Friendly.FormsStandardControls;
using ScDriver;

namespace ScDriver.VoiceroidEx
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private readonly string DrvName =
"VoiceroidEx.Driver@echoseika.hgotoh.jp";
        private readonly string DrvVersion = "20200430/c";
        private readonly string DrvProdName = "VOICEROID+EX";

        private readonly int CidBase = 1700;
        private readonly string[] VoiceroidExTitles =
```

```
        {
                "VOICEROID□ 京町セイカ EX",
                "VOICEROID□ 民安ともえ EX",
                "VOICEROID□ 結月ゆかり EX",
                "VOICEROID□ 東北ずん子 EX",
                "VOICEROID□ 水奈瀬コウ EX",
                "VOICEROID□ 東北きりたん EX",
                "VOICEROID□ 琴葉茜",
                "VOICEROID□ 琴葉葵",
                "VOICEROID□ 東北ずん子",
                "VOICEROID□ 鷹の爪 吉田くん EX",
                "VOICEROID□ 月読アイ EX",
                "VOICEROID□ 月読ショウタ EX",
                "音街ウナTalk Ex",
                "ギャラ子Talk",
                "ギャラ子 Talk"
        };

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
            ScDrvProdName = DrvProdName;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            foreach (VoiceroidEx.AvatorParam avator in
GetVoiceroidProcess())
            {
                AvatorParams.Add(avator.AvatorIndex, avator);

                avator.AvatorUI = new VoiceroidEx.AvatorUIParam();

                try
                {
                    avator.AvatorUI._app = new
WindowsAppFriend(avator.AvatorProcess);
                    avator.AvatorUI.uiTreeTop =
WindowControl.FromZTop(avator.AvatorUI._app);

                    // Zインデクスはコーディア様のTestAssistantで確認可能。音
声効果タブへ切り替えてUI要素を取得する。
                    dynamic VoiceroidExUiTab = new
FormsTabControl(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0,
0, 0));
                    VoiceroidExUiTab.EmulateTabSelect(2);
```

```
                    avator.AvatorUI.TalkTextBox =
avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0, 1, 1);
                    avator.AvatorUI.PlayButton = new
FormsButton(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0,
1, 0, 3));
                    avator.AvatorUI.SaveButton = new
FormsButton(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 1, 0,
1, 0, 1));
                    avator.AvatorUI.VolumeText = new
FormsTextBox(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 0,
0, 0, 0, 8));
                    avator.AvatorUI.SpeedText = new
FormsTextBox(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 0,
0, 0, 0, 9));
                    avator.AvatorUI.PitchText = new
FormsTextBox(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 0,
0, 0, 0, 10));
                    avator.AvatorUI.IntonationText = new
FormsTextBox(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0, 0,
0, 0, 0, 11));
                }
                catch (Exception e)
                {
                    ThrowException(string.Format(@"{0} {1}", e.Message,
e.StackTrace));
                }

                avator.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                {
                    { EnumVoiceEffect.volume,     new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.volume), 0.0m, 2.0m, 0.01m)},
                    { EnumVoiceEffect.speed,      new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.speed), 0.5m, 4.0m, 0.01m)},
                    { EnumVoiceEffect.pitch,      new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.pitch), 0.5m, 2.0m, 0.01m)},
                    { EnumVoiceEffect.intonation, new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.intonation), 0.0m, 2.0m, 0.01m)}
                };
                avator.VoiceEffects = new Dictionary<EnumVoiceEffect,
EffectValueInfo>
                {
                    { EnumVoiceEffect.volume,     new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.volume), 0.0m, 2.0m, 0.01m)},
                    { EnumVoiceEffect.speed,      new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
```

```
EnumVoiceEffect.speed), 0.5m, 4.0m, 0.01m)},
                    { EnumVoiceEffect.pitch,      new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.pitch), 0.5m, 2.0m, 0.01m)},
                    { EnumVoiceEffect.intonation, new
EffectValueInfo(GetSliderValue(avator.AvatorIndex,
EnumVoiceEffect.intonation), 0.0m, 2.0m, 0.01m)}
                };
                avator.VoiceEmotions_default = new Dictionary<string,
EffectValueInfo>();
                avator.VoiceEmotions = new Dictionary<string,
EffectValueInfo>();
            }

            IsAlive = AvatorParams.Count != 0;
        }

        private List<ScDriver.VoiceroidEx.AvatorParam>
GetVoiceroidProcess()
        {
            Process[] ProcessList = Process.GetProcesses();
            List<VoiceroidEx.AvatorParam> VoiceroidProcesses = new
List<VoiceroidEx.AvatorParam>();

            //for (int idx = 0; idx < VoiceroidExTitles.Length; idx++)
            //{
            //    string WinTitle1 = VoiceroidExTitles[idx];
            //    string WinTitle2 = WinTitle1 + "*";

            //    foreach (Process p in ProcessList)
            //    {
            //        if ((p.MainWindowHandle != IntPtr.Zero) &&
            //             ((p.MainWindowTitle.Equals(WinTitle1)) ||
(p.MainWindowTitle.Equals(WinTitle2))))
            //        {
            //            VoiceroidEx.AvatorParam avator = new
VoiceroidEx.AvatorParam();
            //            avator.AvatorIndex = idx;
            //            avator.AvatorName = VoiceroidExTitles[idx];
            //            avator.AvatorProcess = p;

            //            VoiceroidProcesses.Add(avator);
            //            break;
            //        }
            //    }
            //}

            int idx = 0;
            foreach(string prodTitle in VoiceroidExTitles)
```

```
            {
                string WinTitle1 = prodTitle;
                string WinTitle2 = WinTitle1 + "*";

                foreach (Process p in ProcessList)
                {
                    if ((p.MainWindowHandle != IntPtr.Zero) &&
                         ((p.MainWindowTitle.Equals(WinTitle1)) ||
(p.MainWindowTitle.Equals(WinTitle2))))
                    {
                        VoiceroidEx.AvatorParam avator = new
VoiceroidEx.AvatorParam();
                        avator.AvatorIndex = idx;
                        avator.AvatorName = prodTitle;
                        avator.AvatorProcess = p;

                        VoiceroidProcesses.Add(avator);
                        idx++;

                        break;
                    }
                }
            }

            return VoiceroidProcesses;
        }

        public void Dispose()
        {
            Dispose(true);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string talkText)
        {
            Stopwatch stopWatch = new Stopwatch();
            int avatorIdx = ConvertAvatorIndex(cid);
            VoiceroidEx.AvatorParam avator = AvatorParams[avatorIdx] as
VoiceroidEx.AvatorParam;
            avator.Semaphore.Wait();

            if (avator.AvatorUI.PlayButton == null) return 0.0;
            if (avator.AvatorUI.SaveButton == null) return 0.0;
            if (avator.AvatorUI.TalkTextBox == null) return 0.0;

            dynamic VoiceroidExUiTab = new
```

```
FormsTabControl(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0,
0, 0));
            VoiceroidExUiTab.EmulateTabSelect(2);

            ApplyEffectParameters(avatorIdx);
            ApplyEmotionParameters(avatorIdx);

            //再生中なので再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            avator.AvatorUI.TalkTextBox["Text"](talkText);
            Thread.Sleep(10);

            stopWatch.Start();

            avator.AvatorUI.PlayButton.EmulateClick();

            //再生開始を待つ(音声保存ボタンがDisableになるのを待つ)
            if (avator.AvatorUI.SaveButton.Enabled)
            {
                while (avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            //再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            stopWatch.Stop();
            avator.Semaphore.Release();

            return stopWatch.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
```

```
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        public override void PlayAsync(int cid, string talkText)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            VoiceroidEx.AvatorParam avator = AvatorParams[avatorIdx] as
VoiceroidEx.AvatorParam;

            Task.Run(() =>
            {
                avator.Semaphore.Wait();

                if (avator.AvatorUI.PlayButton == null) return;
                if (avator.AvatorUI.SaveButton == null) return;
                if (avator.AvatorUI.TalkTextBox == null) return;

                dynamic VoiceroidExUiTab = new
FormsTabControl(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0,
0, 0));
                VoiceroidExUiTab.EmulateTabSelect(2);

                ApplyEffectParameters(avatorIdx);
                ApplyEmotionParameters(avatorIdx);

                //再生中なので再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
                if (!avator.AvatorUI.SaveButton.Enabled)
                {
                    while (!avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                avator.AvatorUI.TalkTextBox["Text"](talkText);
                Thread.Sleep(10);

                avator.AvatorUI.PlayButton.EmulateClick();

                //再生開始を待つ(音声保存ボタンがDisableになるのを待つ)
                if (avator.AvatorUI.SaveButton.Enabled)
                {
                    while (avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                //再生終了を待つ(音声保存ボタンがEnableになるのを待つ)
                if (!avator.AvatorUI.SaveButton.Enabled)
                {
```

```
                    while (!avator.AvatorUI.SaveButton.Enabled)
                    {
                        Thread.Sleep(10);
                    }
                }

                avator.Semaphore.Release();
            });
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string talkText, string
saveFilename)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            VoiceroidEx.AvatorParam avator = AvatorParams[avatorIdx] as
VoiceroidEx.AvatorParam;

            if (avator.AvatorUI.PlayButton == null) return 0.0;
            if (avator.AvatorUI.SaveButton == null) return 0.0;
            if (avator.AvatorUI.TalkTextBox == null) return 0.0;

            dynamic VoiceroidExUiTab = new
FormsTabControl(avator.AvatorUI.uiTreeTop.IdentifyFromZIndex(2, 0, 0,
0, 0));
            VoiceroidExUiTab.EmulateTabSelect(2);

            ApplyEffectParameters(avatorIdx);
            ApplyEmotionParameters(avatorIdx);

            if (!avator.AvatorUI.SaveButton.Enabled)
            {
                while (!avator.AvatorUI.SaveButton.Enabled)
                {
                    Thread.Sleep(10);
                }
            }

            avator.AvatorUI.TalkTextBox["Text"](talkText);
            Thread.Sleep(10);

            avator.AvatorUI.SaveButton.EmulateClick(new Async());
```

```
            bool finish_savefileSetup = false;
            while (finish_savefileSetup == false)
            {
                //名前を付けて保存 ダイアログで名前を設定
                var FileDlgs =
WindowControl.GetFromWindowText(avator.AvatorUI._app, "音声ファイルの保
存");
                try
                {
                    if ((FileDlgs.Length != 0) &&
(FileDlgs[0].WindowClassName == "#32770"))
                    {
                        // https://github.com/mikoto2000/TTSController
UI特定の記述を参照
                        NativeButton OkButton = new
NativeButton(FileDlgs[0].IdentifyFromDialogId(1));
                        NativeEdit SaveNameText = new
NativeEdit(FileDlgs[0].IdentifyFromZIndex(11, 0, 4, 0, 0));

                        //ファイル名を設定
                        SaveNameText.EmulateChangeText(saveFilename);
                        Thread.Sleep(100);

                        //OKボタンを押す
                        OkButton.EmulateClick(new Async());
                        finish_savefileSetup = true;
                    }
                }
                catch (Exception)
                {
                    //
                }

                Thread.Sleep(10);
            }

            return 0.0;
        }

        /// <summary>
        ///感情パラメタをデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEmotion(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;

            foreach (var emotion in avator.VoiceEmotions_default)
            {
```

```
                avator.VoiceEmotions[emotion.Key].value =
emotion.Value.value;
            }

            ApplyEmotionParameters(avatorIdx);
        }

        /// <summary>
        /// 音声効果をデフォルト値に戻す
        /// </summary>
        /// <param name="cid">話者CID</param>
        public override void ResetVoiceEffect(int cid)
        {
            int avatorIdx = ConvertAvatorIndex(cid);
            AvatorParam avator = AvatorParams[avatorIdx] as
AvatorParam;

            foreach (var effect in avator.VoiceEffects_default)
            {
                avator.VoiceEffects[effect.Key].value =
effect.Value.value;
            }

            ApplyEffectParameters(avatorIdx);
        }

        public override void Dispose(bool disposing)
        {
            if (Disposed) return;

            if (disposing)
            {
                foreach(var item in AvatorParams)
                {
                    ResetVoiceEffect(item.Key + CidBase);
                    ResetVoiceEmotion(item.Key + CidBase);
                    (item.Value as
VoiceroidEx.AvatorParam).AvatorUI._app.Dispose();
                }

                AvatorParams.Clear();

                //throw new NotImplementedException();
            }

            Disposed = true;
        }

        private void ApplyEmotionParameters(int avatorIndex)
```

```
        {
            //
        }

        private void ApplyEffectParameters(int avatorIdx)
        {
            FormsTextBox TargetTextBox = null;
            double value = 0.00;
            VoiceroidEx.AvatorParam avator = AvatorParams[avatorIdx] as
VoiceroidEx.AvatorParam;

            foreach (var effect in avator.VoiceEffects)
            {
                switch (effect.Key)
                {
                    case EnumVoiceEffect.volume:
                        TargetTextBox = avator.AvatorUI.VolumeText;
                        break;

                    case EnumVoiceEffect.speed:
                        TargetTextBox = avator.AvatorUI.SpeedText;
                        break;

                    case EnumVoiceEffect.pitch:
                        TargetTextBox = avator.AvatorUI.PitchText;
                        break;

                    case EnumVoiceEffect.intonation:
                        TargetTextBox = avator.AvatorUI.IntonationText;
                        break;
                }

                value =
Convert.ToDouble(avator.VoiceEffects[effect.Key].value);

                if (TargetTextBox != null)
                {
TargetTextBox.EmulateChangeText(string.Format("{0:0.00}", value));
                }
            }
        }

        private decimal GetSliderValue(int avatorIdx, EnumVoiceEffect
effect)
        {
            decimal value = 0.00m;
            FormsTextBox TargetTextBox = null;
            VoiceroidEx.AvatorParam avator = AvatorParams[avatorIdx] as
VoiceroidEx.AvatorParam;

            switch (effect)
```

```
            {
                case EnumVoiceEffect.volume:
                    TargetTextBox = avator.AvatorUI.VolumeText;
                    break;

                case EnumVoiceEffect.speed:
                    TargetTextBox = avator.AvatorUI.SpeedText;
                    break;

                case EnumVoiceEffect.pitch:
                    TargetTextBox = avator.AvatorUI.PitchText;
                    break;

                case EnumVoiceEffect.intonation:
                    TargetTextBox = avator.AvatorUI.IntonationText;
                    break;
            }

            if (TargetTextBox != null)
            {
                value = Convert.ToDecimal(TargetTextBox.Text);
            }

            return value;
        }

    }

}
```

CeVIO制御用コード

32bit/64bit 共用です。

プロキシクラス

CeVIOの場合、プロキシークラスを用意して遅延バインディングしています。こうしないと解放ができなくなるので。

CevioProxy.cs

```
using System;
using System.Collections.Generic;

namespace ScDriver.CeVio
{
```

```
    public class CevioProxy : MarshalByRefObject
    {
        dynamic CevioTalkerProd;
        double Timeout = 3 * 60 * 1000;

        public string[] AvailableCasts
        {
            get
            {
                List<string> x = new List<string>();

                dynamic ac = CevioTalkerProd.AvailableCasts;
                for (int i = 0; i < ac.Length; i++)
                {
                    x.Add(ac[i]);
                }
                return x.ToArray();
            }
        }

        public string Cast
        {
            get
            {
                return CevioTalkerProd.Cast;
            }
            set
            {
                CevioTalkerProd.Cast = value;
            }
        }

        public uint Volume
        {
            get
            {
                return CevioTalkerProd.Volume;
            }
            set
            {
                CevioTalkerProd.Volume = value;
            }
        }

        public uint Speed
        {
            get
            {
                return CevioTalkerProd.Speed;
            }
            set
```

```
            {
                CevioTalkerProd.Speed = value;
            }
        }

        public uint Tone
        {
            get
            {
                return CevioTalkerProd.Tone;
            }
            set
            {
                CevioTalkerProd.Tone = value;
            }
        }

        public uint Alpha
        {
            get
            {
                return CevioTalkerProd.Alpha;
            }
            set
            {
                CevioTalkerProd.Alpha = value;
            }
        }

        public uint ToneScale
        {
            get
            {
                return CevioTalkerProd.ToneScale;
            }
            set
            {
                CevioTalkerProd.ToneScale = value;
            }
        }

        public Dictionary<string, uint> Components
        {
            get
            {
                var emo = CevioTalkerProd.Components;
                Dictionary<string, uint> param = new Dictionary<string,
uint>();
```

```
                for (int i = 0; i < emo.Length; i++)
                {
                    param.Add(emo[i].Name, emo[i].Value);
                }
                return param;
            }
        }

        public void SetComponent(string emoName, uint emoValue)
        {
            CevioTalkerProd.Components[emoName].Value = emoValue;
        }

        public string GetComponent(string emoName)
        {
            return CevioTalkerProd.Components[emoName].Value;
        }

        public void Speak(string text)
        {
            dynamic status = CevioTalkerProd.Speak(text);
//SpeakingState status = cevioTalker.Speak(TalkText);

            status.Wait(Timeout);
            CevioTalkerProd.Stop();
        }

        public void Save(string text, string WavFilePath)
        {
            CevioTalkerProd.OutputWaveToFile(text, WavFilePath);
        }

        public override object InitializeLifetimeService()
        {
            return null;
        }

        public CevioProxy()
        {
            try
            {
                CevioTalkerProd =
Activator.CreateInstance(Type.GetTypeFromProgID("CeVIO.Talk.RemoteServi
ce.Talker")); //遅延バインドで使用しますCeVIO AI だと
"CeVIO.Talk.RemoteService2.Talker2"
            }
            catch(Exception)
            {
                CevioTalkerProd = null;
            }
        }
```

```
    }
}
```

AssistantSeika用コード

ScDeviceDriver.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Reflection;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;

namespace ScDriver.CeVio
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private const string DrvName =
"CeVIO.Driver@echoseika.hgotoh.jp";
        private const string DrvVersion = "20210327/c";
        private const string DrvProdName = "CeVIO";
        private const int DrvTextMaxLength = 100;
        private const int CidBase = 3000;
        private const int MaxAvators = 100;
        private const int SplitLine = 50;

        public SemaphoreSlim Semaphore = new SemaphoreSlim(1, 1);

        private CeVio.CevioProxy CevioTalkProxy = null;

        class titles
        {
            public int PresetType { get; private set; }
            public string Avator { get; private set; }
            public int FixedCid { get; private set; }
            public int AvatorIndex { get; private set; }

            public titles(int priset, string avator, int cid)
            {
                PresetType = priset;
                Avator = avator;
                FixedCid = cid;
```

```
                AvatorIndex = 0;
            }
        }

        private readonly titles[] CeVIO2Names =
        {
            new titles(0, "さとうささら",             CidBase + 1),
            new titles(0, "すずきつづみ",             CidBase + 2),
            new titles(0, "タカハシ",                  CidBase + 3),
            new titles(0, "ONE",                       CidBase + 4),
            new titles(0, "IA",                        CidBase + 5),
        };

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
            ScDrvProdName = DrvProdName;
            ScDrvTextMaxLength = DrvTextMaxLength;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            if(GetCeVIOEditorProcess())
            {
                titles ent = null;

                //認識対象外、もしくはユーザ定義プリセットに割り当てるcidのカウンター
                int cbaseCounter = 1 + CeVIO2Names.Select(v => v.FixedCid).Max();
                if (cbaseCounter < (CidBase + SplitLine)) cbaseCounter = CidBase + SplitLine;

                try
                {
                    AppDomain appDomain = AppDomain.CreateDomain("SeikaCenterCevioPlugin");
                    CevioTalkProxy = (CeVio.CevioProxy)appDomain.CreateInstanceAndUnwrap(Assembly.GetExecutingAssembly().FullName, typeof(CeVio.CevioProxy).FullName);

                    string[] talkers = CevioTalkProxy.AvailableCasts;

                    if (talkers.Length != 0)
                    {
                        for (int avatorIndex = 0; avatorIndex < talkers.Length; avatorIndex++)
                        {
                            if (cbaseCounter >= (CidBase + MaxAvators))
```

```
break;

                                CevioTalkProxy.Cast = talkers[avatorIndex];

                                CeVio.AvatorParam avator = new
CeVio.AvatorParam();

                                avator.AvatorIndex = avatorIndex;
                                avator.AvatorName = talkers[avatorIndex];

                                //認識している話者名ならば固定のcidを設定
                                try
                                {
                                    ent = CeVIO2Names.Where(v => v.Avator
== avator.AvatorName).First();
                                    avator.FixedCid = ent.FixedCid;
                                }
                                catch (Exception)
                                {
                                    avator.FixedCid = cbaseCounter;
                                    cbaseCounter++;
                                }

                                //このタイミングで一旦登録する
                                AvatorParams.Add(avator.FixedCid, avator);

                                avator.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                                {
                                    {EnumVoiceEffect.volume,     new
EffectValueInfo( (decimal)CevioTalkProxy.Volume, 0.0m, 100.0m, 1.00m)},
                                    {EnumVoiceEffect.speed,      new
EffectValueInfo( (decimal)CevioTalkProxy.Speed, 0.0m, 100.0m, 1.00m)},
                                    {EnumVoiceEffect.pitch,      new
EffectValueInfo( (decimal)CevioTalkProxy.Tone, 0.0m, 100.0m, 1.00m)},
                                    {EnumVoiceEffect.alpha,      new
EffectValueInfo( (decimal)CevioTalkProxy.Alpha, 0.0m, 100.0m, 1.00m)},
                                    {EnumVoiceEffect.intonation, new
EffectValueInfo( (decimal)CevioTalkProxy.ToneScale, 0.0m, 100.0m,
1.00m)}
                                };
                                avator.VoiceEffects = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                                {
                                    {EnumVoiceEffect.volume,     new
EffectValueInfo( (decimal)CevioTalkProxy.Volume, 0.0m, 100.0m, 1.00m)},
                                    {EnumVoiceEffect.speed,      new
EffectValueInfo( (decimal)CevioTalkProxy.Speed, 0.0m, 100.0m, 1.00m)},
                                    {EnumVoiceEffect.pitch,      new
```

```
EffectValueInfo( (decimal)CevioTalkProxy.Tone, 0.0m, 100.0m, 1.00m)},
                                {EnumVoiceEffect.alpha,      new
EffectValueInfo( (decimal)CevioTalkProxy.Alpha, 0.0m, 100.0m, 1.00m)},
                                {EnumVoiceEffect.intonation, new
EffectValueInfo( (decimal)CevioTalkProxy.ToneScale, 0.0m, 100.0m,
1.00m)}
                            };

                            avator.VoiceEmotions_default = new
Dictionary<string, EffectValueInfo>();
                            avator.VoiceEmotions = new
Dictionary<string, EffectValueInfo>();
                            Dictionary<string, uint> emoparams =
CevioTalkProxy.Components;
                            if (0 < emoparams.Count)
                            {
                                foreach (KeyValuePair<string, uint>
emotion in emoparams)
                                {
avator.VoiceEmotions_default.Add(emotion.Key, new
EffectValueInfo((decimal)(emotion.Value), 0.00m, 100.0m, 1.00m));
avator.VoiceEmotions.Add(emotion.Key, new
EffectValueInfo((decimal)(emotion.Value), 0.00m, 100.0m, 1.00m));
                                }
                            }

                        }
                    }
                }
                catch (Exception)
                {
                    AvatorParams.Clear();
                    // ThrowException(string.Format(@"{0} {1}",
e.Message, e.StackTrace));
                }
            }
            else
            {
                AvatorParams.Clear();
            }

            IsAlive = AvatorParams.Count != 0;

            // cidエイリアス用
            if (IsAlive)
            {
                CeVio.AvatorParam avator = new CeVio.AvatorParam();
                int firsFindtCid = AvatorParams.First().Value.FixedCid;

                avator.FixedCid              = CidBase;
                avator.AvatorIndex           = AvatorParams.Count;
```

```
                avator.AvatorName            =
AvatorParams[firsFindtCid].AvatorName;
                avator.AliasCode           = true;
                avator.VoiceEffects          =
(AvatorParams[firsFindtCid] as CeVio.AvatorParam).VoiceEffects;
                avator.VoiceEmotions         =
(AvatorParams[firsFindtCid] as CeVio.AvatorParam).VoiceEmotions;
                avator.VoiceEffects_default  =
(AvatorParams[firsFindtCid] as CeVio.AvatorParam).VoiceEffects_default;
                avator.VoiceEmotions_default =
(AvatorParams[firsFindtCid] as
CeVio.AvatorParam).VoiceEmotions_default;

                AvatorParams.Add(avator.FixedCid, avator);
            }

        }

        public override void Dispose(bool disposing)
        {
            if (Disposed) return;

            if (disposing)
            {
                AvatorParams.Clear();
            }

            Disposed = true;
        }

        public void Dispose()
        {
            Dispose(true);
        }

        /// <summary>
        /// CeVIO起動の確認
        /// </summary>
        private bool GetCeVIOEditorProcess()
        {
            string winTitle1 = @"^.+ - CeVIO CS6$";
            string winTitle2 = @"^.+ - CeVIO CS7$";
            string runProcName = @"CeVIO Creative Studio";

            int RetryCount = 3;
            int RetryWaitms = 500;
            bool p = false;

            for (int i = 0; i < 3; i++)
```

```
            {
                Process[] ps = Process.GetProcesses();

                foreach (Process pitem in ps)
                {
                    if ((pitem.MainWindowHandle != IntPtr.Zero) &&
(pitem.ProcessName == runProcName))
                    {
                        if (Environment.Is64BitProcess)
                        {
                            if (Regex.IsMatch(pitem.MainWindowTitle,
winTitle2))
                            {
                                p = true;
                                break;
                            }
                        }
                        else
                        {
                            if (Regex.IsMatch(pitem.MainWindowTitle,
winTitle1))
                            {
                                p = true;
                                break;
                            }
                        }
                    }
                }

                if (p) break;
                if (i < (RetryCount - 1)) Thread.Sleep(RetryWaitms);
            }

            return p;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string talkText)
        {
            Semaphore.Wait();

            Stopwatch sw = new Stopwatch();

            int avatorIndex = ConvertAvatorIndex(cid);

            AvatorSelect(avatorIndex);
```

```
            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            sw.Start();

            try
            {
                int len = talkText.Length > ScDriverTextMaxLength ?
ScDriverTextMaxLength : talkText.Length;
                CevioTalkProxy.Speak(talkText.Substring(0, len));
            }
            catch (Exception)
            {
                //
            }

            sw.Stop();
            Semaphore.Release();

            return sw.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string[] talkTexts)
        {
            Semaphore.Wait();

            Stopwatch sw = new Stopwatch();

            int avatorIndex = ConvertAvatorIndex(cid);

            AvatorSelect(avatorIndex);
            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            sw.Start();

            try
            {
                foreach(string txt in talkTexts)
                {
                    int len = txt.Length > ScDriverTextMaxLength ?
ScDriverTextMaxLength : txt.Length;
                    CevioTalkProxy.Speak(txt.Substring(0, len));
```

```
                }
            }
            catch (Exception)
            {
                //MessageBox.Show(String.Format("{0}:{1}", e.Message,
e.StackTrace));
            }

            sw.Stop();
            Semaphore.Release();

            return sw.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        public override void PlayAsync(int cid, string talkText)
        {
            Task.Run(() =>
            {
                Semaphore.Wait();

                int avatorIndex = ConvertAvatorIndex(cid);

                AvatorSelect(avatorIndex);
                ApplyEffectParameters(avatorIndex);
                ApplyEmotionParameters(avatorIndex);

                try
                {
                    int len = talkText.Length > ScDriverTextMaxLength ?
ScDriverTextMaxLength : talkText.Length;
                    CevioTalkProxy.Speak(talkText.Substring(0, len));
                }
                catch (Exception)
                {
                    //
                }

                Semaphore.Release();
            });
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
```

```
        public override void PlayAsync(int cid, string[] talkTexts)
        {
            Task.Run(() =>
            {
                Semaphore.Wait();

                int avatorIndex = ConvertAvatorIndex(cid);

                AvatorSelect(avatorIndex);
                ApplyEffectParameters(avatorIndex);
                ApplyEmotionParameters(avatorIndex);

                try
                {
                    foreach (string txt in talkTexts)
                    {
                        int len = txt.Length > ScDriverTextMaxLength ?
ScDriverTextMaxLength : txt.Length;
                        CevioTalkProxy.Speak(txt.Substring(0, len));
                    }
                }
                catch (Exception)
                {
                    //
                }

                Semaphore.Release();
            });
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string talkText, string
saveFilename)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            AvatorSelect(avatorIndex);

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            try
            {
```

```
                int len = talkText.Length > ScDriverTextMaxLength ?
ScDriverTextMaxLength : talkText.Length;
                CevioTalkProxy.Save(talkText.Substring(0, len),
saveFilename);
            }
            catch (Exception)
            {
                //
            }

            return 0.0;
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkTexts">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string[] talkTexts, string
saveFilename)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            AvatorSelect(avatorIndex);

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            try
            {
                string s = String.Join("", talkTexts);
                int len = s.Length > ScDriverTextMaxLength ?
ScDriverTextMaxLength : s.Length;
                CevioTalkProxy.Save(s.Substring(0, len), saveFilename);
            }
            catch (Exception)
            {
                //
            }

            return 0.0;
        }

        public override void ResetVoiceEmotion(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            foreach (KeyValuePair<string, EffectValueInfo> item in
AvatorParams[avatorIndex].VoiceEmotions_default)
```

```
            {
                AvatorParams[avatorIndex].VoiceEmotions[item.Key].value
= item.Value.value;
            }

            ApplyEmotionParameters(avatorIndex);
        }

        public override void ResetVoiceEffect(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            foreach (var effect in
AvatorParams[avatorIndex].VoiceEffects_default)
            {
AvatorParams[avatorIndex].VoiceEffects[effect.Key].value =
effect.Value.value;
            }

            ApplyEffectParameters(avatorIndex);
        }

        public override void SetAvator(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);
            AvatorSelect(avatorIndex);
        }

        private void AvatorSelect(int avatorIndex)
        {
            CevioTalkProxy.Cast = AvatorParams[avatorIndex].AvatorName;
        }

        private void ApplyEmotionParameters(int avatorIndex)
        {
            foreach (KeyValuePair<string, EffectValueInfo> item in
AvatorParams[avatorIndex].VoiceEmotions)
            {
                CevioTalkProxy.SetComponent(item.Key,
(uint)(item.Value.value));
            }
        }

        private void ApplyEffectParameters(int avatorIndex)
        {
            foreach (KeyValuePair<EnumVoiceEffect, EffectValueInfo>
item in AvatorParams[avatorIndex].VoiceEffects)
            {
                switch (item.Key)
```

```
                {
                    case EnumVoiceEffect.volume:
                        CevioTalkProxy.Volume =
(uint)(item.Value.value);
                        break;

                    case EnumVoiceEffect.speed:
                        CevioTalkProxy.Speed =
(uint)(item.Value.value);
                        break;

                    case EnumVoiceEffect.pitch:
                        CevioTalkProxy.Tone = (uint)(item.Value.value);
                        break;

                    case EnumVoiceEffect.alpha:
                        CevioTalkProxy.Alpha =
(uint)(item.Value.value);
                        break;

                    case EnumVoiceEffect.intonation:
                        CevioTalkProxy.ToneScale =
(uint)(item.Value.value);
                        break;
                }
            }
        }


    }
}
```

棒読みちゃん制御コード

共有オブジェクト

棒読みちゃんで提供する.NET Remotingのクラス定義

BouyomiChanRemoting.cs

```
using System;

namespace FNF.Utility
{
    /// <summary>
    /// .NET Remotingのためのクラス。（本クラスの内容を変更してしまうと通信できなくなってしまいます）
    /// See BouyomiChan 0.1.10.0 BouyomiChan\SampleSrc\IpcClientChannel
```

```
で読み上げ指示を送る(ローカル専用□.NET専用)\Src\BouyomiChanSample
    /// </summary>
    public class BouyomiChanRemoting : MarshalByRefObject
    {
        public void AddTalkTask(string sTalkText) { }
        public void AddTalkTask(string sTalkText, int iSpeed, int
iVolume, int vType) { }
        public void AddTalkTask(string sTalkText, int iSpeed, int
iTone, int iVolume, int vType) { }
        public int AddTalkTask2(string sTalkText) { throw null; }
        public int AddTalkTask2(string sTalkText, int iSpeed, int
iTone, int iVolume, int vType) { throw null; }
        public void ClearTalkTasks() { }
        public void SkipTalkTask() { }

        public int TalkTaskCount { get { throw null; } }
        public int NowTaskId { get { throw null; } }
        public bool NowPlaying { get { throw null; } }
        public bool Pause { get { throw null; } set { } }
    }
}
```

プロキシクラス

BouyomiChanProxy.cs

```
//////////////////////////////////////////////////////////////////////
//////////////////
// BouyomiChan- Ver0.1.10.0 の配布アーカイブにあるソース
BouyomiChanClient.cs に手を入れています。
// GUIを見ているわけではなく、話者のインデクスは決め打ちです
//////////////////////////////////////////////////////////////////////
//////////////////
using System;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Ipc;
using System.Threading;

namespace FNF.Utility
{
    /// <summary>
    /// 声の種類。(0:デフォルト□1□8:AquesTalk□10001□□SAPI5)
    /// </summary>
    public enum VoiceType { Default = 0, Female1 = 1, Female2 = 2,
Male1 = 3, Male2 = 4, Imd1 = 5, Robot1 = 6, Machine1 = 7, Machine2 = 8
}
```

```
    /// <summary>
    /// 棒読みちゃんへ接続するためのクラス。
    /// </summary>
    public class BouyomiChanProxy : IDisposable
    {
        protected IpcClientChannel ClientChannel;
        protected BouyomiChanRemoting RemotingObject;
        public bool IsActive { get; private set; }

        /// <summary>
        /// オブジェクト生成。
        /// 利用後にはDispose()で開放してください。
        /// </summary>
        public BouyomiChanProxy()
        {
            try
            {
                ClientChannel = new IpcClientChannel("hogehoge", null);
//チャンネル名は何でもいい
                ChannelServices.RegisterChannel(ClientChannel, false);
                RemotingObject =
(BouyomiChanRemoting)Activator.GetObject(typeof(BouyomiChanRemoting),
"ipc://BouyomiChan/Remoting");

                bool f = RemotingObject.NowPlaying; //通信可能かどうかの確
認
                Thread.Sleep(10);

                IsActive = true;
            }
            catch (Exception ebo)
            {
                Console.WriteLine("ebo:{0},{1}", ebo.Message,
ebo.StackTrace);
                IsActive = false;
            }
        }

        /// <summary>
        /// ファイナライザ（Dispose Finalizeパターン実装）
        /// </summary>
        ~BouyomiChanProxy()
        {
            if (ClientChannel != null)
            {
                ChannelServices.UnregisterChannel(ClientChannel);
                ClientChannel = null;
            }
            Dispose();
        }
```

```
        /// <summary>
        /// オブジェクト開放。
        /// </summary>
        public void Dispose()
        {
            if (ClientChannel != null)
            {
                ChannelServices.UnregisterChannel(ClientChannel);
                ClientChannel = null;
            }
            GC.SuppressFinalize(this);
        }

        /// <summary>
        /// 棒読みちゃんに音声合成タスクを追加します。
        /// </summary>
        /// <param name="sTalkText">喋らせたい文章</param>
        public void AddTalkTask(string sTalkText)
        {
            RemotingObject.AddTalkTask(sTalkText);
        }

        /// <summary>
        /// 棒読みちゃんに音声合成タスクを追加します。(音程指定無し版。以前のバージョンとの互換性の為に残しています。)
        /// </summary>
        /// <param name="sTalkText">喋らせたい文章</param>
        /// <param name="iSpeed"   >再生。(-1で棒読みちゃん側の画面で選んでいる速度)</param>
        /// <param name="iVolume"  >音量。(-1で棒読みちゃん側の画面で選んでいる音量)</param>
        /// <param name="vType"    >声の種類(Defaultで棒読みちゃん側の画面で選んでいる声)</param>
        public void AddTalkTask(string sTalkText, int iSpeed, int iVolume, VoiceType vType)
        {
            RemotingObject.AddTalkTask(sTalkText, iSpeed, iVolume, (int)vType);
        }

        /// <summary>
        /// 棒読みちゃんに音声合成タスクを追加します。
        /// </summary>
        /// <param name="sTalkText">喋らせたい文章</param>
        /// <param name="iSpeed"   >速度。(-1で棒読みちゃん側の画面で選んでいる速度)</param>
        /// <param name="iTone"    >音程。(-1で棒読みちゃん側の画面で選んでいる音程)</param>
        /// <param name="iVolume"  >音量。(-1で棒読みちゃん側の画面で選んでいる音量)</param>
```

```
        /// <param name="vType"    >声の種類(Defaultで棒読みちゃん側の画面で選んでいる声)</param>
        public void AddTalkTask(string sTalkText, int iSpeed, int iTone, int iVolume, VoiceType vType)
        {
            RemotingObject.AddTalkTask(sTalkText, iSpeed, iTone, iVolume, (int)vType);
        }

        /// <summary>
        /// 棒読みちゃんに音声合成タスクを追加します。読み上げタスクIDを返します。
        /// </summary>
        /// <param name="sTalkText">喋らせたい文章</param>
        /// <returns>読み上げタスクID</returns>
        public int AddTalkTask2(string sTalkText)
        {
            return RemotingObject.AddTalkTask2(sTalkText);
        }

        /// <summary>
        /// 棒読みちゃんに音声合成タスクを追加します。読み上げタスクIDを返します。
        /// </summary>
        /// <param name="sTalkText">喋らせたい文章</param>
        /// <param name="iSpeed"   >速度。(-1で棒読みちゃん側の画面で選んでいる速度)</param>
        /// <param name="iTone"    >音程。(-1で棒読みちゃん側の画面で選んでいる音程)</param>
        /// <param name="iVolume"  >音量。(-1で棒読みちゃん側の画面で選んでいる音量)</param>
        /// <param name="vType"    >声の種類(Defaultで棒読みちゃん側の画面で選んでいる声)</param>
        /// <returns>読み上げタスクID</returns>
        public int AddTalkTask2(string sTalkText, int iSpeed, int iTone, int iVolume, VoiceType vType)
        {
            return RemotingObject.AddTalkTask2(sTalkText, iSpeed, iTone, iVolume, (int)vType);
        }

        /// <summary>
        /// 棒読みちゃんの残りのタスクを全て消去します。
        /// </summary>
        public void ClearTalkTasks()
        {
            RemotingObject.ClearTalkTasks();
        }

        /// <summary>
        /// 棒読みちゃんの現在のタスクを中止して次の行へ移ります。
        /// </summary>
        public void SkipTalkTask()
```

```
        {
            RemotingObject.SkipTalkTask();
        }

        /// <summary>
        /// 棒読みちゃんの現在のタスク数（再生待ち行数）を取得します。
        /// </summary>
        public int TalkTaskCount
        {
            get { return RemotingObject.TalkTaskCount; }
        }

        /// <summary>
        /// 棒読みちゃんの現在再生中のタスクIDを取得します。
        /// </summary>
        public int NowTaskId
        {
            get { return RemotingObject.NowTaskId; }
        }

        /// <summary>
        /// 棒読みちゃんが現在、音声を再生している最中かどうかを取得します。
        /// </summary>
        public bool NowPlaying
        {
            get { return RemotingObject.NowPlaying; }
        }

        /// <summary>
        /// 棒読みちゃんが一時停止中かどうかを取得・設定します。
        /// ※現在の行の再生が終了するまで停止しません。
        /// </summary>
        public bool Pause
        {
            get { return RemotingObject.Pause; }
            set { RemotingObject.Pause = value; }
        }
    }
}
```

AssistantSeika用コード

ScDeviceDriver.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Diagnostics;
using System.Threading;
using System.Threading.Tasks;
using FNF.Utility;

namespace ScDriver.BouyomiChan
{
    public class ScDeviceDriver : ScBaseDriver, IScBaseDriver
    {
        private readonly string DrvName =
"BouyomiChan.Driver@echoseika.hgotoh.jp";
        private readonly string DrvVersion = "20200430/c";
        private readonly string DrvProdName = "BOUYOMICHAN";
        private readonly int CidBase = 4000;

        public SemaphoreSlim Semaphore = new SemaphoreSlim(1, 1);
        private BouyomiChanProxy BouyomiTalkProxy = null;
        private string[] AvatorNames = { "女性1", "女性2", "男性1", "男性
2", "中性", "ロボット", "機械1", "機械2" };
        private FNF.Utility.VoiceType VT;

        public ScDeviceDriver()
        {
            ScDrvName = DrvName;
            ScDrvVersion = DrvVersion;
            ScDrvProdName = DrvProdName;
            CidBaseIndex = CidBase;
            AvatorParams = new Dictionary<int, ScDriver.AvatorParam>();

            IsAlive = false;

            try
            {
                BouyomiTalkProxy = new BouyomiChanProxy();

                if (BouyomiTalkProxy.IsActive)
                {
                    for (int i = 0; i < AvatorNames.Length; i++)
                    {
                        AvatorParam item = new AvatorParam();

                        item.AvatorIndex = i;
                        item.AvatorName = AvatorNames[i];

                        item.VoiceEffects_default = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            {EnumVoiceEffect.volume,     new
EffectValueInfo( 100m, 0.0m, 100.0m, 1.00m)},
                            {EnumVoiceEffect.speed,      new
EffectValueInfo( 100m, 50.0m, 200.0m, 1.00m)},
```

```
                            {EnumVoiceEffect.pitch,      new
EffectValueInfo( 100m, 50.0m, 200.0m, 1.00m)}
                        };
                        item.VoiceEffects = new
Dictionary<EnumVoiceEffect, EffectValueInfo>
                        {
                            {EnumVoiceEffect.volume,     new
EffectValueInfo( 100m, 0.0m, 100.0m, 1.00m)},
                            {EnumVoiceEffect.speed,      new
EffectValueInfo( 100m, 50.0m, 200.0m, 1.00m)},
                            {EnumVoiceEffect.pitch,      new
EffectValueInfo( 100m, 50.0m, 200.0m, 1.00m)}
                        };

                        item.VoiceEmotions_default = new
Dictionary<string, EffectValueInfo>();
                        item.VoiceEmotions = new Dictionary<string,
EffectValueInfo>();

                        AvatorParams.Add(i, item);
                    }
                }
            }
            catch (Exception)
            {
                // ThrowException(string.Format(@"{0} {1}", e.Message,
e.StackTrace));
            }

            IsAlive = AvatorParams.Count != 0;
        }

        ~ScDeviceDriver()
        {
            BouyomiTalkProxy?.Dispose();
            Dispose(false);
        }

        public override void Dispose(bool disposing)
        {
            if (Disposed) return;

            if (disposing)
            {
                AvatorParams.Clear();
                BouyomiTalkProxy?.Dispose();
                GC.SuppressFinalize(this);
            }
```

```
            Disposed = true;
        }

        public void Dispose()
        {
            Dispose(true);
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <returns>発声にかかった時間（ミリ秒）</returns>
        public override double Play(int cid, string talkText)
        {
            Semaphore.Wait();

            Stopwatch sw = new Stopwatch();

            int avatorIndex = ConvertAvatorIndex(cid);
            int speed =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
speed].value);
            int tone =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
pitch].value);
            int volume =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
volume].value);

            AvatorSelect(avatorIndex);
            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            sw.Start();

            try
            {
                //再生中なら待つ
                while (BouyomiTalkProxy.NowPlaying == true)
                {
                    Thread.Sleep(10);
                }

                BouyomiTalkProxy.AddTalkTask2(talkText, speed, tone,
volume, VT);

                //再生中フラグが立つまで待つ
                while (BouyomiTalkProxy.NowPlaying == false)
                {
```

```
                    Thread.Sleep(10);
                }

                //再生終了まで待つ
                while (BouyomiTalkProxy.NowPlaying == true)
                {
                    Thread.Sleep(10);
                }

            }
            catch (Exception)
            {
                //
            }

            sw.Stop();
            Semaphore.Release();

            return sw.ElapsedMilliseconds;
        }

        /// <summary>
        ///指定話者で指定テキストで発声
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        public override void PlayAsync(int cid, string talkText)
        {
            Task.Run(() =>
            {
                Semaphore.Wait();

                int avatorIndex = ConvertAvatorIndex(cid);
                int speed =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
speed].value);
                int tone =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
pitch].value);
                int volume =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
volume].value);

                AvatorSelect(avatorIndex);
                ApplyEffectParameters(avatorIndex);
                ApplyEmotionParameters(avatorIndex);

                try
                {
```

```
                    //再生中なら待つ
                    while (BouyomiTalkProxy.NowPlaying == true)
                    {
                        Thread.Sleep(10);
                    }

                    BouyomiTalkProxy.AddTalkTask2(talkText, speed,
tone, volume, VT);

                    //再生中フラグが立つまで待つ
                    while (BouyomiTalkProxy.NowPlaying == false)
                    {
                        Thread.Sleep(10);
                    }

                    //再生終了まで待つ
                    while (BouyomiTalkProxy.NowPlaying == true)
                    {
                        Thread.Sleep(10);
                    }
                }
                catch (Exception)
                {
                    //
                }

                Semaphore.Release();
            });
        }

        /// <summary>
        ///指定話者で指定テキストで発声した結果をファイルに保存
        /// </summary>
        /// <param name="cid">話者CID</param>
        /// <param name="talkText">発声させるテキスト</param>
        /// <param name="saveFilename">保存先ファイル名</param>
        /// <returns>0.0ミリ秒固定</returns>
        public override double Save(int cid, string talkText, string
saveFilename)
        {
            int avatorIndex = ConvertAvatorIndex(cid);
            int speed =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
speed].value);
            int tone =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
pitch].value);
            int volume =
Convert.ToInt32(AvatorParams[avatorIndex].VoiceEffects[EnumVoiceEffect.
volume].value);
```

```
            AvatorSelect(avatorIndex);

            ApplyEffectParameters(avatorIndex);
            ApplyEmotionParameters(avatorIndex);

            try
            {
                //再生中なら待つ
                while (BouyomiTalkProxy.NowPlaying == true)
                {
                    Thread.Sleep(10);
                }

                BouyomiTalkProxy.AddTalkTask2(talkText, speed, tone,
volume, VT);

                //再生中フラグが立つまで待つ
                while (BouyomiTalkProxy.NowPlaying == false)
                {
                    Thread.Sleep(10);
                }

                //再生終了まで待つ
                while (BouyomiTalkProxy.NowPlaying == true)
                {
                    Thread.Sleep(10);
                }
            }
            catch (Exception)
            {
                //
            }

            return 0.0;
        }

        public override void ResetVoiceEmotion(int cid)
        {
        }

        public override void ResetVoiceEffect(int cid)
        {
            int avatorIndex = ConvertAvatorIndex(cid);

            foreach (var effect in
AvatorParams[avatorIndex].VoiceEffects_default)
            {
AvatorParams[avatorIndex].VoiceEffects[effect.Key].value =
effect.Value.value;
```

```
            }

            ApplyEffectParameters(avatorIndex);
        }

        private void AvatorSelect(int avatorIndex)
        {
            switch (avatorIndex)
            {
                case 0: VT = VoiceType.Female1; break;
                case 1: VT = VoiceType.Female2; break;
                case 2: VT = VoiceType.Male1; break;
                case 3: VT = VoiceType.Male2; break;
                case 4: VT = VoiceType.Imd1; break;
                case 5: VT = VoiceType.Robot1; break;
                case 6: VT = VoiceType.Machine1; break;
                case 7: VT = VoiceType.Machine2; break;
            }
        }

        private void ApplyEmotionParameters(int avatorIndex)
        {
        }

        private void ApplyEffectParameters(int avatorIndex)
        {
        }

    }
}
```

技術資料, Windows, voiceroid, Voiceroid2, CeVIO, 棒読みちゃん, Codeer.Friendly