VoiceAttack Quick Start Guide (v1.7.7+) 3
VoiceAttackクイックスタートガイド(v1.7.7 +)3

VoiceAttack Screen Guide 5
VoiceAttack画面ガイド5

VoiceAttack's Main Screen 5
VoiceAttackのメイン画面5

Profile and Profile Options Screens 9
プロファイルおよびプロファイルオプション画面9

Command Screen 17
コマンド画面17

Key Press Screen 32
キー押下画面32

Pause Screen 36
一時停止画面36

Variable Pause Screen 36
可変一時停止画面36

Other Stuff Screen 37
その他のスタッフ画面37

Key Press / Mouse Event Recorder Screen 96
キー押下/マウスイベントレコーダ画面96

Mouse Action Screen 98
マウスアクション画面98

Registration Screen 102
登録画面102

Options Screen 103
オプション画面103

Exporting Profiles 118
プロファイルのエクスポート118

Creating Quick-Reference Lists 120
クイックリファレンスリストの作成120

Importing Profiles and Profile Packages 121
プロファイルとプロファイルパッケージのインポート121

Importing Individual Commands 122
個々のコマンドのインポート122

No Speech Engine / Alternate Speech Engines in VoiceAttack 123
VoiceAttack 123の音声エンジンなし/代替音声エンジン

Using the Condition Builder 125
条件ビルダーの使用125

Command Line Options 131
コマンドラインオプション131

Text (and Text-To-Speech) Tokens 135
テキスト(および音声合成)トークン135

VoiceAttack Path Tokens 162
VoiceAttackパストークン162

Quick Input, Variable Keypress and Hotkey Key Indicators 163
クイック入力、可変キープレス、ホットキーキーインジケーター163

Key State Token Parameter Values 166
キー状態トークンのパラメーター値166

VoiceAttack Plugins (for the truly mad) 168
VoiceAttackプラグイン(真に狂った人向け)168

VoiceAttack Load Options Screen 200
VoiceAttackロードオプション画面200

Advanced Variable Control (Scope) 202
高度な変数制御(スコープ)202

Application Focus (Process Target) Guide 204
アプリケーションフォーカス(プロセスターゲット)ガイド204

Command Execution Queues Overview 212
コマンド実行キューの概要212

VoiceAttack Profile Package Reference 215
VoiceAttackプロファイルパッケージリファレンス215

Troubleshooting Guide 217
トラブルシューティングガイド217

Setting up Microphone Input 220
マイク入力220のセットアップ

VoiceAttack's Data Storage 221
VoiceAttackのデータストレージ221

VoiceAttack Author Flags 222
VoiceAttack作成者フラグ222

For fun... maybe 225
楽しみのために...多分225

VoiceAttack Quick Start Guide (v1.7.7+)
VoiceAttackクイックスタートガイド (v1.7.7 +)

A few things that you'll need for VoiceAttack to work:
VoiceAttackが機能するために必要なもの:

1. Microsoft Windows Vista, 7, 8, 8.1, 10 or XP. Windows Vista and up come with the Windows Speech Recognition Engine built in. Windows XP, by default, does not. If your copy of Windows XP does not have these components, you will need to download them from the VoiceAttack site:
1. Microsoft Windows Vista、7、8、8.1、10、またはXP。Windows Vista以降には、Windows音声認識エンジンが組み込まれています。WindowsXPは、デフォルトでは組み込まれていません。Windows XPのコピーにこれらのコンポーネントがない場合は、VoiceAttackサイトからそれらをダウンロードする必要があります。

http://www.voiceattack.com/download_for_xp.aspx
http://www.voiceattack.com/download_for_xp.aspx

You will know right away when you launch VoiceAttack if you do not have the Windows Speech Recognition Engine :) Note: a link will appear with the same address listed above.
Windows音声認識エンジンがない場合は、VoiceAttackを起動するとすぐにわかります:)注:上記と同じアドレスのリンクが表示されます。

If you feel adventurous and find out that VoiceAttack works on other versions of Windows, please let us know & we'll make sure to update this document.
冒険心があり、VoiceAttackが他のバージョンのWindowsで動作することがわかった場合は、お知らせください。このドキュメントを必ず更新します。

2. The .Net Framework v4.5. This is a requirement. The installer will show you where to get it if you don't already have it.
2. .Net Framework v4.5。これは要件です。インストーラーは、まだ入手していない場合に入手先を示します。

3. A microphone... Although not technically REQUIRED for the program to run, you're not going to get very far without one. A USB headset is recommended, since you are probably going to use VoiceAttack to play games. Setting up your microphone properly to work with Windows' speech recognition is a very vital step. There is a short section on how to do this, near the end of this document (See, 'Setting up Microphone Input').
3. マイク...プログラムを実行するために技術的には必要ではありませんが、マイクなしではそれほど遠くに行くことはできません。おそらくVoiceAttackを使用してゲームをプレイするため、USBヘッドセットの使用をお勧めします。Windowsの音声認識で動作するようにマイクを適切に設定することは、非常に重要なステップです。この方法については、このドキュメントの終わり近くに短いセクションがあります(「マイク入力の設定」を参照)。

4. A voice (see #3). :)
4. 音声(#3を参照)。:)

5. Your Windows Speech Recognition Engine needs to be trained up. Again, not an absolute requirement, however, the difference between a trained and untrained system is like night and day.
5. Windows音声認識エンジンをトレーニングする必要があります。繰り返しますが、絶対的な要件ではありませんが、訓練されたシステムと訓練されていないシステムの違いは、昼と夜のようなものです。

Hint: Start in Control Panel... I ran the trainer three times in a row & now recognition works great!
ヒント:コントロールパネルで起動します...トレーナーを3回連続で実行しましたが、認識機能は非常に優れています。

Getting things going...
物事を進める...

Once you've got VoiceAttack installed and you have found out that you meet all the requirements outlined above, you can just jump right in. To keep things simple, we're going to assume that you are working with the trial version of VoiceAttack, and this is your first time running VoiceAttack on this computer. If your microphone is on and the input volume of your microphone is properly set, you should see VoiceAttack's Level bar moving when you speak. If the Level bar is not moving, VoiceAttack can't hear you. See the 'Troubleshooting Guide' at the end of this document.
VoiceAttackをインストールし、上記のすべての要件を満たしていることがわかったら、すぐに使用できます。簡単にするために、VoiceAttackの試用版を使用していると仮定します。このコンピューターでVoiceAttackを実行するのは初めてです。マイクがオンになっていて、マイクの入力音量が適切に設定されている場合、話すときにVoiceAttackのレベルバーが動くのが見えるはずです。レベルバーが動いていない場合、VoiceAttackはあなたの声を聞くことができません。このドキュメントの最後にある「トラブルシューティングガイド」を参照してください。

Notice that VoiceAttack is pretty much not recognizing anything you say. This is good, because we have not added any commands to the profile. Click the 'Edit' button to view the commands for this profile.
VoiceAttackはあなたの言うことをほとんど認識していないことに注意してください。プロファイルにコマンドを追加していないため、これは良いことです。[編集]ボタンをクリックして、このプロファイルのコマンドを表示します。

What is a command? A command is simply a word or phrase you are going to say to VoiceAttack. When VoiceAttack recognizes the command that you say, it will perform a series of actions (which can be keyboard key presses, pauses, mouse clicks, application launches,
コマンドとは何ですか？コマンドは、VoiceAttackに発言する単語またはフレーズです。VoiceAttackがあなたの言うコマンドを認識すると、一連のアクション（キーボードのキーの押下、一時停止、マウスクリック、アプリケーションの起動、

sound effects, etc.).
効果音など）。

Notice that VoiceAttack has a set of commands already set up for demonstration purposes. You can edit and/or remove all of these items. Let's try one out! Hit the, 'Cancel' button to go back to the Main screen. Now, you must speak into your microphone. Say the word, 'Calculator'. If everything is lined up right (microphone is on, you have adequate volume and your speech recognition engine is trained up), you should see the Windows calculator on your screen. Now say, 'Close Calculator'. The calculator should now be closed. If you are not seeing the Windows calculator, please refer to the Troubleshooting Guide at the end of this document.
VoiceAttackには、デモ目的で既にセットアップされた一連のコマンドがあることに注意してください。これらのアイテムはすべて編集または削除できます。試してみましょう！「キャンセル」ボタンを押して、メイン画面に戻ります。次に、マイクに向かって話す必要があります。「電卓」という言葉を発声してください。すべてが正しく並んでいる場合（マイクがオンで、十分な音量があり、音声認識エンジンがトレーニングされている場合）、画面にWindows計算機が表示されます。次に、「電卓を閉じる」と言います。これで電卓が閉じられます。Windows計算機が表示されない場合は、このドキュメントの最後にあるトラブルシューティングガイドを参照してください。

Note that this help file is kind of basic and doesn't explain some things in super deep detail. Swing by the VoiceAttack user forum or Discord server for more detailed explanations/conversations about pretty much everything:
このヘルプファイルは基本的なものであり、詳細については説明していません。VoiceAttackユーザーフォーラムまたはDiscordサーバーでスイングして、ほとんどすべての詳細な説明/会話をご覧ください。

http://voiceattack.com/forum https://discord.gg/v7qf36M
http://voiceattack.com/forum https://discord.gg/v7qf36M

Thank you for trying VoiceAttack!
VoiceAttackをお試しいただきありがとうございます！

PS – This help document is also available online, so please save a tree by not printing this huge document that changes a lot (pretty please).
PS–このヘルプドキュメントはオンラインでも利用できます。このため、大きく変化するこの巨大なドキュメントを印刷しないでツリーを保存してください（かなりお願いします）。

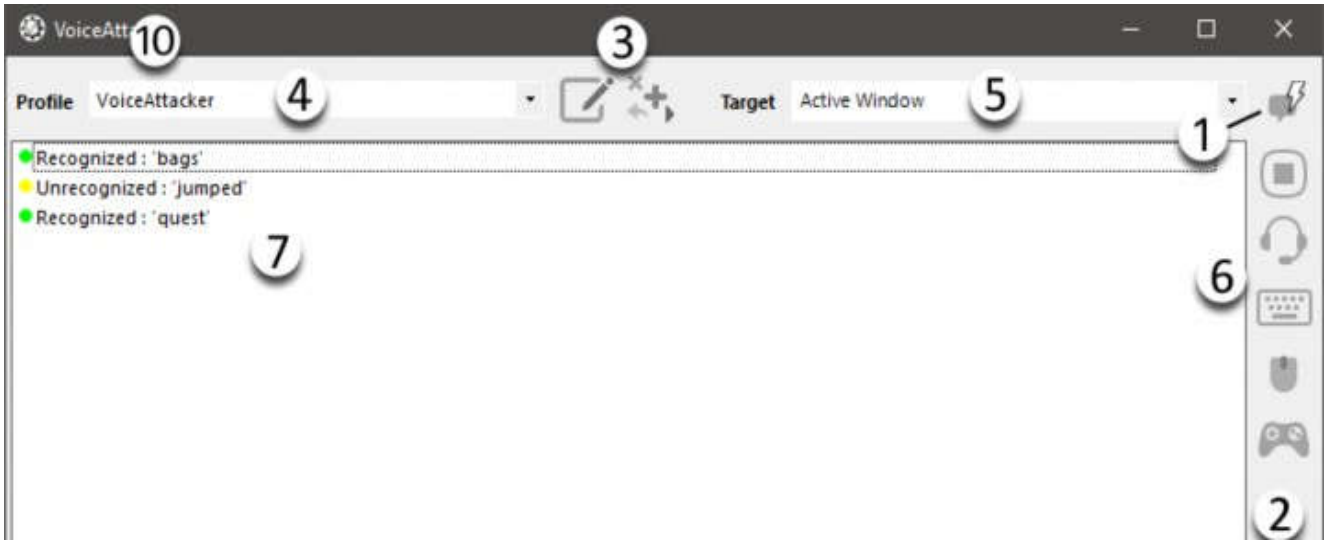
http://www.voiceattack.com/help
http://www.voiceattack.com/help

VoiceAttack Screen Guide
VoiceAttack画面ガイド

This document will make an attempt to explain the main features of VoiceAttack.
このドキュメントでは、VoiceAttackの主な機能を説明しようとします。

VoiceAttack's Main Screen
VoiceAttackのメイン画面

This is the main hub for all VoiceAttack activity. There's a lot going on here, so, I've numbered the main areas of the screen and describe each part below.
これは、すべてのVoiceAttackアクティビティのメインハブです。ここでは多くのことが行われているので、画面の主要な領域に番号を付け、以下の各部分について説明します。



1. – Audio indicator
1.– 音声インジケータ

This icon indicates the status of your commands (recognized, unrecognized, error), as well as a way to tell if your mic is muted.
このアイコンは、コマンドのステータス（認識、認識、エラー）、およびマイクがミュートされているかどうかを示す方法を示します。

2. – Options button
2.– オプションボタン

Opens the options screen where you will find various settings for VoiceAttack (See 'Options Screen'). Additionally, the registration screens for VoiceAttack are available through this button (See 'Registration Screen').
VoiceAttackのさまざまな設定があるオプション画面を開きます（「オプション画面」を参照）。さらに、VoiceAttackの登録画面はこのボタンから利用できます（「登録画面」を参照）。

3. – Profile management buttons
3.– プロファイル管理ボタン

These two buttons will allow you to edit, delete, export and duplicate your currently– selected profile or create or import a new profile. Note: Creating, importing, deleting and duplicating profiles is only available in the registered version of VoiceAttack.
これらの2つのボタンを使用すると、現在選択されているプロファイルを編集、削除、エクスポート、複製したり、新しいプロファイルを作成またはインポートしたりできます。注：プロファイルの作成、インポート、削除、および複製は、VoiceAttackの登録済みバージョンでのみ使用可能です。

4. – Profile selector
4.– プロファイルセレクター

Drop down the list to select one of your created profiles. Each profile contains a set of commands that you specify.
リストをドロップダウンして、作成したプロファイルの1つを選択します。各プロファイルには、指定した一連のコマンドが含まれています。

5. – 'Target' selector
5.– 「ターゲット」セレクター

When VoiceAttack recognizes a command, it needs a place to send input. You can either send input to the Active Window (almost always the most likely case), or, you can choose to send the input to an application with a specified window title, or even a named process. Note that this can be overridden at both the profile and even at the command level (see, 'Profile' and 'Command' screens for more details).
VoiceAttackがコマンドを認識する場合、入力を送信する場所が必要です。アクティブウィンドウに入力を送信するか（ほとんどの場合、ほとんどの場合）、指定したウィンドウタイトルまたは名前付きプロセスにアプリケーションに入力を送信することを選択できます。これは、プロファイルとコマンドレベルの両方でオーバーライドできることに注意してください（詳細については、「プロファイル」および「コマンド」画面を参照してください）。

To choose the Active Window, simply choose, 'Active Window' from the list (it's the first item in the list, and, it is the default selection).
アクティブウィンドウを選択するには、リストから「アクティブウィンドウ」を選択するだけです（リストの最初のアイテムであり、デフォルトの選択です）。

To select a window title or named process, just drop down the list. Choosing a window from the list will indicate to the commands that you issue that you want to send input to that window. To refresh this list, just drop the list down and select, 'Refresh this list'.
ウィンドウのタイトルまたは名前付きプロセスを選択するには、リストをドロップダウンします。リストからウィンドウを選択すると、発行するコマンドに対して、そのウィンドウに入力を送信することが示されます。このリストを更新するには、リストをドロップダウンして、「このリストを更新」を選択します。

This will display the current set of open windows on your system.
これにより、システムで現在開いているウィンドウのセットが表示されます。

Note that this is a free input text box and you can modify your selection (as detailed below). Just select the option labeled, 'Add your own target' or select the blank option and start typing.
これは無料の入力テキストボックスであり、選択を変更できることに注意してください（以下に詳細を示します）。「独自のターゲットを追加」というラベルの付いたオプションを選択するか、空のオプションを選択して入力を開始します。

The value you type into the drop-down box can contain wildcards indicated by asterisks (*). This is handy when the title of the window changes (or even if you just don't feel like typing the entire title). To indicate that the window title contains the value you type in the box, put an asterisk on each end. For instance, if you want to target any window that contains, 'Notepad' in the title, put, '*Notepad*' (without quotes) in the box. To indicate that the window title starts with the value in the box, put an asterisk at the end: 'Notepad*'. To indicate that the window title ends with the value in the box, put an asterisk at the beginning: '*Notepad'. The values are not case- sensitive. The first window found to match the criteria indicated will be selected.
ドロップダウンボックスに入力する値には、アスタリスク(*)で示されるワイルドカードを含めることができます。これは、ウィンドウのタイトルが変更された場合(または、タイトル全体を入力したくない場合でも)便利です。ウィンドウのタイトルにボックスに入力した値が含まれていることを示すには、両端にアスタリスクを付けます。たとえば、タイトルに「メモ帳」と入力し、ボックスに「*メモ帳*」(引用符なし)を含むウィンドウをターゲットにしたい場合。ウィンドウのタイトルがボックス内の値で始まることを示すには、最後にアスタリスク「Notepad *」を付けます。ウィンドウのタイトルがボックス内の値で終わることを示すには、先頭にアスタリスクを付けます: ' *メモ帳」。値は大文字と小文字を区別しません。示された基準に一致する最初のウィンドウが選択されます。

Advanced: Note that you can also use process names as they appear in the Windows Task Manager. You can use wildcards the same as you do with the window titles.
高度:Windowsタスクマネージャーに表示されるプロセス名を使用することもできます。ウィンドウのタイトルと同じようにワイルドカードを使用できます。

Window titles are checked first, and then the process names. Don't worry, this happens very fast.
最初にウィンドウのタイトルがチェックされ、次にプロセス名がチェックされます。心配しないでください、これは非常に高速です。

More advanced: If a window cannot be found by title or process name, the window class names will then be checked. Wildcards apply if you need them. Note that this will be the class name of the window itself and not the class name of a child control. Again, this is an advanced feature that you may never ever use.
より高度な:タイトルまたはプロセス名でウィンドウが見つからない場合、ウィンドウクラス名がチェックされます。ワイルドカードは必要な場合に適用されます。これはウィンドウ自体のクラス名であり、子コントロールのクラス名ではないことに注意してください。繰り返しになりますが、これは使用することのない高度な機能です。

Optimization note - As indicated above, the, 'Target' input box will accept the name of a window title, process name or window class name, and checks for each of these items in that order. The reason for doing all of this in one go is for user simplicity (less user interface) as well as user assistance in locating their intended target (as oftentimes there is an overlap in naming). This works rather quickly in most cases, but in some situations the processing could be unnecessarily excessive. For instance, if you are looking for a process name that contains, 'widget', all window titles will be searched for, 'widget' first before the process names are searched. Again, this is a fast check, but it is unnecessary checking if you already know for sure that, 'widget' will
最適化に関する注意-上記のように、「ターゲット」入力ボックスはウィンドウタイトルの名前、プロセス名、またはウィンドウクラス名を受け入れ、これらの各項目をこの順序でチェックします。このすべてを一度に実行する理由は、ユーザーの単純さ(ユーザーインターフェイスが少ない)と、目的のターゲットを見つける際のユーザーの支援(多くの場合、名前の重複があるため)です。ほとんどの場合、これはかなり速く動作しますが、状況によっては処理が不必要に過剰になる可能性があります。たとえば、「widget」を含むプロセス名を探している場合、プロセス名が検索される前に「widget」が最初にすべてのウィンドウタイトルが検索されます。繰り返しますが、これは高速チェックですが、「ウィジェット」が確実に知っている場合は不要なチェックです

only appear in a process name (and not a window's title). In order to not add more user interface clutter and to provide those of you looking to eek out every bit of performance you can get, VoiceAttack now has character prefixes to limit searches to just the window title, process or class name.
プロセス名にのみ表示されます（ウィンドウのタイトルには表示されません）。ユーザーインターフェースの乱雑さを追加せず、取得できるすべてのパフォーマンスを探しているユーザーに提供するために、VoiceAttackには、検索をウィンドウタイトル、プロセス、またはクラス名のみに制限する文字プレフィックスがあります。

If '^' is prepended to the target value, the target search is limited to only window titles. If '~' is prepended, only process names are searched.
' ^ 'がターゲット値の前に追加された場合、ターゲット検索はウィンドウタイトルのみに制限されます。「～」が先頭に追加された場合、プロセス名のみが検索されます。

If '+' is prepended, only class names are searched.
' + 'が先頭に追加された場合、クラス名のみが検索されます。

These new prefixes can also be used in conjunction with wildcards.
これらの新しいプレフィックスは、ワイルドカードと組み合わせて使用することもできます。

For example, '^*notepad*' will indicate to only search window titles that contain, 'notepad'. If, 'notepad' is not found within a window title, the search will stop and not continue on to search processes or class names. '~widget*' indicates that the search is to only look for process names that start with, 'widget'. That means that no window titles or class names are searched. '+foo' indicates that the search should only be for window class names that match, 'foo' (again, window titles and process names are not searched first which saves some time).
たとえば、「^ * notepad *」は、「notepad」を含む検索ウィンドウタイトルのみを示します。ウィンドウタイトル内に「notepad」が見つからない場合、検索は停止し、検索プロセスまたはクラス名に進みません。「～widget *」は、「widget」で始まるプロセス名のみを検索することを示します。つまり、ウィンドウのタイトルやクラス名は検索されません。「+ foo」は、一致するウィンドウクラス名のみを検索する必要があることを示します。「foo」（再び、ウィンドウタイトルとプロセス名は最初に検索されないため、時間を節約できます）。

See, 'Application Focus (Process Target) Guide' near the end of this document for more help on this topic.
このトピックの詳細については、このドキュメントの最後にある「アプリケーションフォーカス（プロセスターゲット）ガイド」を参照してください。

6. – Listening button
6.– リスニングボタン

This is a toggle button that enables/disables VoiceAttack's 'listening'. That is, VoiceAttack will stop performing actions on commands that it recognizes. The only commands VoiceAttack will process are the commands that tell VoiceAttack to start listening again (if you specified one or more... see 'Command Screen' for more info). The hotkey for this button can be configured through the VoiceAttack Options screen
これは、VoiceAttackの「リスニング」を有効/無効にするトグルボタンです。つまり、VoiceAttackは、認識したコマンドに対するアクションの実行を停止します。VoiceAttackが処理するコマンドは、VoiceAttackに再度リスニングを開始するように指示するコマンドです（1つ以上を指定した場合...詳細については、「コマンド画面」を参照）。このボタンのホットキーは、VoiceAttackオプション画面から設定できます

Keyboard shortcuts toggle
キーボードショートカットの切り替え

This toggle button enables/disables VoiceAttack's keyboard shortcuts.
このトグルボタンは、VoiceAttackのキーボードショートカットを有効/無効にします。

Mouse shortcuts toggle
マウスショートカットの切り替え

This toggle button enables/disables VoiceAttack's mouse button shortcuts.
このトグルボタンは、VoiceAttackのマウスボタンショートカットを有効/無効にします。

Joystick button toggle
ジョイスティックボタンの切り替え

This toggle button enables/disables VoiceAttack's joystick button detection.
このトグルボタンは、VoiceAttackのジョイスティックボタンの検出を有効/無効にします。

Stop Commands button
[停止コマンド]ボタン

This will halt all macros that are in progress. Useful if your macros happen to be long- running. Note this will also stop any playing sounds or text-to-speech, and any keys that are pressed down will be released.
これにより、進行中のすべてのマクロが停止します。マクロが実行時間の長い場合に役立ちます。これにより、再生中の音声や音声合成も停止し、押されたキーはすべて解除されます。

7. - Recognition log
7.- 認識ログ

This log shows what VoiceAttack is 'hearing' and the actions that VoiceAttack is invoking. This list is quite useful when determining if you need to speak more clearly or rethink the names of your commands. It's also a lot of fun to say weird phrases and
このログは、VoiceAttackが「ヒアリング」していることと、VoiceAttackが呼び出しているアクションを示します。このリストは、より明確に話す必要があるかどうか、またはコマンドの名前を再考する必要があるかどうかを判断するときに非常に役立ちます。変なフレーズを言うのも楽しいし、

see what the speech recognition engine *thinks* you said. Fun for the whole family (maybe). Right-clicking on this log will allow you to copy and clear the text, as well as allow you to specify some options. For instance, you can indicate if new log entries appear at the top or at the bottom of the log, as well as specify how many entries the log will hold. You can also filter out unrecognized commands or just show the latest log entry only.
音声認識エンジンが*あなたが言ったと思う*を参照してください。家族全員で楽しめる(たぶん)。このログを右クリックすると、テキストをコピーしてクリアしたり、いくつかのオプションを指定したりできます。たとえば、新しいログエントリがログの上部に表示されるか下部に表示されるかを示したり、ログが保持するエントリの数を指定したりできます。認識されないコマンドを除外したり、最新のログエントリのみを表示したりすることもできます。

Note: Right-clicking (or double-clicking) a 'Recognized' log entry will take you to the 'Edit Command' screen for the selected command in the current profile. If the log entry is 'Unrecognized', you will be taken to the 'Add Command' screen. This is to aid in testing new commands and has probably saved me a few pulled hairs. Your mileage may vary :)
注:[認識済み]ログエントリを右クリック(またはダブルクリック)すると、現在のプロファイルで選択したコマンドの[コマンドの編集]画面が表示されます。ログエントリが「認識されない」場合、「コマンドの追加」画面に移動します。これは、新しいコマンドのテストを支援するためであり、おそらくいくつかの髪の毛を節約できました。あなたのマイレージは異なる場合があります :)

8. - Level Bar
8.- レベルバー

A graphical indicator of the microphone input for VoiceAttack. My guess is that the first thing you will do when you are launching VoiceAttack is to look at this bar... and then say, 'hellooooooooooo' into your microphone to see if all is working. Maybe that's just me. Note that this bar will turn red as an additional indicator when listening is turned off.
VoiceAttackのマイク入力のグラフィカルインジケーター。私の推測では、VoiceAttackを起動するときに最初に行うことは、このバーを見てから、マイクに「hellooooooooooo」と言って、すべてが機能しているかどうかを確認することです。たぶんそれは私だけです。リスニングがオフになると、このバーは追加のインジケータとして赤に変わります。

9. - Compact Mode
9.- コンパクトモード

Click this button to toggle between VoiceAttack's full-size and compact modes.
このボタンをクリックして、VoiceAttackのフルサイズモードとコンパクトモードを切り替えます。

10. - Context Menu
10.- コンテキストメニュー

The context menu that is accessible from the icon in the top-left corner contains various options and functions:
左上隅のアイコンからアクセスできるコンテキストメニューには、さまざまなオプションと機能が含まれています。

Always on Top – VoiceAttack will remain as the top-most application while this option is turned on.
常に手前-VoiceAttackは、このオプションがオンになっている間、一番上のアプリケーションとして残ります。

Help Document – Access the VoiceAttack help documentation that is located in the VoiceAttack installation directory.
ヘルプドキュメント-VoiceAttackインストールディレクトリにあるVoiceAttackヘルプドキュメントにアクセスします。

VoiceAttack User Forum – This will launch your browser pointed to the VoiceAttack user forum. Lots of great help from some really great people in there.
VoiceAttackユーザーフォーラム-これにより、VoiceAttackユーザーフォーラムを指すブラウザーが起動します。そこには本当に素晴らしい人々からのたくさんの大きな助けがあります。

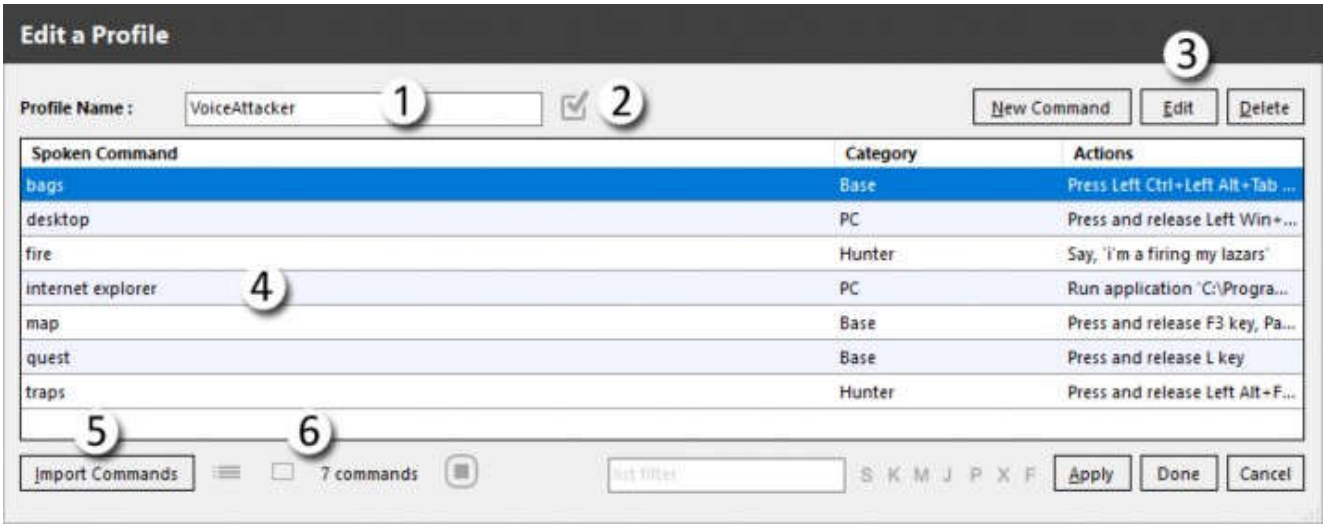
Reset Speech Recognition – This item will reset the speech engine that VoiceAttack is currently using. Although a properly configured and working speech engine generally requires no reset during a VoiceAttack running session, this is in place if you need it.
音声認識のリセット-このアイテムは、VoiceAttackが現在使用している音声エンジンをリセットします。適切に設定され動作している音声エンジンは、通常、VoiceAttackの実行中のセッション中にリセットする必要はありませんが、必要に応じてこれを設定します。

Mute Speech Recognition Device – Checking and unchecking this option mutes and unmutes the recording device that he speech engine is currently using.
音声認識デバイスのミュート-このオプションをオンまたはオフにすると、音声エンジンが現在使用している録音デバイスのミュートとミュート解除が行われます。

Cover of Darkness – Checking and unchecking this option will toggle VoiceAttack's 'dark' mode for nighttime use.
カバーオブダークネス-このオプションをオンまたはオフにすると、夜間のVoiceAttackの「ダーク」モードが切り替わります。

Profile and Profile Options Screens
プロファイルおよびプロファイルオプション画面

This is where you will update each of your profiles. A profile is basically a set of commands that you define.
ここで、各プロファイルを更新します。プロファイルは基本的に、ユーザーが定義する一連のコマンドです。



1. – Profile Name
1.– プロファイル名

Type a unique name here for your profile. Make something descriptive!
プロファイルの一意の名前をここに入力します。わかりやすいものにしてください！

2. – Profile Options
2.– プロファイルオプション

This is where you can set up some additional attributes of your profile, as well as override some of the global items found in the Options screen (such as the various ways of turning on and off VoiceAttack's listening).
ここで、プロファイルの追加属性を設定したり、オプション画面にあるグローバル項目の一部をオーバーライドしたりできます（VoiceAttackのリスニングをオンまたはオフにするさまざまな方法など）。

Click this button to go to the profile options/global override screen shown below (four tabs, General, Exec, Hotkeys and Advanced):
このボタンをクリックして、以下に示すプロファイルオプション/グローバルオーバーライド画面に移動します（4つのタブ、[全般]、[実行]、[ホットキー]、[詳細]）。

Profile Options General Tab
プロファイルオプションの[全般]タブ

The, 'Override listening if my spoken command begins with...' option was added as a fun way to interact with VoiceAttack (even if VoiceAttack is not listening). If you say what is in the input box before your command, VoiceAttack will listen to that command and go back to not listening. For instance, let's say you have a spoken command called, 'Attack', and VoiceAttack's listening is off. Let's also say that your listening override is, 'computer' (as shown). While listening is off, you can say, 'Attack' all day, and nothing will happen. If this option is on, you can say, 'Computer, Attack' and VoiceAttack will execute the, 'Attack' command (and then resume not listening).
「音声コマンドが次で始まる場合はリスニングをオーバーライドします...」オプションが、VoiceAttackと対話する楽しい方法として追加されました(VoiceAttackがリスニングしていない場合でも)。コマンドの前に入力ボックスの内容を言うと、VoiceAttackはそのコマンドをリッスンし、リッスンしない状態に戻ります。たとえば、「攻撃」と呼ばれる音声コマンドがあり、VoiceAttackのリスニングがオフになっているとします。また、リスニングオーバーライドが「コンピューター」であるとしましょう(図を参照)。リスニングがオフになっている間、一日中「攻撃」と言うことができますが、何も起こりません。このオプションがオンの場合、「コンピューター、攻撃」と言うことができ、VoiceAttackは「攻撃」コマンドを実行します(その後、リスニングを再開します)。

'Override global minimum confidence level' allows you to set the minimum confidence level of recognized phrases. This overrides the value set globally on the, 'Options' screen. Note that you can override this value on individual commands as well. See, 'Options' screen for more information about confidence levels.
「グローバル最小信頼レベルのオーバーライド」では、認識されたフレーズの最小信頼レベルを設定できます。これは、「オプション」画面でグローバルに設定された値を上書きします。個々のコマンドでもこの値をオーバーライドできることに注意してください。信頼レベルの詳細については、「オプション」画面を参照してください。

'Include commands from other profiles' will allow you to reference, or, 'include' the commands from any or all of your other profiles. This way, you can create common profiles filled with commands that can be shared among several other profiles. The profiles that you include can be arranged in priority, from highest to lowest. When duplicate-named commands are encountered, the command in the profile with the higher priority will be retained. For instance, let's say you have two profiles that you want to include: Profile A and Profile B. Profile A has been given a higher priority than Profile B (it's higher up on the list). Both profiles have a command called, 'Fire Weapons'. When you issue the command, 'Fire Weapons', the command from Profile A will be used, since Profile A has a higher priority.
「他のプロファイルのコマンドを含める」を使用すると、他のプロファイルの一部またはすべてのコマンドを参照したり、「含める」ことができます。この方法で、他の複数のプロファイル間で共有できるコマンドで満たされた共通プロファイルを作成できます。含めるプロファイルは、優先度の高いものから順に並べることができます。重複した名前のコマンドが検出されると、プロファイル内の優先度の高いコマンドが保持されます。たとえば、含めるプロファイルとしてプロファイルAとプロファイルBの2つのプロファイルがあるとします。プロファイルAにはプロファイルBよりも高い優先順位が与えられています(リストの上位にあります)。両方のプロファイルには、「Fire Weapons」というコマンドがあります。コマンド「Fire Weapons」を発行すると、プロファイルAの優先度が高いため、プロファイルAのコマンドが使用されます。

Note: The included profiles in this set are higher priority than the Global Profiles selected from the Options screen (See, 'Global Profiles' on the Options screen). Also, the current, active profile always has the highest priority.
注:このセットに含まれるプロファイルは、オプション画面で選択されたグローバルプロファイルよりも優先されます(オプション画面の「グローバルプロファイル」を参照)。また、現在のアクティブなプロファイルは常に最高の優先度を持ちます。

To edit the list of included profiles, click on the, '…' button to the right. This will bring up the, 'Include Profile Commands' screen. Use the controls on the right side of the screen to add, arrange and delete included profiles. Click, 'OK' when you are finished.
含まれるプロファイルのリストを編集するには、右側の「…」ボタンをクリックします。これにより、[プロファイルコマンドを含める]画面が表示されます。画面の右側にあるコントロールを使用して、含まれるプロファイルを追加、配置、削除します。完了したら、「OK」をクリックします。

'Enable profile switching for the following windows' – Checking this option will turn on automatic profile switching for this profile (this option works in conjunction with the option, 'Enable Auto Profile Switching' which is located on the Options screen. What automatic profile switching does is it allows you to specify one or more running applications to look out for. If a specified application is detected as the foreground window, VoiceAttack will automatically switch to this profile. To keep things as simple as possible, there is a text box that allows you to input the name of the window of the application that you want to look for. The input for this box is semicolon-delimited so you can associate your profile with more than one application. Since window titles vary depending on what you are doing, you can also add asterisks (*) as kind of a basic wildcard. If you put the asterisk at the end of the title, the search becomes, 'starts with' (for example, 'Notepad*'). If you put the asterisk at the beginning ('*Notepad'), the search becomes, 'ends with'. If you put an asterisk on both ends ('*Notepad*'), the search becomes, 'contains'. No asterisks ('Notepad') means a direct comparison (equals), and a single asterisk ('*') indicates that the profile is to be switched to if no other matches have been made (If VoiceAttack is the active window, the profile will not automatically switch (for obvious reasons)).
' 次のウィンドウのプロファイル切り替えを有効にします'-このオプションをオンにすると、このプロファイルの自動プロファイル切り替えが有効になります(このオプションは、[オプション]画面にある[自動プロファイル切り替えを有効にする]オプションと連動します。自動プロファイル切り替えでは、指定したアプリケーションがフォアグラウンドウィンドウとして検出された場合、VoiceAttackは自動的にこのプロファイルに切り替わります。可能な限りシンプルにするために、名前を入力できるテキストボックスがあります。検索するアプリケーションのウィンドウ。このボックスへの入力はセミコロンで区切られているため、プロファイルを複数のアプリケーションに関連付けることができます。ウィンドウタイトルは実行内容によって異なるため、アスタリスク(*)基本的なワイルドカードの一種として。タイトルの最後にアスタリスクを付けると、検索は「次で始まる」になります(たとえば、「メモ帳」* ')。アスタリスクを先頭('* Notepad')に置くと、検索は 'ends with'になります。両端にアスタリスクを付けると(「*メモ帳*」)、検索は「含む」になります。アスタリスクなし(「メモ帳」)は直接比較(等しい)を意味し、単一のアスタリスク(「*」)は、他の一致がない場合にプロファイルが切り替えられることを示します(VoiceAttackがアクティブなウィンドウの場合、プロファイル(明白な理由のため)自動的に切り替わりません)。

So, let's say you want your profile to automatically change when you switch over to either your desktop or Notepad. The desktop window name (oddly enough) is, 'Program Manager' (I know that's weird... there's some help about this down below). Notepad's window title will change depending on the document you are editing. Your input would look like this: 'Program Manager;*Notepad*' (without the quotes). That means VoiceAttack will look for a window titled, 'Program Manager' as well as any window that has a title that contains, 'Notepad'.
そのため、デスクトップまたはメモ帳に切り替えたときにプロファイルを自動的に変更したいとします。デスクトップウィンドウ名(奇妙なことに)は、「プログラムマネージャー」です(これは奇妙なことです...これについては、以下にいくつかのヘルプがあります)。メモ帳のウィンドウタイトルは、編集中のドキュメントに応じて変わります。入力は次のようになります:
'Program Manager; * Notepad *'(引用符なし)。つまり、VoiceAttackは、「プログラムマネージャー」というタイトルのウィンドウと、「メモ帳」を含むタイトルを持つウィンドウを検索します。

To help with finding out window titles, a new option has been added to the VoiceAttack Load Options screen. To get to this screen, simply start VoiceAttack while holding down CTRL + Shift. Select the option titled, 'Show window titles (requires 'Enable
ウィンドウのタイトルを見つけやすくするために、VoiceAttack Load Options画面に新しいオプションが追加されました。この画面を表示するには、CTRL + Shiftを押しながらVoiceAttackを開始します。「ウィンドウのタイトルを表示する」というタイトルのオプションを選択します(「有効にする

Automatic Profile Switching' to be checked in the Options screen).' This will show the window titles in the log so you can see what VoiceAttack sees. Check out the Load Options screen for more details.
オプション画面でチェックするプロファイルの自動切り替え」)。これにより、ログにウィンドウのタイトルが表示されるため、VoiceAttackの表示内容を確認できます。詳細については、[読み込みオプション]画面をご覧ください。

Update: VoiceAttack will now also allow you to search by process name in addition to window title. Just include the process name as you would the window title as indicated above, including wildcards. VoiceAttack will first search by window title, then by process name. Note that this is an advanced feature, so, if you have no idea what's being said here, just ignore this ;)
更新：VoiceAttackでは、ウィンドウタイトルに加えてプロセス名でも検索できるようになりました。上記のウィンドウタイトルと同様に、ワイルドカードを含むプロセス名を含めるだけです。VoiceAttackは、最初にウィンドウタイトルで検索し、次にプロセス名で検索します。これは高度な機能であるため、ここで何が言われているのかわからない場合は、これを無視してください;)

'Send commands to this target' – Enabling this option will make this profile's commands target either the active window or another window/process that you specify to receive input. This will override the target indicated on the main screen (global level). Lots of detail about process targets can be found in this document in the section titled, 'Application Focus (Process Target) Guide'. Note that this can be overridden at the command level (see the 'Command Screen' for details).
' このターゲットにコマンドを送信 '–このオプションを有効にすると、このプロファイルのコマンドは、アクティブなウィンドウまたは入力を受け取るように指定した別のウィンドウ/プロセスをターゲットにします。これは、メイン画面（グローバルレベル）に示されたターゲットをオーバーライドします。プロセスターゲットに関する詳細は、このドキュメントの「アプリケーションフォーカス（プロセスターゲット）ガイド」というタイトルのセクションにあります。これはコマンドレベルでオーバーライドできることに注意してください（詳細については、「コマンド画面」を参照してください）。

To send input to the active window, choose, 'Active Window'. Choosing the active window at this level is handy when the global settings are directed to a specific application. Whatever window that is currently active will receive input from
アクティブウィンドウに入力を送信するには、「アクティブウィンドウ」を選択します。グローバル設定が特定のアプリケーションに向けられている場合、このレベルでアクティブウィンドウを選択すると便利です。現在アクティブなウィンドウが何であれ、入力を受け取ります

the commands in this profile.
このプロファイルのコマンド。

To send input to a specific window or process, choose the option next to the dropdown box. To see what windows are available, drop down the list. Choosing a window from the list will indicate to the commands in this profile that you want to send input to it.
特定のウィンドウまたはプロセスに入力を送信するには、ドロップダウンボックスの横にあるオプションを選択します。使用可能なウィンドウを確認するには、リストをドロップダウンします。リストからウィンドウを選択すると、このプロファイルのコマンドに入力を送信するよう指示されます。

Note that this is a free input text box and you can modify your selection (as detailed below).
これは無料の入力テキストボックスであり、選択を変更できることに注意してください（以下に詳細を示します）。

The value in the dropdown box can contain wildcards indicated by asterisks (*). This is handy when the title of the window changes. To indicate that the window title contains the value in the box, put an asterisk on each end. For instance, if you want to target any window that contains, 'Notepad' in the title, put, '*Notepad*' (without quotes) in the box. To indicate that the window title starts with the value in the box, put an asterisk at the end: 'Notepad*'. To indicate that the window title ends with the
ドロップダウンボックスの値には、アスタリスク（*）で示されるワイルドカードを含めることができます。これは、ウィンドウのタイトルが変更されたときに便利です。ウィンドウのタイトルにボックスの値が含まれていることを示すには、両端にアスタリスクを付けます。たとえば、タイトルに「メモ帳」と入力し、ボックスに「*メモ帳*」（引用符なし）を含むウィンドウをターゲットにしたい場合。ウィンドウのタイトルがボックス内の値で始まることを示すには、最後にアスタリスク「Notepad *」を付けます。ウィンドウのタイトルがで終わることを示すために

value in the box, put an asterisk at the beginning: '*Notepad'. The values are not case- sensitive. The first window found to match the criteria indicated will be selected.
ボックスに値を入力し、先頭にアスタリスク「* Notepad」を入力します。値は大文字と小文字を区別しません。示された基準に一致する最初のウィンドウが選択されます。

Advanced: Note that you can also use process names as they appear in the Windows Task Manager. You can use wildcards the same as you do with the window titles.
高度：Windowsタスクマネージャーに表示されるプロセス名を使用することもできます。ウィンドウのタイトルと同じようにワイルドカードを使用できます。

Window titles are checked first, and then the process names.
最初にウィンドウのタイトルがチェックされ、次にプロセス名がチェックされます。

More advanced: If a window cannot be found by title or process name, the window class names will then be checked. Wildcards apply if you need them. Note that this will be the class name of the window itself and not the class name of a child control. Again, this is an advanced feature that you may never ever use.
より高度な：タイトルまたはプロセス名でウィンドウが見つからない場合、ウィンドウクラス名がチェックされます。ワイルドカードは必要な場合に適用されます。これはウィンドウ自体のクラス名であり、子コントロールのクラス名ではないことに注意してください。繰り返しになりますが、これは使用することのない高度な機能です。

Optimization note - As indicated above, the, 'target' input box will accept the name of a window title, process name or window class name, and checks for each of these items in that order. The reason for doing all of this in one go is for user simplicity (less
最適化に関する注意-上記のように、「ターゲット」入力ボックスはウィンドウタイトルの名前、プロセス名、またはウィンドウクラス名を受け入れ、これらの各項目をこの順序でチェックします。このすべてを一度に実行する理由は、ユーザーの単純さ（より少ない

user interface) as well as user assistance in locating their intended target (as oftentimes there is an overlap in naming). This works rather quickly in most cases, but in some situations the processing could be unnecessarily excessive. For instance, if you are looking for a process name that contains, 'widget', all window titles will be searched for, 'widget' first before the process names are searched. Again, this is a fast check, but it is unnecessary checking if you already know for sure that, 'widget' will only appear in a process name (and not a window's title). In order to not add more user interface clutter and to provide those of you looking to eek out every bit of performance you can get, VoiceAttack now has character prefixes to limit searches to just the window title, process or class name.
（ユーザーインターフェイス）、および目的のターゲットを見つける際のユーザー支援（多くの場合、名前の重複があります）。ほとんどの場合、これはかなり速く動作しますが、状況によっては処理が不必要に過剰になる可能性があります。たとえば、「widget」を含むプロセス名を探している場合、プロセス名が検索される前に「widget」が最初にすべてのウィンドウタイトルが検索されます。繰り返しますが、これは高速なチェックですが、「ウィジェット」はプロセス名にのみ表示され（ウィンドウのタイトルには表示されない）ことが確実にわかっている場合は不要なチェックです。ユーザーインターフェースの乱雑さを追加せず、取得できるすべてのパフォーマンスを探しているユーザーに提供するために、VoiceAttackには、検索をウィンドウタイトル、プロセス、またはクラス名のみに制限する文字プレフィックスがあります。

If '^' is prepended to the target value, the target search is limited to only window titles. If '~' is prepended, only process names are searched.
' ^ 'がターゲット値の前に追加された場合、ターゲット検索はウィンドウタイトルのみに制限されます。「～」が先頭に追加された場合、プロセス名のみが検索されます。

If '+' is prepended, only class names are searched.
' + 'が先頭に追加された場合、クラス名のみが検索されます。

These new prefixes can also be used in conjunction with wildcards.
これらの新しいプレフィックスは、ワイルドカードと組み合わせて使用することもできます。

For example, '^*notepad*' will indicate to only search window titles that contain, 'notepad'. If, ‘notepad’ is not found within a window title, the search will stop and not continue on to search processes or class names. ‘~widget*’ indicates that the search is to only look for process names that start with, ‘widget’. That means that no window titles or class names are searched. ‘+foo’ indicates that the search should only be for window class names that match, ‘foo’ (again, window titles and process names are not searched first which saves some time).
たとえば、「^ * notepad *」は、「notepad」を含む検索ウィンドウタイトルのみを示します。ウィンドウタイトル内に「notepad」が見つからない場合、検索は停止し、検索プロセスまたはクラス名に進みません。「～widget *」は、「widget」で始まるプロセス名のみを検索することを示します。つまり、ウィンドウのタイトルやクラス名は検索されません。「+ foo」は、一致するウィンドウクラス名のみを検索する必要があることを示します。「foo」（再び、ウィンドウタイトルとプロセス名は最初に検索されないため、時間を節約できます）。

'Default Text-to-speech voice' – Selecting a value from this list will allow you to indicate a voice to be used profile-wide when, 'Default' is selected in a, 'Say Something with Text-to-Speech' action. Also, an attempt is made at using this selected voice when something goes wrong with the selected voice in the, ‘'Say Something with Text- to-Speech' action. You can select a voice from the list, or you can freely type in this box. The value typed in must resolve to an active voice. This can be a text variable, tokens or various combinations of literal text and tokens. Note that selecting, ‘None’ indicates that the Windows default voice will continue to be used.
「デフォルトのテキスト読み上げ音声」-このリストから値を選択すると、「テキスト読み上げで何かを言う」アクションで「デフォルト」が選択されている場合に、プロファイル全体で使用される音声を指定できます。 。また、「テキスト読み上げで何かを言う」アクションで選択した音声に問題が発生した場合、この選択した音声を使用しようとします。リストから音声を選択するか、このボックスに自由に入力できます。入力する値は、アクティブな音声に解決される必要があります。これは、テキスト変数、トークン、またはリテラルテキストとトークンのさまざまな組み合わせです。[なし]を選択すると、Windowsのデフォルトの音声が引き続き使用されることを示します。

Profile Options Exec Tab
プロファイルオプションの[実行]タブ

'Execute a command each time a phrase is unrecognized' allows you to pick a command to run any time a phrase is not recognized. The selected command could be something as simple as playing a sound to calling a plugin function. Note that the '{CMD}' token takes on the unrecognized value so you can use it for processing (see the section on tokens near the end of this document).
' フレーズが認識されないたびにコマンドを実行する ' フレーズが認識されないときに実行するコマンドを選択できます。選択したコマンドは、サウンドを再生してプラグイン関数を呼び出すだけの簡単なものにすることができます。'{CMD}'トークンは認識されない値を受け取るため、処理に使用できることに注意してください（このドキュメントの終わり近くにあるトークンに関するセクションを参照してください）。

'Execute a command each time this profile is loaded' allows you to pick a command to execute when you switch to this profile (or when VoiceAttack is started and this profile is already selected). Again, your command can do simple stuff or it could initialize values for use later. If you need to disable this feature temporarily due to a problem (for example, loading another profile when your current profile loads... not a good idea), you can hold down CTRL and Shift when you launch VoiceAttack and
' このプロファイルが読み込まれるたびにコマンドを実行する'このプロファイルに切り替えたとき（またはVoiceAttackが開始され、このプロファイルが既に選択されているとき）に実行するコマンドを選択できます。繰り返しになりますが、コマンドは単純なことを行うことも、値を初期化して後で使用することもできます。問題のためにこの機能を一時的に無効にする必要がある場合（たとえば、現在のプロファイルが読み込まれたときに別のプロファイルを読み込むことはお勧めできません）、VoiceAttackを起動してCtrl

choose the option, 'Disable profile initialization commands (this session only)' and click, 'OK'.
「プロファイル初期化コマンドを無効にする（このセッションのみ）」オプションを選択し、「OK」をクリックします。

'Execute a command each time this profile is unloaded' allows you to pick a command that can be specified to execute immediately before the current, active profile is unloaded. A profile is unloaded when another profile is selected, or, when VoiceAttack is shutting down. Note that the profile that is selected to be loaded next will wait until the specified unload command completes, and any startup command in the selected profile will execute after the specified unload command. Note also that VA's shutdown will be delayed until the specified unload command completes executing. Tip: You can distinguish between the two types of unloading by checking the value of the {CMDACTION} token (see the section on tokens, later in this document). Tip: You can get some information about the profile that will load up after the current one unloads by checking out the, 'NEXTPROFILE' set of tokens (also later in this document).
' このプロファイルがアンロードされるたびにコマンドを実行する ' 現在のアクティブなプロファイルがアンロードされる直前に実行するように指定できるコマンドを選択できます。プロファイルは、別のプロファイルが選択されたとき、またはVoiceAttackがシャットダウンしたときにアンロードされます。指定されたアンロードコマンドが完了するまで次のロードされるように選択されたプロファイルが待機することに注意してください、そして選択したプロファイルのいずれかの起動コマンドが実行されます後指定されたアンロードコマンド。また、VAのシャットダウンは、指定されたアンロードコマンドの実行が完了するまで遅延されることに注意してください。ヒント：{CMDACTION}トークンの値を確認することにより、2種類のアンロードを区別できます（このドキュメントの後半のトークンに関するセクションを参照）。ヒント：トークンの「NEXTPROFILE」セットをチェックアウトすることで、現在のプロファイルがアンロードされた後にロードされるプロファイルに関する情報を取得できます（このドキュメントの後半でも）。

'Execute a command each time a dictation phrase is recognized' allows you to choose a command that is run any time an entry is made into the dictation buffer. This could be useful for playing a sound, reading back what was last said or for plugin use.
' ディクテーションフレーズが認識されるたびにコマンドを実行 ' ' ディクテーションバッファにエントリが作成されるたびに実行されるコマンドを選択できます。これは、サウンドの再生、最後に言われた内容の読み戻し、またはプラグインの使用に役立ちます。

Profile Options Hotkey Tab
プロファイルオプションの[ホットキー]タブ

Each of the items on this tab override what is available on the Options > Hotkeys screen. Select each item that you want to override by checking the appropriate box. Where a, '…' button is indicated, you can click it and be presented with a configuration screen (the base configuration is outlined in the Options screen). For example, if the Recognition Global Hotkey is, 'Ctrl + F5' on the Options screen (as a global setting), and we would like to have this particular profile use 'Alt + F1', we can override that value here. That means that for every other profile, you would press, 'Ctrl + F5' to toggle VoiceAttack's listening, but, when you are in this profile, you would use, 'Alt + F1'. This same principle is carried over into the, 'Mouse Click Recognition', 'Stop Command Hotkey' and the, 'Joystick Recognition Button' (See the Options page (Hotkeys tab) for more on what these features do).
このタブの各項目は、[オプション]> [ホットキー]画面で使用可能なものよりも優先されます。適切なボックスをオンにして、オーバーライドする各アイテムを選択します。「…」ボタンが示されている場合は、クリックして構成画面を表示できます(基本構成は「オプション」画面で説明されています)。たとえば、認識グローバルホットキーが(グローバル設定として)オプション画面で 'Ctrl + F5'であり、この特定のプロファイルで 'Alt + F1'を使用する場合、ここでその値をオーバーライドできます。つまり、他のすべてのプロファイルでは、「Ctrl + F5」を押してVoiceAttackのリスニングを切り替えますが、このプロファイルでは「Alt + F1」を使用します。この同じ原則は、「マウスクリック認識」、「コマンドホットキーの停止」、および「

Profile Options Advanced Tab
プロファイルオプションの[詳細設定]タブ

The 'Block potentially harmful profile actions' option is useful for allowing you to inspect your profile before you put it into service. What this does is it works to prevent certain elements of a profile and its commands/actions from executing either intentionally or not intentionally. When this option is selected, the following elements are blocked from occurring:
「ブロック潜在的に有害なプロファイルのアクション」オプションでは、サービスにそれを置く前に、あなたのプロフィールを検査することを可能にするための便利です。これが行うことは、プロファイルの特定の要素とそのコマンド/アクションが意図的にまたは意図的に実行されないようにすることです。このオプションを選択すると、次の要素の発生がブロックされます。

· Profile switch actions (action not executed)
・ プロファイル切り替えアクション(アクションは実行されません)

· Profile startup command execution (command not executed)
・ プロファイルスタートアップコマンドの実行(コマンドは実行されません)

· Profile unload command execution (command not executed)
・ プロファイルアンロードコマンドの実行(コマンドは実行されません)

· Commands locked by author flag (command not executed)
・作成 者フラグによってロックされたコマンド(実行されないコマンド)

· Run an application actions (action not executed)
· アプリケーションアクションを実行する（実行されないアクション）

· Stop process actions (action not executed)
· プロセスアクションの停止（アクションは実行されません）

· Inline function actions (action not executed)
· インライン関数アクション（アクションは実行されません）

· Plugin execution actions (action not executed)
· プラグイン実行アクション（アクションは実行されません）

· Plugin ExecuteCommand() function (command not executed)
· プラグインExecuteCommand（）関数（コマンドは実行されません）

· Unrecognized catch-all command execution (command not executed)
· 認識されないキャッチオールコマンドの実行（コマンドは実行されません）

· Dictation recognized command execution (command not executed)
· ディクテーション認識コマンドの実行（コマンドは実行されません）

· Commands executed by command line (-command) (command not executed)
· コマンドラインで実行されるコマンド（-command）（実行されないコマンド）

A message in the log will appear if any of the events above are invoked while this flag is set. Once you are satisfied with the contents of your profile, simply deselect this option.
このフラグが設定されている間に上記のイベントのいずれかが呼び出されると、ログにメッセージが表示されます。プロファイルの内容に満足したら、このオプションを選択解除します。

NOTE: If you would like all profiles that are subsequently imported to have this option turned on by default, simply go to the Options screen and then to the, System/Advanced tab. On that screen you will find the option labeled, 'Upon import, profiles will have, 'Block potentially harmful profile actions' selected'. Choose that option and each profile that is imported will have the, 'Block potentially harmful profile actions' automatically set.
注：後でインポートされるすべてのプロファイルでデフォルトでこのオプションを有効にするには、[オプション]画面に移動し、[システム/詳細設定]タブに移動します。その画面には、「インポート時、プロファイルには「有害なプロファイルアクションをブロックする」が選択されています」というラベルのオプションがあります。そのオプションを選択すると、インポートされる各プロファイルには、「潜在的に有害なプロファイルアクションをブロックする」が自動的に設定されます。

Now, back to the profile screen... 3 - New Command button
次に、プロファイル画面に戻ります... 3-新しいコマンドボタン

Click this button to add a new command to your profile (See 'Command Screen').
このボタンをクリックして、新しいコマンドをプロファイルに追加します（「コマンド画面」を参照）。

Edit Command button
[コマンドの編集]ボタン

Click to edit the currently selected command (same as pressing the Enter button on your keyboard or double-clicking a command in the command list) (See 'Command Screen').
クリックして、現在選択されているコマンドを編集します（キーボードのEnterボタンを押すか、コマンドリストのコマンドをダブルクリックするのと同じです）（「コマンド画面」を参照）。

Delete Command button
コマンド削除ボタン

Click to delete the currently selected command (same as pressing the Delete button on your keyboard).
クリックして、現在選択されているコマンドを削除します（キーボードの[削除]ボタンを押すのと同じ）。

1. - Command list
1.- コマンドリスト

This list shows all the commands that have been added by you for the currently selected profile. The first column shows the command name (the words that you will say into your microphone). In screenshot, the second column shows the category
このリストには、現在選択されているプロファイルに対して追加されたすべてのコマンドが表示されます。最初の列には、コマンド名（マイクに向かって言う言葉）が表示されます。スクリーンショットでは、2番目の列にカテゴリが表示されます

of your command (depending on your command list, you may also see the description and shortcut columns), and, the third column shows the actions that will be performed when VoiceAttack recognizes the command. For example, on the first line, the command, 'bags' is indicated. If you say the word, 'bags' into your microphone, VoiceAttack will press the, 'Left Shift' key, plus the, 'B' key. You can double-click a command in this list to edit the command. Right-clicking on this list will allow you to add, edit and delete commands. You can also copy and paste commands as well as copy commands to completely different profiles.
コマンドの（コマンドリストによっては、説明とショートカットの列も表示される場合があります）、3番目の列には、VoiceAttackがコマンドを認識したときに実行されるアクションが表示されます。たとえば、最初の行には、コマンド「bags」が示されています。マイクに「バッグ」という言葉を言うと、VoiceAttackは「左シフト」キーと「B」キーを押します。このリスト内のコマンドをダブルクリックして、コマンドを編集できます。このリストを右クリックすると、コマンドを追加、編集、削除できます。コマンドをコピーして貼り付けたり、コマンドを完全に異なるプロファイルにコピーしたりすることもできます。

2. - Import Commands button
2 .-[コマンドのインポート]ボタン

Click this button to selectively import commands from previously-saved profiles (See, 'Importing Commands').
このボタンをクリックして、以前に保存したプロファイルからコマンドを選択的にインポートします（「コマンドのインポート」を参照）。

3. - List filter buttons
3.- リストフィルターボタン

Expand/collapse multipart commands - This button toggles the view to show multipart commands (commands that have names separated by a semicolon) as one row or multiple rows.
マルチパートコマンドの展開/折りたたみ- このボタンはビューを切り替えて、マルチパートコマンド（セミコロンで区切られた名前を持つコマンド）を1行または複数行として表示します。

Toggle Category Grouping - Use this button to group by category. Grouping by category will let you show/hide groups of commands that have the same category. Note that when the list is grouped by category, you can click on the group headers to expand or collapse that group. You can expand/collapse all groups If you hold down the CTRL key while expanding or collapsing a group. Renaming categories for multiple commands can be done by right-clicking on the group header and selecting, 'Rename Category'.
カテゴリのグループ化の切り替え-このボタンを使用して、カテゴリ別にグループ化します。カテゴリでグループ化すると、同じカテゴリを持つコマンドのグループを表示/非表示できます。リストがカテゴリ別にグループ化されている場合、グループヘッダーをクリックして、そのグループを展開または折りたたむことができます。Ctrlキーを押しながらグループを展開または折りたたむと、すべてのグループを展開または折りたたむことができます。複数のコマンドのカテゴリの名前を変更するには、グループヘッダーを右クリックして[カテゴリの名前を変更]を選択します。

In this area, you will also see an indicator that shows how many commands are in your profile. If you have filters applied, you will see how many commands are available, plus how many are displayed. The tool tip you see when you hover over this information will display the number of derived commands that are created by things like dynamic commands or composite (prefix/suffix) commands.
この領域には、プロファイルに含まれるコマンドの数を示すインジケーターも表示されます。フィルターが適用されている場合、使用可能なコマンドの数と表示されているコマンドの数が表示されます。この情報にカーソルを合わせると表示されるツールヒントには、動的コマンドや複合（プレフィックス/サフィックス）コマンドなどによって作成された派生コマンドの数が表示されます。

Stop Commands Button - Clicking this button will stop any running commands (this works exactly like the, 'Stop All Commands' button on the Main screen).
[コマンドの停止]ボタン-このボタンをクリックすると、実行中のコマンドがすべて停止します（メイン画面の[すべてのコマンドを停止]ボタンとまったく同じように機能します）。

List Filter Input Box - Start typing in this box and the command list will be filtered down to display any field that contains the text that is typed. Clear this box to remove the filter (this does not affect the underlying data).
リストフィルター入力ボックス-このボックスへの入力を開始すると、コマンドリストがフィルター処理され、入力されたテキストを含むフィールドが表示されます。フィルターを削除するには、このボックスをクリアします（これは基礎となるデータには影響しません）。

List Filter Toggle Buttons – There are six buttons to quickly filter the list based on command state. You can toggle each filter by simply clicking the buttons (this does not affect the underlying data). The six filters are:
リストフィルタートグルボタン–コマンドの状態に基づいてリストをすばやくフィルターする6つのボタンがあります。ボタンをクリックするだけで各フィルターを切り替えることができます（これは基になるデータには影響しません）。6つのフィルターは次のとおりです。

- Hide/show commands that have 'When I say' disabled. (S)
- [発言時]が無効になっているコマンドを非表示/表示する。(S)

- Hide/show commands that have 'When I press keys' disabled. (K)
- [キーを押したとき]が無効になっているコマンドの非表示/表示。(K)

- Hide/show commands that have 'When I press button' disabled. (J)
- [ボタンを押したとき]が無効になっているコマンドの非表示/表示。(J)

- Hide/show commands that are prefixes. (P)
- プレフィックスであるコマンドの非表示/表示。(P)

- Hide/show commands that are suffixes. (X)
- サフィックスであるコマンドの非表示/表示。(バツ)

- Hide/show commands that are full commands. (F)
- 完全なコマンドであるコマンドの非表示/表示。(F)

Done button
完了ボタン

No commands will be saved unless you hit the Done button. NOTE: New and edited commands ARE NOT AVAILABLE to the speech recognition engine until you press 'Done' (or, 'Apply'). There have been a lot of times when I have entered a new command on the Profile screen and was puzzled by why it wasn't working when I would speak. It's because I never clicked the, 'Done' button. Happens a lot.
[完了]ボタンを押さない限り、コマンドは保存されません。注：新規および編集されたコマンドは、[完了]（または[適用]）を押すまで音声認識エンジンで使用できません。プロファイル画面で新しいコマンドを入力し、話すときに機能しない理由に困惑したことが何度もありました。「完了」ボタンをクリックしたことがないためです。たくさん起こります。

Seriously... I am that dense :)
真剣に...私はその密度が高いです:)

Apply button
適用ボタン

Works just like the, 'Done' button, but does not close the screen.
「完了」ボタンと同じように機能しますが、画面は閉じません。

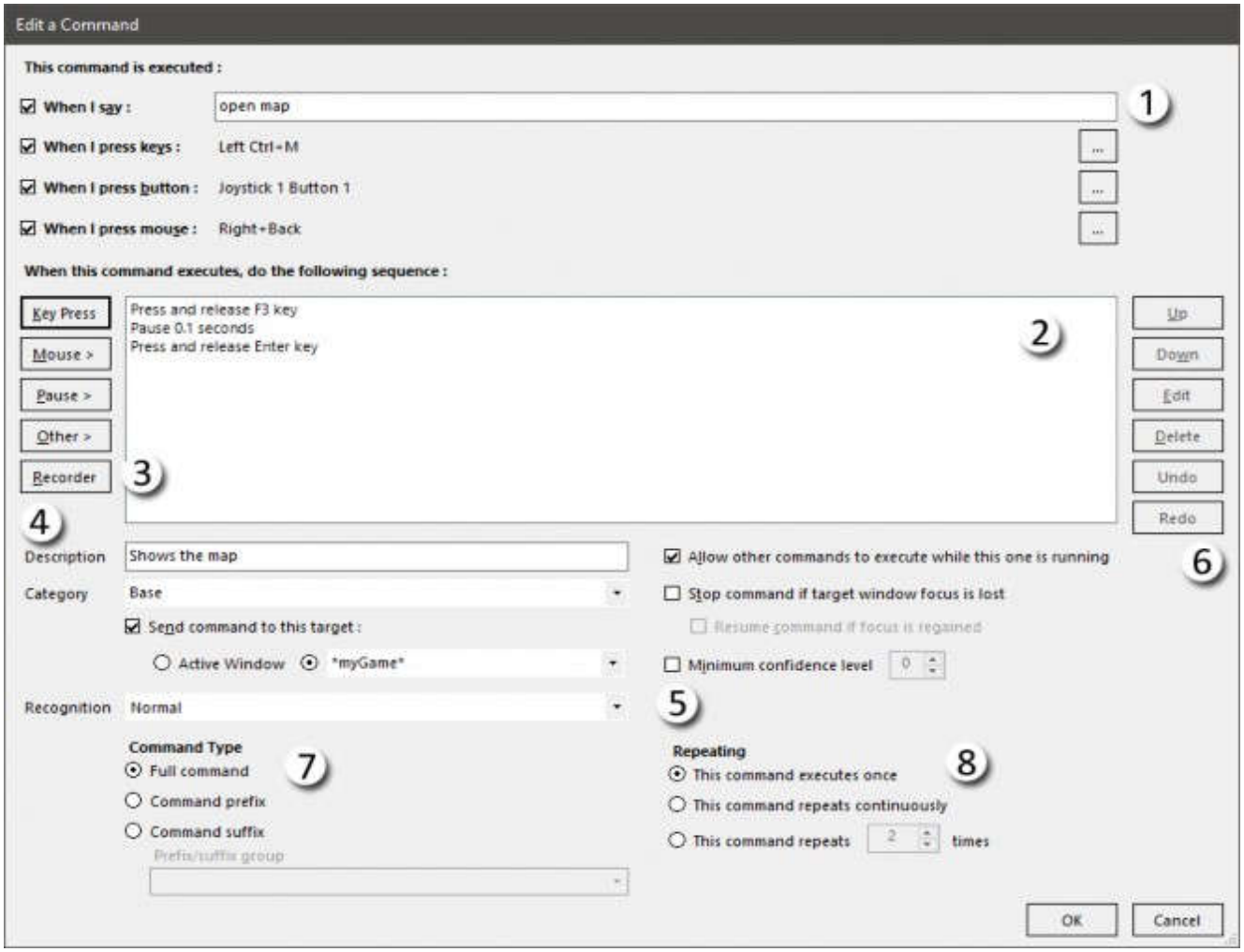
Cancel button
キャンセルボタン

All changes to commands for this profile can be canceled by hitting the Cancel button.
このプロファイルのコマンドに対するすべての変更は、キャンセルボタンを押すことでキャンセルできます。

Command Screen
コマンド画面

A VoiceAttack command is basically a macro that performs a series of actions. A command can be executed with a spoken word or phrase, with the press of a keyboard shortcut or the press of joystick or mouse buttons. Simple commands can be run one at a time, while lengthy commands can be configured to run in the background (asynchronously).
VoiceAttackコマンドは、基本的に一連のアクションを実行するマクロです。コマンドは、キーボードショートカットを押すか、ジョイスティックまたはマウスボタンを押すことで、話し言葉またはフレーズで実行できます。単純なコマンドは一度に1つずつ実行できますが、長いコマンドはバックグラウンドで（非同期に）実行するように構成できます。

The Command Screen is where you will add and edit commands and their actions that execute when VoiceAttack recognizes your spoken phrase or detects your keyboard shortcut / joystick button press. For instance, you can add a command called, 'Help' and then actions that press and release the 'F1' key in your application.
コマンド画面では、VoiceAttackが音声フレーズを認識したとき、またはキーボードショートカット/ジョイスティックボタンの押下を検出したときに実行されるコマンドとそのアクションを追加および編集します。たとえば、「Help」というコマンドを追加してから、アプリケーションで「F1」キーを押して放すアクションを追加できます。



In the above example, this command can be executed in four different ways. The first way is to speak the phrase, 'open map' into the microphone. The second is to press Ctrl + M on the keyboard. The third way is by pressing button 2 on joystick 1, and the fourth way is to press the right and back mouse buttons at the same time. VoiceAttack will react by sending the 'F3' key to your application, pausing briefly, and then send the 'Enter' key (your keen eye may have noticed that this command will run in the background (option 5). More about that below). There is a lot going on here,
上記の例では、このコマンドは4つの異なる方法で実行できます。最初の方法は、「地図を開く」というフレーズをマイクに向かって話すことです。2つ目は、キーボードのCtrl + Mを押すことです。3番目の方法はジョイスティック1のボタン2を押すことで、4番目の方法はマウスの右ボタンと戻るボタンを同時に押すことです。VoiceAttackは、「F3」キーをアプリケーションに送信して一時停止し、「Enter」キーを送信することで反応します（このコマンドがバックグラウンドで実行されることに気付いているかもしれません（オプション5）。詳細は以下を参照）。 。ここでは多くのことが行われていますが、

but we'll go through each numbered section, one at a time.
しかし、各番号付きセクションを1つずつ確認します。

1. – Command Input
1.– コマンド入力

The command input section up near the top is where you'll indicate how your command will be executed by user interaction. This can be with a spoken phrase, a keyboard hotkey/shortcut, a joystick button press, or by a particular mouse button click.
上部近くのコマンド入力セクションでは、ユーザーの操作によってコマンドがどのように実行されるかを示します。これは、話し言葉、キーボードのホットキー/ショートカット、ジョイスティックボタンの押下、または特定のマウスボタンのクリックによるものです。

Checking the box labeled, 'When I say...' indicates to VoiceAttack that you want this command to be executed by speaking a word or phrase. In the example, we want to say the phrase, 'open map' into the microphone to get VoiceAttack to react.
「When I say ...」というラベルの付いたボックスをチェックすると、VoiceAttackに対して、単語またはフレーズを話すことによってこのコマンドを実行することを示します。この例では、「マップを開く」というフレーズをマイクに発声して、VoiceAttackが反応するようにします。

Note that you *must* fill out the input box if this option is checked, and, what you put in the input box must be unique for the selected profile (in this case, you can only have one command with, 'open map' as the spoken phrase).
このオプションがチェックされている場合は入力ボックスに入力する必要があり、入力ボックスに入力するものは選択したプロファイルに対して一意である必要があることに注意してください（この場合、「マップを開く」で使用できるコマンドは1つだけです）話し言葉として）。

NOTE - VoiceAttack supports multiple phrases in the input box by separating the phrases with a semicolon ; For example, if you have three commands that do the same thing: Fire, Open Fire and Fire Weapons, instead of adding three separate commands, you can add one command like this: 'Fire;Open Fire;Fire Weapons' and VoiceAttack will execute the commands all the same way.
注-VoiceAttackは、セミコロンでフレーズを区切ることにより、入力ボックスで複数のフレーズをサポートしています。たとえば、同じことを行う3つのコマンドがある場合：Fire、Open Fire、Fire Weapons、3つの個別のコマンドを追加する代わりに、次のような1つのコマンドを追加できます： 'Fire; Open Fire; Fire Weapons'とVoiceAttackが実行されますコマンドはすべて同じ方法です。

Dynamic command sections
動的コマンドセクション

Dynamic sections allow you to specify a part of your command that may vary. Sometimes you may want to say, 'Hello computer' and sometimes you may want to say, 'Greetings computer' and execute the same command. To indicate that you want to use a dynamic section, enclose the section in square brackets: [ ], with each element separated by a semicolon. Your command may look something like this:
動的セクションを使用すると、コマンドのさまざまな部分を指定できます。「Hello computer」と言いたいこともあれば、「Greetings computer」と言って同じコマンドを実行したいこともあります。動的セクションを使用することを示すには、セクションを角括弧[]で囲み、各要素をセミコロンで区切ります。コマンドは次のようになります。

[Hello;Greetings]computer
[Hello; Greetings] computer

In this case, you can say, 'Hello computer' and 'Greetings computer' and the command will be executed.
この場合、「Hello computer」および「Greetings computer」と言うことができ、コマンドが実行されます。

Note that multipart commands are also still separated with a semicolon (as demonstrated by adding, 'Hi' to the end):
マルチパートコマンドもセミコロンで区切られていることに注意してください（末尾に「Hi」を追加することで示されるように）。

[Greetings;Hello]computer;Hi
[あいさつ;こんにちは]コンピュータ;こんにちは

With this example, to execute the command, you can say: Greetings computer
この例では、コマンドを実行するために、次のように言うことができます。

Hello computer Hi
こんにちはコンピューターこんにちは

The dynamic sections don't have to just be at the beginning. They can be anywhere in the command. Also, as a side-effect, if you put a semicolon on at the end of the selections, it makes the section optional:
動的セクションは最初にある必要はありません。コマンド内の任意の場所に配置できます。また、副作用として、選択の最後にセミコロンを付けると、セクションがオプションになります:

[Greetings;Hello]computer[how are you;]
[あいさつ;こんにちは]コンピュータ[お元気ですか。]

You can say the following to execute the command: Greetings computer how are you
コマンドを実行するには、次のように言うことができます。

Hello computer how are you Greetings computer
こんにちは、お元気ですか。

Hello computer
こんにちはコンピューター

Note that there is a semicolon after 'how are you' to indicate that the entire section is optional.
「how are you」の後にセミコロンがあり、セクション全体がオプションであることを示すことに注意してください。

Something to consider when using this feature is that you can create a lot of permutations from very few words. Use with care :)
この機能を使用する際に考慮すべきことは、ごくわずかな単語から多くの順列を作成できることです。注意して使用してください:)

Dynamic command sections can also contain numeric ranges. This is kind of an advanced feature that is not very useful on its own, but when used in conjunction with other features (such as the {TXTNUM} token (see, 'Tokens' section) and Quick Input), it can be used to consolidate large numbers of commands into just a few.
動的コマンドセクションには数値範囲を含めることもできます。これは一種の高度な機能であり、単独ではあまり有用ではありませんが、他の機能({TXTNUM}トークン(「トークン」セクションを参照)やクイック入力など)と組み合わせて使用すると、多数のコマンドを少数に統合します。

To indicate a numeric range in a dynamic section, just include the minimum and maximum values separated by an ellipsis (..). For example, let's say you have 100 racers in a race game. Instead of creating 100 separate commands to eject racers, you can have a single command, 'eject car [1..100]'. With this example, you can say, 'eject car 1', 'eject car 2', 'eject car 99', etc.
動的セクションの数値範囲を示すには、省略記号(..)で区切られた最小値と最大値を含めるだけです。たとえば、レースゲームに100人のレーサーがいるとします。100個の個別のコマンドを作成してレーサーを排出する代わりに、「eject car [1..100]」という単一のコマンドを使用できます。この例では、「eject car 1」、「eject car 2」、「eject car 99」などと言うことができます。

An additional option for numeric ranges for dynamic command sections is a multiplier. This will allow you to, 'step' the numbers in your range by a specified value. To indicate a multiplier, just include the value with your range like this: [1..5,10]. Just like above, that's the minimum value and maximum value separated by an ellipsis, then a comma, then the multiplier value. [1..5,10] will yield 10, 20, 30, 40, 50. [5..10,20] will
動的コマンドセクションの数値範囲の追加オプションは、乗数です。これにより、指定した値で範囲内の数値を「ステップ」できます。乗数を示すには、[1..5,10]のように範囲に値を含めるだけです。上記と同様に、これは、省略記号、コンマ、乗数の値で区切られた最小値と最大値です。[1..5,10]は10、20、30、40、50を生成します。[5..10,20]は

yield 100, 120, 140, 160, 180, 200.
100、120、140、160、180、200を生成します。

Advanced: If you would like to retrieve the individual portions of a command that contains dynamic command sections, see the, '{CMDSEGMENT:}' token later in this document.
高度:動的コマンドセクションを含むコマンドの個々の部分を取得する場合は、このドキュメントで後述する「{CMDSEGMENT:}」トークンを参照してください。

Wildcards
ワイルドカード

There is a somewhat unsupported* feature in VoiceAttack's 'When I say' feature. You can use, 'wildcards' around the phrases to indicate, 'contains', 'starts with' and 'ends with'.
VoiceAttackの「When I say」機能には、サポートされていない* 機能があります。「含む」、「で始まる」、「で終わる」を示すために、フレーズの周りに「ワイルドカード」を使用できます。

So, let's say you have a spoken phrase 'attack'. Let's also say that you want to execute your command if the word, 'attack' is included in any spoken phrase. To indicate to VoiceAttack that you want the 'attack' command to execute any time it is contained in a phrase, you simply put asterisks around the phrase, like so: *attack*.
それで、「攻撃」という言葉が話されたとしましょう。また、「attack」という単語が話し言葉に含まれている場合にコマンドを実行するとします。VoiceAttackに、フレーズに含まれるときに「attack」コマンドを実行するように指示するには、*attack *のように、フレーズをアスタリスクで囲むだけです。

If you want to indicate that you want the 'attack' command to execute if the spoken
発声された場合に「攻撃」コマンドを実行することを示す場合

phrase starts with the word, 'attack', just put an asterisk at the end, like so: attack*. This way, you can say, 'attack the enemy' and VoiceAttack will execute the 'attack' command. If you say, 'I would like to attack the enemy', the 'attack' command will not be executed, since the word, 'attack' is not at the start of the phrase.
フレーズは「attack」という単語で始まり、最後にアスタリスクを付けるだけです。例：attack *。このように、「敵を攻撃する」と言うことができ、VoiceAttackは「攻撃」コマンドを実行します。「敵を攻撃したい」と言うと、「attack」という単語はフレーズの先頭にないため、「attack」コマンドは実行されません。

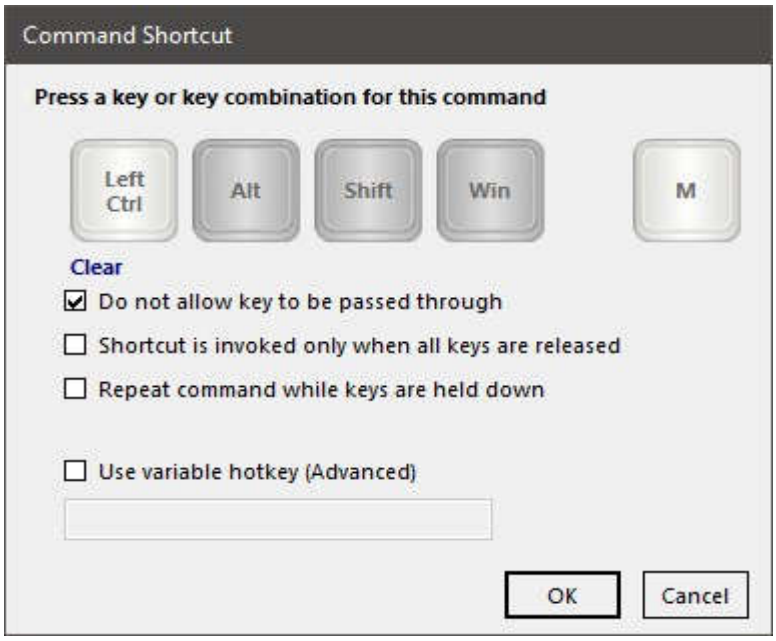
On a similar note, if you only want, 'attack' to be executed if the word, 'attack' is at the end of the phrase, put the asterisk at the beginning, like so: *attack.
同様に、「attack」という単語がフレーズの最後にある場合にのみ「attack」を実行する場合は、* attackのように先頭にアスタリスクを付けます。

VoiceAttack will execute all of the commands in which the wildcards apply. So, if you have '*rocket*' and '*ship*' and '*attack*' as commands, and you happen to say, 'I would like to attack the ship with my rockets', VoiceAttack will attempt to execute 'attack', then, 'ship' and then 'rocket' in that order (the order in which they are spoken, but due to the asynchronous nature of this type of situation, the order cannot be guaranteed (use with caution).
VoiceAttackは、ワイルドカードが適用されるすべてのコマンドを実行します。そのため、コマンドとして「* rocket *」と「* ship *」と「* attack *」があり、「ロケットで船を攻撃したい」と言うと、VoiceAttackは「攻撃」、「船」、そして「ロケット」の順序（発話の順序ですが、この種の状況の非同期性のため、順序は保証できません（注意して使用してください）。

Commands will not repeat with wildcards. If you have commands, '*rocket*' and, '*ship*' and, '*rocket ship*' and you say, 'I want to take a ride in my rocket ship', VoiceAttack will execute the command, 'rocket ship' and not 'rocket' and 'ship'. Also, if you say, 'rocket rocket rocket rocket rocket rocket rocket rocket rocket', the 'rocket' command will only be executed once.
コマンドはワイルドカードでは繰り返されません。「* rocket *」と「* ship *」と「* rocket ship *」というコマンドがあり、「ロケット船に乗ってみたい」と言うと、VoiceAttackはコマンド「rocket」を実行します「ロケット」や「船」ではなく「船」。また、「ロケットロケットロケットロケットロケットロケットロケットロケットロケット」と言うと、「ロケット」コマンドは1回だけ実行されます。

* The reason it is 'somewhat unsupported' is basically because it is not a terribly reliable feature and was added as an attempt to give a little bit more flexibility, especially in the areas of immersion. Your mileage may vary. Good luck, captain!
* 「ややサポートされていない」理由は、基本的にそれはひどく信頼できる機能ではなく、特に没入の分野でもう少し柔軟性を与えるために追加されたためです。あなたのマイレージは異なる場合があります。頑張って、キャプテン！

Checking the box labeled, 'When I press keys' indicates to VoiceAttack that you want to execute this command when pressing a keyboard shortcut. In this example, the keyboard shortcut is Left Ctrl + M. You can assign the keyboard shortcut by clicking on the '…' button to the right. You will be presented with the screen below:
「キーを押すとき」というラベルの付いたボックスをチェックすると、キーボードショートカットを押したときにこのコマンドを実行することをVoiceAttackに示します。この例では、キーボードショートカットは左Ctrl + Mです。右側の[…]ボタンをクリックして、キーボードショートカットを割り当てることができます。以下の画面が表示されます。



This is the, 'Command Shortcut' screen. It allows you to choose the key / key combo to assign to your macro.
これは、「コマンドショートカット」画面です。マクロに割り当てるキー/キーコンボを選択できます。

The, 'Do not allow key to be passed through' option prevents the main key (non- modifier) from being passed through to the application. For example, if your hotkey is F1 and this option is selected, VoiceAttack will respond to the F1 key press and then prevent any other application from receiving this key press (for this example, if F1 is being handled by VoiceAttack, you will not be able to use the F1 key in other applications while VoiceAttack is running. If you rely on F1 to bring up, 'Help', then, you'll have to pick another key).
「キーの通過を許可しない」オプションを使用すると、メインキー(非修飾子)がアプリケーションに渡されなくなります。たとえば、ホットキーがF1でこのオプションが選択されている場合、VoiceAttackはF1キーの押下に応答し、他のアプリケーションがこのキーの押下を受け取らないようにします(この例では、F1がVoiceAttackによって処理されている場合、VoiceAttackの実行中に他のアプリケーションでF1キーを使用できます。F1を使用して「ヘルプ」を表示する場合は、別のキーを選択する必要があります。

The, 'Shortcut is invoked only when all keys are released' option allows you to indicate that the macro will only execute once all keys in the combo are up. This allows you a greater level of flexibility and control (such as having a macro that executes on key down and a separate macro that occurs only on key up). Also, this is the way that VoiceAttack can keep hotkey shortcuts from stepping on each other when some of the keys are involved in different commands. For instance, if there is a command that is executed by pressing, 'ALT + X', and another command that executes by pressing, 'X', setting both shortcuts to work on key release will keep both from executing at the same time.
「すべてのキーがリリースされたときにのみショートカットが呼び出される」オプションを使用すると、コンボ内のすべてのキーが起動したときにのみマクロが実行されることを示すことができます。これにより、より高いレベルの柔軟性と制御が可能になります（キーが押されたときに実行されるマクロや、キーが押されたときにのみ発生する別個のマクロなど）。また、これは、一部のキーが異なるコマンドに関係している場合に、VoiceAttackがホットキーショートカットが互いに踏まないようにする方法です。たとえば、「ALT + X」を押して実行されるコマンドと「X」を押して実行する別のコマンドがある場合、両方のショートカットをキーリリースで機能するように設定すると、両方が同時に実行されなくなります。

The 'Repeat command while keys are held down' option will allow the invoked command to continuously repeat for as long as the selected keyboard keys are held down.
「キーを押しながらコマンドを繰り返す」オプションを使用すると、選択したキーボードのキーを押している間、呼び出されたコマンドを継続的に繰り返すことができます。

The 'Use variable hotkey (Advanced)' option is provided for you to be able to use a hotkey that is indicated by the contents of a text variable. This is so that a hotkey/hotkey combo can be assigned to your command that is not known until the profile is actually loaded or is running. To make this feature work properly for you, there are a few items to keep in mind. To turn on the variable hotkey for a command,
「使用変数ホットキー（上級）テキスト変数の内容によって示されているホットキーを使用できるようにするために」オプションが提供されています。これにより、プロファイルが実際にロードされるか実行されるまでわからないコマンドにホットキー/ホットキーのコンボを割り当てることができます。この機能を適切に機能させるには、いくつかの留意事項があります。コマンドの変数ホットキーをオンにするには、

make sure the box is checked and simply put the name of the text variable to use in the provided input box. The previously-set text variable must contain properly- notated text in order to work. The good news is that the notation is (almost) exactly the same as what you will find in the Quick Input action and for variable keypresses. So, for instance, if your desired hotkey (for now) is ALT + L, set a text variable's value to '[ALT]L' (no quotes).
ボックスがオンになっていることを確認し、使用するテキスト変数の名前を入力ボックスに入力するだけです。事前に設定されたテキスト変数は、正しく機能するために適切に表記されたテキストを含む必要があります。幸いなことに、表記法は（ほぼ）クイック入力アクションや変数キー押下の場合とまったく同じです。したがって、たとえば、目的のホットキー（今のところ）がALT + Lの場合、テキスト変数の値を '[ALT] L'（引用符なし）に設定します。

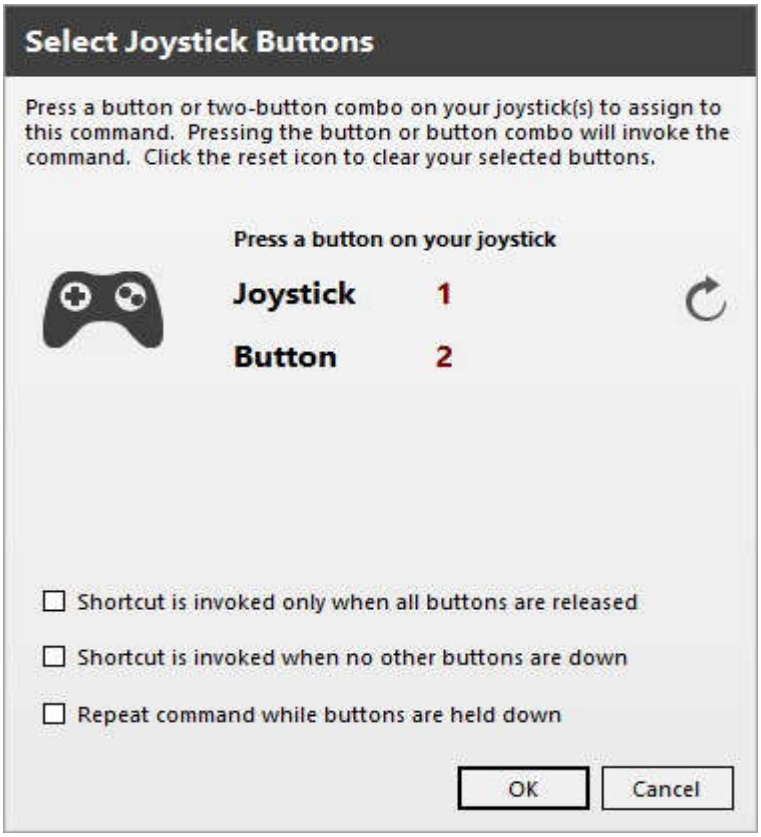
Note that the, 'L' does not have brackets. Keys with a single-character identifier (A-Z,
「L」には角括弧がありません。単一文字の識別子（AZ、

+, ß, ç, etc.) do not need brackets. Special keys, such as Enter, Shift, Alt, F12, etc. will require brackets (see the section titled, 'Quick Input, Variable Keypress and Hotkey Key Indicators' for all the possible key indicators). Note also that there is no space between [ALT] and L. Spaces are actually picked up as hotkeys here, so if there is a space, the space bar will be monitored for the given command.
+、ß、çなど）は括弧を必要としません。Enter、Shift、Alt、F12などの特殊キーにはブラケットが必要です（可能なすべてのキーインジケーターについては、「クイック入力、可変キープレス、ホットキーキーインジケーター」のセクションを参照してください）。また、[ALT]とLの間にスペースがないことに注意してください。スペースはここでホットキーとして実際に取得されるため、スペースがある場合は、指定されたコマンドのスペースバーが監視されます。

Something to understand is that the value in your text variable can change at any time, and the hotkey monitoring process that VoiceAttack uses is optimized so that you must explicitly refresh the hotkeys when a variable value is changed. In order to refresh the hotkeys that VoiceAttack is monitoring, you simply execute a,
理解すべきことは、テキスト変数の値はいつでも変更でき、VoiceAttackが使用するホットキー監視プロセスは最適化されているため、変数値が変更されたときにホットキーを明示的に更新する必要があることです。VoiceAttackが監視しているホットキーを更新するには、単に、

'Refresh Variable Hotkeys' action (see the section about this action later in this document). Note that only global and profile-scoped variables will work with this feature (command-scoped variables are inaccessible).
' 変数ホットキーの更新 'アクション(このドキュメントで後述するこのアクションに関するセクションを参照)。この機能で動作するのは、グローバルおよびプロファイルスコープの変数のみであることに注意してください(コマンドスコープの変数にはアクセスできません)。

Checking the box labeled, 'When I press button' indicates to VoiceAttack that you want to execute this command when pressing a joystick button, or a button combination. Note: Setting up joystick support is presented in more detail in the Options screen under the heading, 'Joystick Options'.) In this example, the selected button is the second button on joystick 1. Note that you can use up to two buttons to create a button combo (the buttons can even be on different sticks if you want). You can assign the joystick button for this command by clicking on the '…' button to the right. You will be presented with the screen below:
「ボタンを押すとき」というラベルの付いたボックスをチェックすると、ジョイスティックボタンまたはボタンの組み合わせを押したときにこのコマンドを実行することをVoiceAttackに示します。注:ジョイスティックサポートの設定については、「ジョイスティックオプション」という見出しの下のオプション画面で詳しく説明します。この例では、選択したボタンはジョイスティック1の2番目のボタンです。最大2つのボタンを使用して、ボタンコンボを作成します(必要に応じて、ボタンを別のスティックに配置することもできます)。右側の[…]ボタンをクリックして、このコマンドにジョイスティックボタンを割り当てることができます。以下の画面が表示されます。

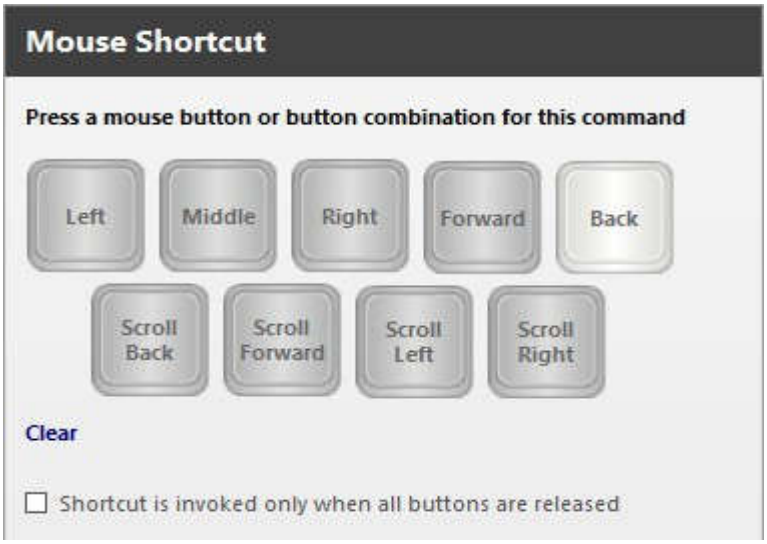


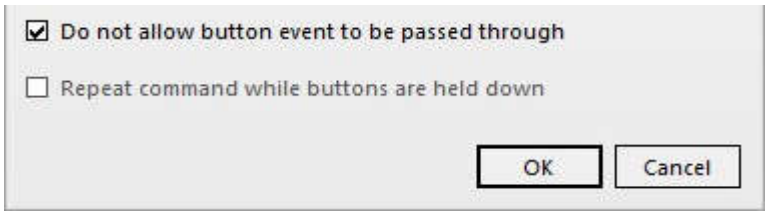
This is the 'Select Joystick Buttons' screen. If your joysticks (up to two joysticks are supported) are plugged in and set up, you can press a button and it will be detected here. To clear anything that you've done on this screen, click the, 'reset' button in the top-right. The option, 'Shortcut is invoked only when all buttons are released' will make the command only execute when all the buttons involved are let go. Note that any superseded shortcuts will not be executed when using this option. That means if you have a shortcut that works when the, 'A' button is released as well as a shortcut that works when, 'A + B' are released, only the shortcut for 'A + B' will be executed.
これが「ジョイスティックボタンの選択」画面です。ジョイスティック（最大2つのジョイスティックがサポートされています）が差し込まれてセットアップされている場合、ボタンを押すと、ここで検出されます。この画面で行ったことをクリアするには、右上の[リセット]ボタンをクリックします。オプション ' すべてのボタンがリリースされたときにのみショートカットが呼び出される ' は、関連するすべてのボタンが放されたときにのみコマンドを実行します。このオプションを使用すると、置き換えられたショートカットは実行されないことに注意してください。つまり、「A」ボタンを放したときに機能するショートカットと「A + B」を放したときに機能するショートカットがある場合、「A + B」のショートカットのみが実行されます。

The option, 'Shortcut is invoked when no other buttons are down' will prevent the command from executing if the buttons indicated are not exclusively involved. For example, if you have a command that is executed when button 'A' is pressed and this option is selected, if any other button is down when, 'A' is pressed, the command will not be executed.
オプション「他のボタンが押されていないときにショートカットが呼び出される」は、示されたボタンが排他的に含まれていない場合、コマンドの実行を妨げます。たとえば、ボタン「A」が押されてこのオプションが選択されたときに実行されるコマンドがある場合、「A」が押されたときに他のボタンがダウンしていると、コマンドは実行されません。

The 'Repeat command while buttons are held down' option will allow the invoked command to repeat for as long as the selected joystick buttons are held down.
[ボタンを押したままコマンドを繰り返す]オプションを使用すると、選択したジョイスティックボタンが押されている限り、呼び出されたコマンドを繰り返すことができます。

Checking the box labeled, 'When I press mouse' indicates to VoiceAttack that you want to execute this command when pressing a mouse button (or button combination) or scrolling with the mouse scroll wheel. In this example, the, 'Back' button is selected. Note that you can use any combination of the five standard mouse buttons, and any single scroll wheel event. You can assign the mouse button for this command by clicking on the '…' button to the right. You will be presented with the screen below:
「マウスを押すとき」というラベルの付いたボックスをオンにすると、マウスボタン（またはボタンの組み合わせ）を押すか、マウススクロールホイールでスクロールするときに、このコマンドを実行することをVoiceAttackに示します。この例では、「戻る」ボタンが選択されています。5つの標準マウスボタンの任意の組み合わせ、および単一のスクロールホイールイベントを使用できることに注意してください。右側の[…]ボタンをクリックして、このコマンドにマウスボタンを割り当てることができます。以下の画面が表示されます。





This is the 'Mouse Shortcut' screen. To clear anything that you've done on this screen, click the, 'clear' link. The option, 'Shortcut is invoked only when all buttons are released' will make the command only execute when all the buttons involved are let go. Note that any superseded shortcuts will not be executed when using this option.
これは「マウスショートカット」画面です。この画面で行ったことをクリアするには、「クリア」リンクをクリックします。オプション 'すべてのボタンがリリースされたときにのみショートカットが呼び出される ' は、関連するすべてのボタンが放されたときにのみコマンドを実行します。このオプションを使用すると、置き換えられたショートカットは実行されないことに注意してください。

That means if you have a shortcut that works when the, 'Back' button is released as well as a shortcut that works when, 'Back + Forward' are released, only the shortcut for 'Back + Forward' will be executed. The, 'Do not allow button down to be passed through' option prevents the button down event for the indicated buttons from being passed through to the application. For example, if your selected button is, 'Back' and this option is selected, VoiceAttack will respond to 'Back' button press and then prevent any other application from receiving this button press (for this example, if 'Back' is being handled by VoiceAttack, you will not be able to use the, 'Back' button in other applications while VoiceAttack is running. If you rely on, 'Back' to go back in your browser, then, you will have to pick another button).
つまり、「戻る」ボタンを放したときに機能するショートカットと、「戻る+進む」を放したときに機能するショートカットがある場合、「戻る+進む」のショートカットのみが実行されます。The、 ' ボタンを通過させない'オプションは、指定されたボタンのボタンダウンイベントがアプリケーションに渡されないようにします。たとえば、選択したボタンが「戻る」でこのオプションが選択されている場合、VoiceAttackは「戻る」ボタンの押下に応答し、他のアプリケーションがこのボタンの押下を受け取れないようにします（この例では、「戻る」が処理されている場合VoiceAttackにより、VoiceAttackの実行中に他のアプリケーションの[戻る]ボタンを使用することはできません。ブラウザに戻るために[戻る]を使用する場合は、別のボタンを選択する必要があります。

The‘Repeat command while buttons are held down’ option will allow the invoked command to repeat for as long as the selected mouse buttons are held down.
[ボタンを押したままコマンドを繰り返す]オプションを使用すると、選択したマウスボタンを押している間、呼び出されたコマンドを繰り返すことができます。

Note: This option is only available when the,‘Do not allow button events to be passed through’ option is NOT checked.
注：このオプションは、[ボタンイベントの通過を許可しない]オプションがオフの場合にのみ使用できます。

Note:‘Shortcut is invoked only when all the buttons are released’ is not applicable to the scroll wheel events as they do not have a,‘down’ or‘up’ state.
注：「すべてのボタンがリリースされたときにのみショートカットが呼び出される」は、スクロールホイールイベントには適用されません。「ダウン」または「アップ」状態ではないためです。

Note: Scroll wheel events are triggered for each, ‘click’, so, when scrolling forward or back, the command will be triggered each time the mouse wheel, ‘clicks’ in either direction. For left and right, holding the mouse down in either direction will cause the command to repeat as long as the wheel is held.
注：スクロールホイールイベントは「クリック」ごとにトリガーされるため、前後にスクロールすると、マウスホイールがいずれかの方向に「クリック」するたびにコマンドがトリガーされます。左右の場合、マウスをどちらかの方向に押し続けると、ホイールが保持されている限りコマンドが繰り返されます。

2. – Command Macro
2.– コマンドマクロ

This is a list of all of the actions that will be performed, in order. In the example, we are sending a series of keyboard key presses with a pause in between. The items in the list can be key presses, mouse clicks, pauses, application launches, etc. You can double-click on any item in the list to edit that item. Note that you can change the order of the items in the list by moving them up and down.
これは、順番に実行されるすべてのアクションのリストです。この例では、一連のキーボードのキーを押して送信し、その間に一時停止します。リスト内の項目は、キーを押す、マウスをクリックする、一時停止する、アプリケーションを起動するなどです。リスト内の項目をダブルクリックして、その項目を編集できます。リスト内の項目を上下に移動して、順序を変更できることに注意してください。

3. – Actions
3. –アクション

Add a Key Press action button
キーを押すアクションボタンを追加する

Click this button to add a keyboard key press to the command action sequence (See 'Key Press Screen'). You will probably be using this the most often.
このボタンをクリックして、キーボードのキープレスをコマンドアクションシーケンスに追加します（「キープレス画面」を参照）。おそらくこれを最も頻繁に使用するでしょう。

Add a Mouse action button
マウスアクションボタンを追加する

Click this button to add a mouse action (such as moving the mouse, clicking a mouse button) to the command action sequence (See 'Mouse Action Screen').
このボタンをクリックして、マウスアクション（マウスの移動、マウスボタンのクリックなど）をコマンドアクションシーケンスに追加します（「マウスアクション画面」を参照）。

Add a Pause action button
一時停止アクションボタンを追加する

Click this button to add a timed pause to the command action sequence (See 'Pause Screen' and 'Variable Pause Screen').
このボタンをクリックして、時間指定された一時停止をコマンドアクションシーケンスに追加します(「一時停止画面」および「可変一時停止画面」を参照)。

Add a miscellaneous, 'Other' action button
その他の「その他」アクションボタンを追加する

Click this button if you want to launch or close an application (among other things). (See 'Other Stuff Screen'). Lots of fun items in here.
(特に)アプリケーションを起動または閉じる場合は、このボタンをクリックします。(「その他の画面」を参照)。ここにはたくさんの楽しいアイテムがあります。

Recorder button
レコーダーボタン

Click this button to bring up the 'Key Press Recorder' screen. This screen will capture key presses as you perform them for insertion into your macro. This will make input easier for a series of keys (instead of adding one at a time). (See, 'Record Keypress Screen').
このボタンをクリックして、「キープレスレコーダー」画面を表示します。この画面は、マクロへの挿入のためにキーを押しながら実行します。これにより、一連のキーの入力が（一度に1つずつ追加するのではなく）簡単になります。（「キープレス画面の記録」を参照）。

4. – Command Organization Description
4.– コマンド編成の説明

This is where you can give a description of your command. Note: The description
ここで、コマンドの説明を入力できます。注：説明

column will not show up in the Commands list of the Profile screen unless you actually have at least one command with a description.
実際に少なくとも1つのコマンドに説明が含まれていない限り、[プロファイル]画面の[コマンド]リストに列は表示されません。

Category
カテゴリー

This allows you to organize your commands in a simple fashion. You can type in a new category in the box, or, you may drop down the list to select a category that already exists in your profile. Note: The category column will not show up in the Commands list unless you actually have at least one command with a category.
これにより、コマンドを簡単な方法で整理できます。ボックスに新しいカテゴリを入力するか、リストをドロップダウンして、プロファイルにすでに存在するカテゴリを選択できます。注：実際にカテゴリーを持つコマンドが少なくとも1つない限り、カテゴリー列は「コマンド」リストに表示されません。

5. – Command Execution Options
5. –コマンド実行オプション

'Allow other commands to execute while this one is running' option.
「このコマンドの実行中に他のコマンドの実行を許可する」オプション。

When VoiceAttack recognizes a command, it is executed in one of two ways. One way is synchronous (the option is unselected). That is, when the command is executing, subsequent, recognized commands must wait until the command is finished
VoiceAttackがコマンドを認識すると、2つの方法のいずれかで実行されます。1つの方法は同期です（オプションは選択されていません）。つまり、コマンドが実行されているとき、後続の認識されたコマンドは、コマンドが終了するまで待機する必要があります

before they execute. This is useful when you want the command to do its work without other commands interfering. When other commands try to execute, they can be canceled, or they can wait and then execute. The setting, ‘Cancel blocked commands’ on the options page will allow you to choose what to do with the commands that are blocked. If you choose to cancel blocked commands, the commands are simply ignored. If you choose to let the commands wait, when the blocking command is finished, all the waiting commands will then execute. Unfortunately, there is no queuing system yet, so these commands will execute simultaneously so use with caution. Note that calling a command that has this option set does not affect commands that are already running.
実行する前に。これは、他のコマンドが干渉することなく、コマンドに作業をさせたい場合に便利です。他のコマンドを実行しようとすると、それらをキャンセルしたり、待機してから実行したりできます。オプションページの[ブロックされたコマンドをキャンセルする]設定では、ブロックされたコマンドの処理を選択できます。ブロックされたコマンドをキャンセルすることを選択した場合、コマンドは単に無視されます。コマンドを待機させることを選択した場合、ブロッキングコマンドが終了すると、すべての待機コマンドが実行されます。残念ながら、キューイングシステムはまだないため、これらのコマンドは同時に実行されるため、注意して使用してください。このオプションが設定されているコマンドを呼び出しても、すでに実行されているコマンドには影響しません。

The second way to execute a command is asynchronously (the option is selected). That is, when the command is executing, VoiceAttack will continue to execute subsequent, recognized commands. This is the default behavior of VoiceAttack. Note that this option is applicable only to the root executed command. Commands executed as sub-commands (see ‘Execute Another Command’ action) will follow the root command’s setting. Please exercise caution when using this option, since key presses can interfere with each other. Note that you can click the 'Stop Commands' button on the Main Screen or add a command to stop all processing commands (see, 'Other Stuff' screen). This is basically a panic button that indicates to all running macros that they need to stop processing.
コマンドを実行する2番目の方法は非同期です（オプションが選択されます）。つまり、コマンドの実行中、VoiceAttackは認識された後続のコマンドの実行を継続します。これはVoiceAttackのデフォルトの動作です。このオプションは、ルート実行コマンドにのみ適用できることに注意してください。サブコマンドとして実行されるコマンド（「別のコマンドを実行」アクションを参照）は、ルートコマンドの設定に従います。キーを押すと相互に干渉する可能性があるため、このオプションを使用する場合は注意してください。メイン画面の[コマンドの停止]ボタンをクリックするか、コマンドを追加してすべての処理コマンドを停止することができます（[その他の項目]画面を参照）。これは基本的に、実行中のすべてのマクロに処理を停止する必要があることを示すパニックボタンです。

'Stop command if target window focus is lost' - Enabling this option will make this command stop if the focused window loses focus. The focused window is the window that has the focus when the command is first executed. One handy use for this feature is if you have a command that continually loops and interacts with a certain application. If you click out of the window and lose focus, you probably do not want processing to continue on the newly active window (especially if keys are being pressed). This feature also has an option called, 'Resume command if focus regained'. What this does is attempt to continue the command if you focus the original window.
「ターゲットウィンドウのフォーカスが失われた場合にコマンドを停止する」-このオプションを有効にすると、フォーカスされたウィンドウがフォーカスを失った場合にこのコマンドが停止します。フォーカスされたウィンドウは、コマンドが最初に実行されたときにフォーカスがあるウィンドウです。この機能の便利な使用法の1つは、特定のアプリケーションと継続的にループして対話するコマンドがある場合です。ウィンドウの外をクリックしてフォーカスを失った場合、おそらく新しいアクティブウィンドウで処理を続行したくないでしょう（特にキーが押されている場合）。この機能には、「フォーカスが回復したらコマンドを再開する」というオプションもあります。これは、元のウィンドウに焦点を合わせた場合にコマンドを続行しようとするものです。

'Send command to this target' - Enabling this option will make this command target either the active window or another window/process that you specify to receive input. This will override the target indicated at the profile level (if specified) as well as the target indicated on the main screen (global level).
「このターゲットにコマンドを送信」-このオプションを有効にすると、このコマンドをアクティブウィンドウまたは入力を受け取るように指定した別のウィンドウ/プロセスのいずれかに設定します。これは、プロファイルレベル（指定されている場合）で示されるターゲットと、メイン画面（グローバルレベル）で示されるターゲットをオーバーライドします。

To send input to the active window, choose, 'Active Window'. Choosing the active window at this level is handy when the profile or global settings are directed to a specific application. Whatever window that is currently active will receive input from this command.
アクティブウィンドウに入力を送信するには、「アクティブウィンドウ」を選択します。このレベルでアクティブウィンドウを選択すると、プロファイルまたはグローバル設定が特定のアプリケーションに向けられている場合に便利です。現在アクティブなウィンドウはすべて、このコマンドから入力を受け取ります。

To send input to a specific window or process, choose the option next to the drop down box. To see what windows are available, drop down the list. Choosing a window from the list will indicate to the command that you want to send input to it. Note that this is a free input text box and you can modify your selection (as detailed below).
特定のウィンドウまたはプロセスに入力を送信するには、ドロップダウンボックスの横にあるオプションを選択します。使用可能なウィンドウを確認するには、リストをドロップダウンします。リストからウィンドウを選択すると、コマンドに入力を送信するよう指示されます。これは無料の入力テキストボックスであり、選択を変更できることに注意してください（以下に詳細を示します）。

The value in the drop down box can contain wildcards indicated by asterisks (*). This is handy when the title of the window changes. To indicate that the window title contains the value in the box, put an asterisk on each end. For instance, if you want to target any window that contains, 'Notepad' in the title, put, '*Notepad*' (without quotes) in the box. To indicate that the window title starts with the value in the box, put an asterisk at the end: 'Notepad*'. To indicate that the window title ends with the value in the box, put an asterisk at the beginning: '*Notepad'. The values are not case- sensitive. The first window found to match the criteria indicated will be selected.
ドロップダウンボックスの値には、アスタリスク（*）で示されるワイルドカードを含めることができます。これは、ウィンドウのタイトルが変更されたときに便利です。ウィンドウのタイトルにボックスの値が含まれていることを示すには、両端にアスタリスクを付けます。たとえば、タイトルに「メモ帳」と入力し、ボックスに「*メモ帳*」（引用符なし）を含むウィンドウをターゲットにしたい場合。ウィンドウのタイトルがボックス内の値で始まることを示すには、最後にアスタリスク「Notepad *」を付けます。ウィンドウのタイトルがボックス内の値で終わることを示すには、先頭にアスタリスク「* Notepad」を付けます。値は大文字と小文字を区別しません。示された基準に一致することが検出された最初のウィンドウが選択されます。

Advanced: Note that you can also use process names as they appear in the Windows Task Manager. You can use wildcards the same as you do with the window titles.
高度：Windowsタスクマネージャーに表示されるプロセス名を使用することもできます。ウィンドウのタイトルと同じようにワイルドカードを使用できます。

Window titles are checked first, and then the process names.
最初にウィンドウのタイトルがチェックされ、次にプロセス名がチェックされます。

More advanced: If a window cannot be found by title or process name, the window class names will then be checked. Wildcards apply if you need them. Note that this will be the class name of the window itself and not the class name of a child control. Again, this is an advanced feature that you may never ever use.
より高度：タイトルまたはプロセス名でウィンドウが見つからない場合、ウィンドウクラス名がチェックされます。ワイルドカードは必要な場合に適用されます。これはウィンドウ自体のクラス名であり、子コントロールのクラス名ではないことに注意してください。繰り返しになりますが、これは使用することのない高度な機能です。

Optimization note – As indicated above, the, ‘target’ input box will accept the name of a window title, process name or window class name, and checks for each of these items in that order. The reason for doing all of this in one go is for user simplicity (less user interface) as well as user assistance in locating their intended target (as oftentimes there is an overlap in naming). This works rather quickly in most cases, but in some situations the processing could be unnecessarily excessive. For instance, if you are looking for a process name that contains, ‘widget’, all window titles will be searched for, ‘widget’ first before the process names are searched. Again, this is a fast check, but it is unnecessary checking if you already know for sure that, ‘widget’ will only appear in a process name (and not a window’s title). In order to not add more user interface clutter and to provide those of you looking to eek out every bit of performance you can get, VoiceAttack now has character prefixes to limit searches to just the window title, process or class name.
最適化に関する注意 –上記のように、「ターゲット」入力ボックスは、ウィンドウタイトルの名前、プロセス名、またはウィンドウクラス名を受け入れ、これらの各項目をこの順序で確認します。このすべてを一度に実行する理由は、ユーザーの単純さ（ユーザーインターフェイスが少ない）と、目的のターゲットを見つける際のユーザーの支援（多くの場合、名前の重複があるため）です。ほとんどの場合、これはかなり速く動作しますが、状況によっては処理が不必要に過剰になる可能性があります。たとえば、「widget」を含むプロセス名を探している場合、プロセス名が検索される前に「widget」が最初にすべてのウィンドウタイトルが検索されます。繰り返しますが、これは高速なチェックですが、「ウィジェット」はプロセス名にのみ表示され（ウィンドウのタイトルには表示されない）ことが確実にわかっている場合は不要なチェックです。

If '^' is prepended to the target value, the target search is limited to only window titles. If '~' is prepended, only process names are searched.
' ^ 'がターゲット値の前に追加された場合、ターゲット検索はウィンドウタイトルのみに制限されます。「～」が先頭に追加された場合、プロセス名のみが検索されます。

If '+' is prepended, only class names are searched.
' + 'が先頭に追加された場合、クラス名のみが検索されます。

These new prefixes can also be used in conjunction with wildcards.
これらの新しいプレフィックスは、ワイルドカードと組み合わせて使用することもできます。

For example, '^*notepad*' will indicate to only search window titles that contain, 'notepad'. If, ‘notepad’ is not found within a window title, the search will stop and not continue on to search processes or class names. ‘~widget*’ indicates that the search is to only look for process names that start with, ‘widget’. That means that no window titles or class names are searched. ‘+foo’ indicates that the search should only be for window class names that match, ‘foo’ (again, window titles and process names are not searched first which saves some time).
たとえば、「^ * notepad *」は、「notepad」を含む検索ウィンドウタイトルのみを示します。ウィンドウタイトル内に「notepad」が見つからない場合、検索は停止し、検索プロセスまたはクラス名に進みません。「～widget *」は、「widget」で始まるプロセス名のみを検索することを示します。つまり、ウィンドウのタイトルやクラス名は検索されません。「+ foo」は、一致するウィンドウクラス名のみを検索する必要があることを示します。「foo」（再び、ウィンドウタイトルとプロセス名は最初に検索されないため、時間を節約できます）。

Lots more information about process targets is available in this document in the section titled, 'Application Focus (Process Target) Guide'.
プロセスターゲットの詳細については、このドキュメントの「アプリケーションフォーカス（プロセスターゲット）ガイド」というタイトルのセクションを参照してください。

'Minimum Confidence Level' allows you to specify to VoiceAttack what the minimum recognition confidence level must be in order to execute this command. This value overrides the values set at global level as well as the profile level. See the, 'Options' page for more information about the handy confidence feature.
「最小信頼レベル」では、このコマンドを実行するために最低限必要な認識信頼レベルをVoiceAttackに指定できます。この値は、グローバルレベルおよびプロファイルレベルで設定された値をオーバーライドします。便利な信頼性機能の詳細については、「オプション」ページを参照してください。

'Recognition' – this is currently an experimental feature that will allow you to possibly speed up how fast your spoken command is recognized by the speech engine.
「認識」–これは現在、音声エンジンによって音声コマンドが認識される速度を高速化できる実験的な機能です。

Normally, your speech engine waits for you to briefly finish speaking before making a final decision about what you just said (and subsequently execute a command if it finds a match). This behavior of the speech engine is represented by the, 'Normal' setting for Recognition, and is the default selection (you know – how VA has always behaved).
通常、スピーチエンジンは、発言の最終決定を下す(そして一致するものが見つかった場合はコマンドを実行する)前に、発言が終了するのをしばらく待ちます。音声エンジンのこの動作は、認識の「標準」設定で表され、デフォルトの選択です(VAが常にどのように動作するか)。

Now for the good stuff… Your speech engine is constantly listening to you, and as the speech engine is listening, it is attempting to contextually piece together what you are saying based on the spoken command phrases you've created in your profile. As the speech engine gathers up what can be constituted as one of your spoken phrases within continuous speech, you have the option to execute your command at that point in time rather than wait for the speech engine that is also waiting for you to stop speaking. This can shave off a few milliseconds in your speech event. Select the, 'Continuous Speech' option to attempt to execute your command as part of continuous speech. The, 'Restricted Continuous Speech' option works exactly like the, 'Continuous Speech' option, but the phrase to be recognized must be at the beginning of the speech event. The, 'Restricted' option is the one that I prefer the most, as it allows me to issue commands by themselves (as we've always done) as well as bypass the speech event delay somewhat. So, for example, let's say you have the spoken phrase, 'fire weapons'. With, 'Normal' selected, if you say, 'Ok fire weapons', the speech engine accepts, 'Ok fire weapons' as the phrase you are trying to execute (this is after the brief pause that we are all familiar with). The phrase, 'Ok fire weapons' is not found and you'll be met with, 'Unrecognized: Ok fire weapons' (of course). If you choose, 'Continuous Speech', when the speech engine encounters, 'Ok fire weapons', it sees, 'fire weapons' and doesn't wait for you to finish speaking before executing the, 'fire weapons' command. This is great, but if you don't like having the chance of your weapons firing in the middle of a sentence, you will want to give, 'Restricted Continuous Speech' a try. The spoken phrase, 'I like turtles and fire weapons' will not be picked up as a recognized command because, 'fire weapons' was not spoken at the start of the speech event. The spoken phrase, 'fire weapons' will get picked up as a recognized command (of course) as well as, 'fire weapons and stuff' (again, since, 'fire weapons' is at the start of the phrase (speech event)).
音声エンジンは常にあなたの話を聞いており、音声エンジンが聞いている間、プロフィールで作成した音声コマンドフレーズに基づいて、あなたの言っていることを文脈的にまとめようとしています。音声エンジンは連続音声内の音声フレーズの1つとして構成できるものを収集するため、音声エンジンが発言を停止するのを待つのではなく、その時点でコマンドを実行するオプションがあります。これにより、スピーチイベントの数ミリ秒を短縮できます。「連続音声」オプションを選択して、連続音声の一部としてコマンドを実行しようとします。、 ' 制限された連続音声'オプションは' Continuous Speech 'オプションとまったく同じように機能しますが、認識されるフレーズはスピーチイベントの先頭になければなりません。
'Restricted'オプションは、音声イベントの遅延を多少バイパスするだけでなく、(常に行っているように)自分でコマンドを発行できるため、私が最も好むオプションです。したがって、たとえば、「火の武器」という話し言葉があるとします。「通常」が選択された状態で、「Ok fire weapons」と言うと、スピーチエンジンは「Ok fire weapons」を実行しようとしているフレーズとして受け入れます(これは私たちがよく知っている短い休止の後です)。「Ok fire weapons」というフレーズは見つかりません。「Unrecognized:Ok fire weapons」(もちろん)が表示されます。「Continuous Speech」を選択すると、ス

So, what's the catch? Well, you knew there would be some kind of catch, right? First, and most importantly, you're only going to want to use the, 'Continuous Speech' and 'Restricted Continuous Speech' options on spoken commands that are very distinct within your profile. How distinct they should be is totally up to you and your speaking style, so, you're going to want to play around with that a bit.
それで、キャッチは何ですか？まあ、あなたはある種のキャッチがあることを知っていましたよね？まず、そして最も重要なことは、プロファイル内で非常に明確な音声コマンドで、[連続音声]および[制限連続音声]オプションのみを使用することです。どのように区別するかは完全にあなたとあなたの話すスタイル次第ですので、あなたはそれを少し試してみたいと思うでしょう。

Second, if you are using minimum confidence level thresholds, you'll notice that the speech engine is a bit less confident while it's in the middle of figuring out your continuous speech. You'll find that you'll probably be lowering your minimum confidence thresholds a lot for commands that do not have, 'Normal' set. I'd suggest not using anything but, 'Normal' on stuff like, 'eject', 'eject cargo', 'self destruct' and stuff like that ;) You've been warned (lol).
第二に、最小信頼レベルのしきい値を使用している場合、連続発話を把握している最中に、音声エンジンの自信が少し低下していることに気付くでしょう。「通常」が設定されていないコマンドの場合、おそらく最小信頼しきい値を大幅に下げることに気付くでしょう。「イジェクト」、「イジェクトカーゴ」、「自己破壊」などのようなものには「通常」を使用することをお勧めします;)あなたは警告されました（笑）。

6. – Action Management
6.– アクション管理

Move action Up button
アクションを上に移動ボタン

Click this button to move a selected action to an earlier part of the sequence. This can also be achieved by holding down the control button and pressing the up arrow key.
このボタンをクリックして、選択したアクションをシーケンスの前の部分に移動します。これは、コントロールボタンを押しながら上矢印キーを押すことでも実現できます。

Move action Down button
アクションを下に移動ボタン

Click this button to move a selected action to a later part of the sequence. This can also be achieved by holding down the control button and pressing the down arrow key.
このボタンをクリックして、選択したアクションをシーケンスの後半に移動します。これは、コントロールボタンを押しながら下矢印キーを押すことでも実現できます。

Edit an action button
アクションボタンを編集する

Click this button to modify the selected action (works the same as double-clicking on the selected item).
このボタンをクリックして、選択したアクションを変更します（選択したアイテムをダブルクリックするのと同じように機能します）。

Delete an action button
アクションボタンを削除する

Click this button to remove the selected action from the sequence (works the same as hitting the Delete key).
このボタンをクリックして、選択したアクションをシーケンスから削除します（Deleteキーを押すのと同じ働きをします）。

Undo
元に戻す

Click this button to undo the last change you made to the command lists.
このボタンをクリックして、コマンドリストに行った最後の変更を元に戻します。

Redo
やり直し

Click this button to redo a change that has been undone.
元に戻した変更をやり直すには、このボタンをクリックします。

7. - Command Type
7.- コマンドタイプ

VoiceAttack supports full commands (this is what you will probably be using almost exclusively) as well as composite (prefix/suffix) commands. Prefixes and suffixes only execute when they are used together in a composite voice command (which means that they will not execute on their own). When they are executed together, the actions from the prefix are executed first, followed by the actions in the suffix. This is handy if you want to create a lot of similar commands, without copying and modifying over and over again.
VoiceAttackは、完全なコマンド（これはおそらくほぼ排他的に使用するものです）および複合（プレフィックス/サフィックス）コマンドをサポートします。プレフィックスとサフィックスは、複合音声コマンドで一緒に使用された場合にのみ実行されます（つまり、単独では実行されません）。一緒に実行されると、プレフィックスのアクションが最初に実行され、その後にサフィックスのアクションが続きます。これは、何度も繰り返しコピーおよび変更せずに、同様のコマンドを多数作成する場合に便利です。

As an example, let's use the actions found in a racing game. Let's say that the races potentially have up to 100 drivers. If you wanted to create commands to eject any one of the drivers from a race, you would need to create 100 commands ('eject driver 1', 'eject driver 2', 'eject driver 3', etc.). Next, if you wanted to mute any one of those drivers you would need to create another 100 commands ('mute driver 1', 'mute driver 2', 'mute driver 3', etc.). For every action that involves drivers, you would need to create another 100 commands. To solve this with prefixes and suffixes, you would first create the suffixes for the 100 drivers (yeah... I know that's a lot). The suffix actions would be something like this (for driver 82):
例として、レースゲームで見られるアクションを使用してみましょう。レースには潜在的に最大100人のドライバーがいるとしましょう。ドライバーをレースからイジェクトするコマンドを作成する場合、100個のコマンドを作成する必要があります（「イジェクトドライバー1」、「イジェクトドライバー2」、「イジェクトドライバー3」など）。次に、これらのドライバーのいずれかをミュートするには、別の100個のコマンド（「ドライバー1をミュート」、「ドライバー2をミュート」、「ドライバー3をミュート」など）を作成する必要があります。ドライバーが関係するすべてのアクションに対して、さらに100個のコマンドを作成する必要があります。プレフィックスとサフィックスを使用してこれを解決するには、最初に100個のドライバーのサフィックスを作成します（ええ...それはたくさんあることを知っています）。サフィックスアクションは次のようなものになります（ドライバー82の場合）。

Press 8
8を押す

Release 8
リリース8

Press 2
2を押す

Release 2 Press Enter Release Enter
リリース2 Enterキーを押すリリースEnter

Once you have your suffixes lined up, you can then create as many prefixes as you need to work with them. For, 'mute driver', you create a command and designate it as a prefix. Its actions would look like this:
サフィックスを並べたら、必要な数だけプレフィックスを作成できます。「ミュートドライバー」の場合、コマンドを作成し、プレフィックスとして指定します。そのアクションは次のようになります。

Press m Release m Press u Release u Press t Release t Press e Release e Press Space Release Space
mを押すmを押すuを押すuを押すtを押すtを押すeを押すeを押す

When you say, 'mute driver 82', all the actions from, 'mute driver' will be executed first, followed by the actions in suffix '82'. You only have to create one prefix and it is automatically paired with all of the available suffixes.
「ミュートドライバー82」と言うと、「ミュートドライバー」からのすべてのアクションが最初に実行され、その後にサフィックス「82」のアクションが続きます。プレフィックスを1つ作成するだけで、使用可能なすべてのサフィックスと自動的にペアになります。

Prefix/Suffix Group -
プレフィックス/サフィックスグループ-

Indicate a group name for your prefix or suffix to have the pairing of the prefix and suffix to occur only within that group. For instance, you can have a group called, 'taunt' where all of the suffixes contain something funny to say when you mute or eject a driver:)
プレフィックスまたはサフィックスのグループ名を指定すると、そのグループ内でのみプレフィックスとサフィックスのペアが発生します。たとえば、「taunt」というグループを作成できます。このグループでは、すべてのサフィックスに、ドライバーをミュートまたはイジェクトするときにおかしなことを言うことができます。

8. - Repeating
8. -繰り返し

VoiceAttack can execute the actions of a command once, or as many times as you like. To get VoiceAttack to repeat the command actions indefinitely, select the option labeled, 'This command repeats continuously'. To repeat the actions a certain number of times, choose, 'This command repeats X times' and fill in the number of times to repeat in the box provided.
VoiceAttackは、コマンドのアクションを1回、または何度でも実行できます。VoiceAttackがコマンドアクションを無期限に繰り返すようにするには、「このコマンドは連続して繰り返す」というラベルの付いたオプションを選択します。特定の回数だけアクションを繰り返すには、「このコマンドをX回繰り返す」を選択し、表示されるボックスに繰り返し回数を入力します。

To get VoiceAttack to stop repeating, you have a couple of options. The most heavy- handed way is to issue a command to stop all processing (this can be invoked from the main screen by clicking the, 'Stop Commands' button, by pressing the stop command hotkey(s) (see Options page) or by issuing a voice command that stops command processing (see Other Stuff screen). Another option is to issue a voice command that calls the, 'Stop Another Command' action. This can be found on the, 'Other Stuff' screen. Note: Looping is also available from within commands as of version 1.5.9.
VoiceAttackに繰り返しを停止させるには、いくつかのオプションがあります。最も手間のかかる方法は、すべての処理を停止するコマンドを発行することです（これは、[停止コマンド]ボタンをクリックするか、停止コマンドホットキー（オプションページを参照）を押すか、コマンドの処理を停止し、音声コマンドを発行する（その他の項目の画面を参照）別のオプションは、これは、「その他の項目」画面で見ることができるアクション「別のコマンドを停止」を呼び出し、音声コマンドを発行することです。。。注：ループバージョン1.5.9以降のコマンド内からも使用できます。

OK button
OKボタン

Click the OK button to commit all changes to the command.
[OK]ボタンをクリックして、コマンドに対するすべての変更をコミットします。

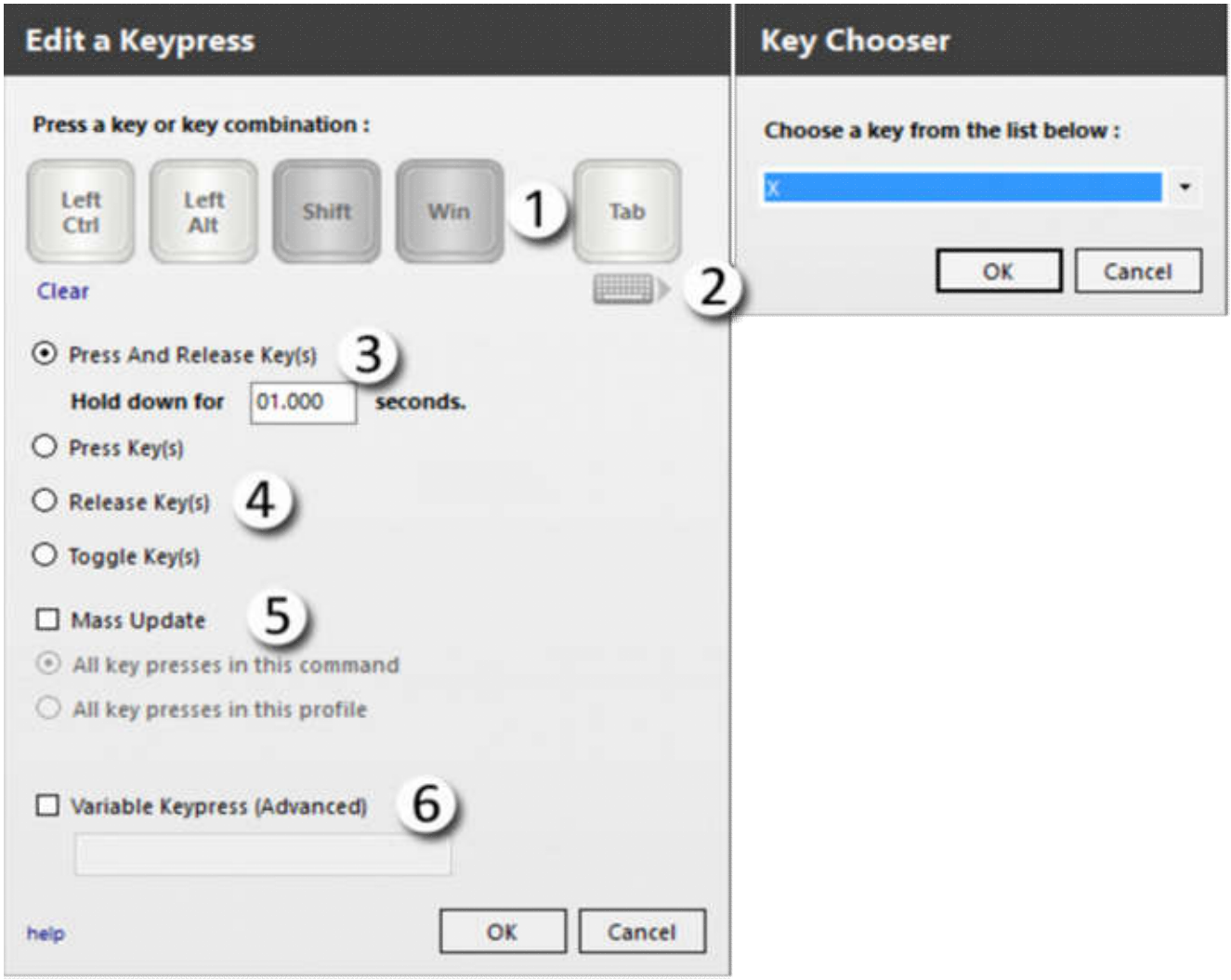
Cancel button
キャンセルボタン

All changes to this command will be undone if you click the Cancel button.
[キャンセル]ボタンをクリックすると、このコマンドに対するすべての変更が取り消されます。

Note: Actions in this list can be copied and pasted within the same command action list as well as command action lists in other profiles. Most actions indicated above are also available in the right-click menu of the command action list.
注：このリストのアクションは、他のプロファイルのコマンドアクションリストと同じコマンドアクションリスト内でコピーして貼り付けることができます。上記のほとんどのアクションは、コマンドアクションリストの右クリックメニューでも使用できます。

Key Press Screen
キープレス画面

This screen allows you to define a single key press for a command action (See 'Command Screen'). You are also allowed to specify a modifier with your key press. The modifiers that are available are 'Shift', 'Control', 'Alt' and 'Windows'. If your key press needs to be held down for a certain amount of time, you can indicate that time frame in seconds (up to a maximum of 99.999 seconds).
この画面では、コマンドアクションの単一キーの押下を定義できます（「コマンド画面」を参照）。また、キーを押して修飾子を指定することもできます。使用可能な修飾子は、「Shift」、「Control」、「Alt」、および「Windows」です。キーを一定時間押し続ける必要がある場合は、その時間枠を秒単位で指定できます（最大99.999秒）。



1. – Selected keys
1.– 選択されたキー

This is where you indicate what keys to press. If you press, 'X', the right-most key icon will display, 'X'. If you press a modifier key (ctrl, alt, shift or Windows), one of the left- most key icons will display what you pressed. This will make up the key combination that VoiceAttack will send to your application.
ここで、押すキーを指定します。「X」を押すと、右端の鍵アイコン「X」が表示されます。修飾キー（Ctrl、Alt、Shift、またはWindows）を押すと、左端のキーアイコンの1つに、押したものが表示されます。これにより、VoiceAttackがアプリケーションに送信するキーの組み合わせが構成されます。

2. – Key chooser
2. –キーチュー

Clicking the mini keyboard pops up the, 'Extended Key Chooser' screen. From this screen, you can select any of the available keyboard keys (for example, if your
ミニキーボードをクリックすると、「拡張キーの選択」画面が表示されます。この画面から、使用可能な任意のキーボードキーを選択できます（たとえば、

physical keyboard does not have media or browser keys, you can select them from here).
物理キーボードにはメディアまたはブラウザのキーはありません。ここから選択できます）。

3. – Press and release key(s) option
3.– キーを押して離すオプション

If you choose this option, VoiceAttack will press down and release a key. This option is further enhanced by using the, 'Hold down for X seconds' box. Enter a value here (in seconds) to indicate how long VoiceAttack is to hold down the key/key combination before releasing. Note that a value of zero is not recommended for most games, as games tend to rely on polling for key state (a value of zero might make the key press occur too quickly for the game to react).
このオプションを選択すると、VoiceAttackはキーを押して離します。このオプションは、「X秒間保留」ボックスを使用することによりさらに強化されます。ここに値（秒単位）を入力して、VoiceAttackがキー/キーの組み合わせを押してから放す時間を示します。ゲームはキー状態のポーリングに依存する傾向があるため、ほとんどのゲームではゼロの値は推奨されないことに注意してください（ゼロの値は、ゲームが反応するためにキーを押す速度が速すぎることがあります）。

4. – Press key(s) option
4.– キーを押します

Select this option if you only want VoiceAttack to press the selected keys down. This is usually used in a macro, with a subsequent, 'Release key(s)' action.
VoiceAttackが選択したキーを押し下げるだけの場合は、このオプションを選択します。これは通常、マクロで使用され、後続の「キーを離す」アクションで使用されます。

o Release key(s) option
o キーを解放するオプション

Select this option if you only want VoiceAttack to release the selected keys. This is usually used in a macro, preceded by a, 'Press key(s)' action.
VoiceAttackに選択したキーのみをリリースさせる場合は、このオプションを選択します。これは通常、「キーを押す」アクションが先行するマクロで使用されます。

o Toggle key(s) option
o トグルキーオプション

Select this option if you want VoiceAttack to press a key if it is not pressed or release a key if it is pressed.
VoiceAttackが押されていない場合はキーを押すか、押されている場合はキーを離すようにする場合は、このオプションを選択します。

5. – Mass update
5.– 一括更新

When you are editing a key press, you have the option to update all of the matching key presses in either the current command or the current profile to be the same as the one you are currently editing.
あなたがいる場合は、編集キーを押して、あなたは、現在編集しているものと同じように、現在のコマンドまたは現在のプロファイルのいずれかでマッチングキープレスのすべてを更新するオプションを持っています。

So, if you have a bunch of commands that press the, 'X' key and you want to change all of them to use the 'Y' key, simply choose one of your key press actions that presses the, 'X' key only and edit it. Change the key to, 'Y'. Then, choose mass update and then the 'all keypresses in this command' option. When you click, 'OK', all the actions in the command that had a key press of, 'X' will now be, 'Y' (all key press actions that did not have a key of, 'X' will be left alone).
したがって、「X」キーを押すコマンドの束があり、それらすべてを「Y」キーを使用するように変更する場合は、「X」キーのみを押すキー押下アクションのいずれかを選択しますそれを編集します。キーを「Y」に変更します。次に、一括更新を選択し、「このコマンドですべてのキーを押す」オプションを選択します。「OK」をクリックすると、「X」のキーが押されたコマンド内のすべてのアクションが「Y」になります（「X」のキーがなかったすべてのキー押下アクションが残ります）単独）。

Note that the keys and duration are the only attributes that are mass updated. The press method (down/up/press) are not updated.
キーと期間が一括更新される唯一の属性であることに注意してください。プレス方法（ダウン/アップ/プレス）は更新されません。

Also note that hitting, 'Cancel' on the Key Press screen will not cancel a mass update. You will need to press, 'Cancel' on the Profile edit screen in order to cancel the operation, as the entire profile will be updated.
また、キープレス画面で[キャンセル]を押しても、一括更新はキャンセルされません。プロファイル全体が更新されるため、操作をキャンセルするには、プロファイル編集画面で「キャンセル」を押す必要があります。

6. – Variable Keypress (Advanced)
6. –可変キー押下（詳細）

Selecting this option will allow you to use a text variable to indicate the keys to press for this action (instead of using the icons at the top of the screen). This is to aid (primarily) in cases where keys for various commands may change periodically (such as with key bindings for games). The idea is that keypress variables would be initialized at profile startup or by plugin activation, thereby not requiring constant, manual updating of commands each time the key bindings change in a game.
このオプションを選択すると、テキスト変数を使用して、画面上部のアイコンを使用する代わりに、このアクションのために押すキーを示すことができます。これは、さまざまなコマンドのキーが定期的に変更される可能性がある場合（ゲームのキーバインドなど）に（主に）支援するためです。キープ変数はプロファイルの起動時またはプラグインのアクティブ化によって初期化されるため、ゲーム内でキーバインディングが変更されるたびにコマンドを絶えず手動で更新する必要はありません。

Usage is pretty straightforward. To turn on variable keypresses for this action, make sure the box is checked and simply put the name of the text variable to use in the provided input box. The previously-set text variable must contain properly-notated text in order to work. The good news is that the notation is (almost) exactly the same as what you will find in Quick Input. So, for instance, if your command is to raise your landing gear and the keypress (for now) is ALT + L, set a text variable's value to '[ALT]L' (no quotes).
使い方はとても簡単です。このアクションの変数キー押下をオンにするには、ボックスがチェックされていることを確認し、使用するテキスト変数の名前を提供された入力ボックスに入力します。事前に設定されたテキスト変数は、正しく機能するために適切に表記されたテキストを含む必要があります。幸いなことに、表記法は(ほぼ)クイック入力で見つかるものとまったく同じです。したがって、たとえば、コマンドが着陸装置を上げることであり、キープレス(今のところ)ALT + Lである場合、テキスト変数の値を '[ALT] L'(引用符なし)に設定します。

Note that the, 'L' does not have brackets. Keys with a single-character identifier (A-Z,
「L」には角括弧がありません。単一文字の識別子(AZ、

+, ß, ç, etc.) do not need brackets. Special keys, such as Enter, Shift, Alt, F12, etc. will require brackets (see the section titled, 'Quick Input and Variable Keypress Key Indicators' for all the possible key indicators). Note also that there is no space between [ALT] and L. Spaces are actually picked up as key presses here, so if there is a space, the space bar will be pressed.
+、ß、çなど)は括弧を必要としません。Enter、Shift、Alt、F12などの特殊キーにはブラケットが必要です(可能なすべてのキーインジケーターについては、「クイック入力および可変キープレスキーインジケーター」のセクションを参照してください)。また、[ALT]とLの間にスペースがないことに注意してください。スペースは実際にここでキーを押すと取得されるため、スペースがある場合はスペースバーが押されます。

Put the name of the variable in the input box, and when the action is executed, the appropriate key method is performed (press, down, release, toggle) with the keys indicated in the variable. Note that with multiple keys that the order that the keys go down are the order that you provide. This happens virtually instantaneously, but order is still important. So, in this case, '[ALT]L', the Alt key is manipulated first, and then L. When releasing keys, the order is reversed. In this case, the L key would be released first and then the Alt key. This is so you can use the same variable to press keys down and then release the keys in the proper order without having to create another variable.
入力ボックスに変数の名前を入力し、アクションが実行されると、変数に示されているキーを使用して適切なキーメソッドが実行されます(押し、下へ、放し、切り替え)。複数のキーを使用する場合、キーが下がる順序が指定した順序になることに注意してください。これは事実上瞬時に行われますが、順序は依然として重要です。したがって、この場合、[[ALT] L '、Altキーが最初に操作され、次にLが操作されます。キーを離すと、順序が逆になります。この場合、最初にLキーがリリースされ、次にAltキーがリリースされます。これは、同じ変数を使用してキーを押し下げ、別の変数を作成せずに適切な順序でキーを放すことができるようにするためです。

Important: Since we are pressing keys and not (necessarily) generating characters (as with Quick Input), the keys that are pressed will be unmodified keys. So, for instance, if you are using an English keyboard and you put in, '@' as the keypress you will notice that 2 key will be pressed. In order to reproduce the, '@' as a keypress, you will need to modify the keypress yourself by including Shift, like so: [SHIFT]2 or [SHIFT]@ (either of these will work). This is the same as it has always been with using keypresses, however it seems a little more pronounced now.
重要:キーを押しているので、(クイック入力のように)文字を(必ずしも)生成していないため、押されたキーは変更されていないキーになります。したがって、たとえば、英語のキーボードを使用していて、キー入力として「@」を入力すると、2つのキーが押されることに気付くでしょう。キープレスとして「@」を再現するには、[SHIFT] 2または[SHIFT] @のように、Shiftキーを押してキープレスを自分で変更する必要があります(これらのいずれかが機能します)。これは、キー押下を使用した場合と同じでしたが、今ではもう少しはっきりしているようです。

Important: As an interesting side effect, you can pretty much manipulate as many keys as you want at once, including all of the modifier keys (LAlt, RAlt, LShift,RShift, etc.).
重要：興味深い副作用として、すべての修飾キー（LAlt、RAlt、LShift、RShiftなど）を含め、必要なだけ多くのキーを一度に操作できます。

Remember, again, they will all occur in sequence in the order you provide (no pauses between). Multiple instances of the same key repeated in a keypress may not have the desired result.
'[ENTER][ENTER][ENTER][ENTER][ENTER]' probably will not press the enter key five times, however, it will usually press it more than once due to timing (your system may vary). Just don't do that lol.
繰り返しますが、これらはすべて、指定した順序で順番に発生します（間に休止はありません）。キーを押すたびに同じキーの複数のインスタンスが繰り返されると、望ましい結果が得られない場合があります。「[ENTER] [ENTER] [ENTER] [ENTER] [ENTER]」はおそらくEnterキーを5回押しませんが、通常はタイミングのために1回以上押します（システムによって異なる場合があります）。そんなことしないでください。

Note: The associated input box can also accept tokens (if, for some reason the variable name needs to be variable… yikes lol).
注：関連付けられた入力ボックスは、トークンを受け入れることもできます（何らかの理由で変数名を変数にする必要がある場合は... yikes lol）。

In the above example, at position 1, we want VoiceAttack to press 'Control' plus 'Alt' and 'X' keys all at the same time and release them after holding them down for 1 second. Note that 'Shift' and 'Win' are not selected.
上記の例では、位置1でVoiceAttackに「Control」キーと「Alt」キーと「X」キーをすべて同時に押し、1秒間押し続けてから離します。「Shift」と「Win」は選択されていないことに注意してください。

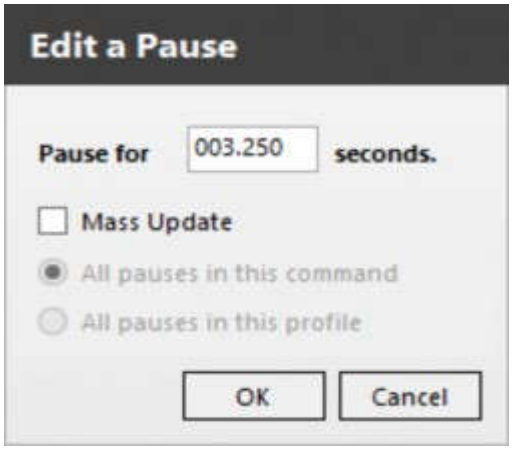
When you click the 'OK' button, the indicated key press is added to your command action
「OK」ボタンをクリックすると、指定されたキーの押下がコマンドアクションに追加されます

sequence. If the 'Cancel' button is pressed, you are returned to the Add/Edit Command screen, and no changes are recorded.
シーケンス。[キャンセル]ボタンを押すと、コマンドの追加/編集画面に戻り、変更は記録されません。

Pause Screen
一時停止画面

This screen allows you to define a single, timed pause for a command action (See 'Command Screen'). This is useful for waiting between key presses or waiting for a program to launch.
この画面では、コマンドアクションの単一の時間指定された一時停止を定義できます（「コマンド画面」を参照）。これは、キーを押すたびに待つか、プログラムが起動するのを待つのに便利です。

Simply add the amount of time you would like VoiceAttack to wait (up to 999.999 seconds) and click the 'OK' button. To cancel adding or editing a pause, click the, 'Cancel' button and no changes will be recorded.
VoiceAttackが待機する時間（最大999.999秒）を追加して、[OK]ボタンをクリックします。一時停止の追加または編集をキャンセルするには、[キャンセル]ボタンをクリックします。変更は記録されません。



In the example, above, we are adding a pause for 3 and a quarter seconds.
上記の例では、3秒と1/4秒の一時停止を追加しています。

Note that in the, 'Edit a Pause' screen, you have the ability to mass update all the pauses in the current command or in the current profile. Just select the, 'Mass Update' option and then choose the scope of the change. This will only change pause actions where the value matches the original pause value.
[一時停止の編集]画面では、現在のコマンドまたは現在のプロファイルのすべての一時停止を一括更新することができます。[一括更新]オプションを選択して、変更の範囲を選択します。これは、値が元の一時停止値と一致する一時停止アクションのみを変更します。

So, let's say we have a lot of pauses in our command that are 1 second long, but we want to change them all to 2 seconds. Just edit one of the pause actions that have 1 second, change the value to 2 seconds and then enable, 'mass update' and choose, 'all pauses in this command'. When you click, 'OK', all pause actions in the command that were 1 second will now be 2 seconds (pauses with different values other than 1 will not be altered).
したがって、1秒の長さのコマンドに多くの一時停止があるとしますが、それらをすべて2秒に変更したいとします。1秒の一時停止アクションのいずれかを編集し、値を2秒に変更してから、「一括更新」を有効にして「このコマンドのすべての一時停止」を選択します。[OK]をクリックすると、コマンド内の1秒であったすべての一時停止アクションが2秒になります（1以外の異なる値の一時停止は変更されません）。

Note that hitting, 'Cancel' on the Pause screen will not cancel a mass update. You will need to press, 'Cancel' on the Profile edit screen in order to cancel the operation, as the entire profile will be updated.
一時停止画面で「キャンセル」を押しても、大量更新はキャンセルされません。プロファイル全体が更新されるため、操作をキャンセルするには、プロファイル編集画面で「キャンセル」を押す必要があります。

Variable Pause Screen
可変一時停止画面

The Variable Pause screen works just like the Pause screen except instead of providing an exact pause time, you can specify a decimal variable to use, or any combination of tokens/literals that either resolve to a variable name or to a decimal value. The decimal variable must be set prior to using this feature (See, 'Set a Decimal Value'), otherwise the pause will not occur (since it will be zero seconds). Variable pauses can be used to have control over profile- wide pauses (change in one place, versus changing in many places). They can also be used with random values to give a slightly more natural feel to TTS, for instance.
変数一時停止画面は、正確な一時停止時間を提供する代わりに、使用する10進数変数、または変数名または10進数値に解決されるトークン/リテラルの任意の組み合わせを指定できることを除いて、一時停止画面と同様に機能します。この機能を使用する前に10進変数を設定する必要があります(「10進値の設定」を参照)。そうしないと、一時停止は発生しません(ゼロ秒になるため)。可変の一時停止を使用して、プロファイル全体の一時停止を制御できます(多くの場所で変更するのではなく、1つの場所で変更します)。たとえば、ランダムな値を使用して、TTSに少し自然な感じを与えることもできます。

Other Stuff Screen
その他の画面

Clicking the, 'Other' button on the Add/Edit Command screen will display a fly out menu with several submenus: VoiceAttack Action, Sounds, Windows, Dictation and Advanced. Clicking on any of these submenus will display the various, 'special' actions you can add to your commands. Note that there is a little star up near the top. Clicking on this star will add this action type to your favorites which can then be accessed from the, 'Other' button on the Command screen. If you hold down CTRL while clicking the star, you can optionally clear all your favorites.
[コマンドの追加/編集]画面の[その他]ボタンをクリックすると、VoiceAttack Action、Sounds、Windows、Dictation、Advancedのサブメニューを含むフライアウトメニューが表示されます。これらのサブメニューのいずれかをクリックすると、コマンドに追加できるさまざまな「特別な」アクションが表示されます。上部近くに小さな星があります。この星をクリックすると、このアクションタイプがお気に入りに追加され、コマンド画面の[その他]ボタンからアクセスできます。Ctrlキーを押しながら星をクリックすると、オプションですべてのお気に入りをクリアできます。

When you select a special action, you'll be presented with its corresponding add/edit screen.
特別なアクションを選択すると、対応する追加/編集画面が表示されます。

Some items in here may be useful to you. Other items you will probably never use. There's a lot going on here, so, I'll try to do my best to explain each item. The special actions are organized below in the same manner they are organized in the submenus:
ここにあるいくつかのアイテムはあなたにとって役に立つかもしれません。おそらく決して使用しないその他のアイテム。ここでは多くのことが行われているので、各項目を説明するために最善を尽くします。特別なアクションは、サブメニューで整理されるのと同じ方法で以下に整理されます。

VoiceAttack Action
VoiceAttackアクション

'Make VoiceAttack Start Listening'
「VoiceAttackがリスニングを開始する」

Select this item if you want VoiceAttack to start listening. This is useful either by itself or in a macro. Note that there is a button the Main screen as well as a hotkey for this same action (See 'Options Screen' and 'VoiceAttack's Main Screen').
VoiceAttackでリスニングを開始する場合は、このアイテムを選択します。これは、単独でもマクロでも便利です。メイン画面にボタンがあり、この同じアクションのホットキーがあることに注意してください(「オプション画面」および「音声攻撃のメイン画面」を参照)。

Also, note that if a, 'Start Listening' action is found AS THE FIRST ACTION within a command, the entire sequence will be executed as if VoiceAttack is already, 'listening'.
また、「リスニングを開始」アクションがコマンド内の最初のアクションとして見つかった場合、VoiceAttackがすでに「リッスン」しているかのようにシーケンス全体が実行されることに注意してください。

'Make VoiceAttack Stop Listening'
「VoiceAttackのリスニングを停止する」

Same as above, except this stops VoiceAttack from listening.
上記と同じですが、これによりVoiceAttackがリスニングを停止する点が異なります。

'Make VoiceAttack Stop Processing All Commands'
「VoiceAttackですべてのコマンドの処理を停止する」

Use this to stop all currently-processing commands. This works the same way as clicking the 'Stop Commands' button on the Main Screen. Note that this command action will only execute if your currently-executing command allows for other commands to run at the same time (see, 'Command Screen' - 'This command allows other commands to run at the same time' option). Note this will also stop any playing sounds or text-to-speech, and any keys that are pressed down will be released.
これを使用して、現在処理中のすべてのコマンドを停止します。これは、メイン画面の「停止コマンド」ボタンをクリックするのと同じように機能します。このコマンドアクションは、現在実行中のコマンドで他のコマンドを同時に実行できる場合にのみ実行されることに注意してください(「コマンド画面」-「このコマンドで他のコマンドを同時に実行できる」オプションを参照)。これにより、再生中の音声や音声合成も停止し、押されたキーはすべて解除されます。

'Ignore an Unrecognized Word or Phrase'
「認識されない単語またはフレーズを無視する」

This is tricky to explain. This basically just discards the recognized command and does not report it in the recognition log. I added this feature because sometimes the speech recognition engine picks up background noise and breathing as commands. You will see entries in the recognition log like: "Unrecognized Command: 'if if if'".
これは説明が難しい。これは基本的に、認識されたコマンドを単に破棄し、認識ログに報告しません。この機能を追加したのは、音声認識エンジンがバックグラウンドノイズと呼吸をコマンドとして拾うことがあるためです。「認識されないコマンド: 'if if if'」のようなエントリが認識ログに表示されます。

Chances are, you will see things like this, too. Just add the irritating phrase as a command and select this option all by itself in the action sequence.
おそらく、あなたもこのようなものを見るでしょう。刺激的なフレーズをコマンドとして追加し、アクションシーケンスでこのオプションを単独で選択します。

'Switch to Another Profile'
「別のプロファイルへの切り替え」

This will allow you to issue a command to switch over to another profile without having to jump out of your application to do so. If you select the option, 'Switch Profile by
これにより、アプリケーションから飛び出さずに別のプロファイルに切り替えるコマンドを発行できます。オプションを選択した場合、「プロファイルの切り替え

Selection', you will be presented with a list of available profiles in which you can switch. If you choose the option, 'Switch by name', you can freely type the name of your profile in the input box. Note that when using this option, if your profile names change, this value will not be updated. Also note that the input box can contain any combination of tokens that can be rendered to form a valid profile name (this is an advanced feature for those of you that do not know the names of the target profile until run time).
選択」を選択すると、切り替え可能なプロファイルのリストが表示されます。[名前で切り替える]オプションを選択した場合、入力ボックスにプロファイルの名前を自由に入力できます。このオプションを使用する場合、プロファイル名が変更されても、この値は更新されないことに注意してください。また、入力ボックスには、有効なプロファイル名を形成するためにレンダリングできるトークンの任意の組み合わせを含めることができることに注意してください（これは、実行時までターゲットプロファイルの名前を知らないユーザー向けの高度な機能です）。

'Execute Another Command'
「別のコマンドを実行」

Selecting this will allow you to add a previously-created command to your command action list. Having nested commands keeps you from having to recreate entire action lists for duplicated functionality. Also, if any referenced command changes, the changes will be reflected in the nested commands.
これを選択すると、以前に作成したコマンドをコマンドアクションリストに追加できます。コマンドをネストすると、機能を複製するためにアクションリスト全体を再作成する必要がなくなります。また、参照されているコマンドが変更された場合、その変更はネストされたコマンドに反映されます。

There are two options for executing other commands. The first option allows you to select an existing command from a list. This is the safest way to execute other commands, since VoiceAttack knows ahead of time about any loops that may be encountered.
他のコマンドを実行するための2つのオプションがあります。最初のオプションでは、リストから既存のコマンドを選択できます。VoiceAttackは発生する可能性のあるループについて事前に知っているため、これは他のコマンドを実行する最も安全な方法です。

The second option (designated as an, 'advanced' feature) lets you select a command to execute by name (replacement tokens are supported). If a name of a command that exists is given, a simple loop check is done to make sure you will not potentially freeze up or crash VoiceAttack. If the referenced command does not exist (or if a replacement token is used) VoiceAttack will be unable to make the loop check, leaving you at risk for an infinite loop. Use this feature at your own peril:)
2番目のオプション(「高度な」機能として指定)では、名前で実行するコマンドを選択できます(置換トークンがサポートされています)。存在するコマンドの名前が指定されている場合、VoiceAttackがフリーズしたりクラッシュしたりしないように、単純なループチェックが行われます。参照されたコマンドが存在しない場合(または置換トークンが使用されている場合)、VoiceAttackはループチェックを行うことができず、無限ループの危険があります。あなた自身の危険でこの機能を使用してください:)

Note - If you are trying to execute a multipart/dynamic command by name, you must pick one of the commands that will be used (for instance, if you have a multipart command labeled, 'test;test [all;something]', you can just put, 'test' in the box.
注-マルチパート/ダイナミックコマンドを名前で実行しようとしている場合は、使用するコマンドの1つを選択する必要があります(たとえば、「test; test [all; something]」というラベルのマルチパートコマンドがある場合、ボックスに「テスト」と入力するだけです。

The, 'Wait until this command completes before continuing' option allows you to indicate whether or not subsequent actions will execute while the called command is running. If the box is checked, any commands that come after the executed command will have to wait until all the actions in the called command are finished processing.
[続行する前にこのコマンドが完了するまで待機する]オプションを使用すると、呼び出されたコマンドの実行中に後続のアクションを実行するかどうかを指定できます。ボックスがチェックされている場合、実行されたコマンドの後に来るコマンドは、呼び出されたコマンドのすべてのアクションの処理が完了するまで待機する必要があります。

'Command Queues - Enqueue Command'
「コマンドキュー-エンキューコマンド」

This action will allow you to enqueue a command so that it will execute in a specified order with other enqueued commands. Commands that are enqueued first are executed first, and subsequent commands that are added to the queue are processed next after the commands added prior are finished. This is a rather advanced action, so, for more details about command execution queues, see the section labeled, 'Command Execution Queues Overview' later in this document.
このアクションにより、コマンドをキューに登録して、キューに登録された他のコマンドで指定された順序で実行することができます。最初にエンキューされたコマンドが最初に実行され、キューに追加された後続のコマンドは、前に追加されたコマンドが終了した後に処理されます。これはかなり高度なアクションであるため、コマンド実行キューの詳細については、このドキュメントで後述する「コマンド実行キューの概要」というセクションを参照してください。

The, 'Queue Name' input will allow you to indicate the name of the command execution queue that you will be adding your command. If a queue by that name does not exist (that is, have a running instance), a new one will be established and that queue will remain available until VoiceAttack is closed. Note that you can simply type in a name into this box, or, you can select a queue name from the dropdown list. This input box will also accept any combination of tokens to establish a queue name. Also note that you can have as many command execution queues as your system will allow.
「キュー名」入力を使用すると、コマンドを追加するコマンド実行キューの名前を指定できます。その名前のキューが存在しない場合(つまり、実行中のインスタンスがある場合)、新しいキューが確立され、VoiceAttackが閉じられるまでそのキューは使用可能なままになります。このボックスに名前を入力するか、ドロップダウンリストからキュー名を選択できることに注意してください。この入力ボックスは、トークンの任意の組み合わせを受け入れて、キュー名を確立します。また、システムで許可されている数のコマンド実行キューを使用できることに注意してください。

Just like the, ‘Execute Another Command’ action above, there are two options for enqueueing commands. The first option allows you to select an existing command from a list. This is the safest way to execute other commands, since VoiceAttack knows ahead of time about any loops that may be encountered.
上記の「別のコマンドを実行」アクションと同様に、コマンドをキューに入れるための2つのオプションがあります。最初のオプションでは、リストから既存のコマンドを選択できます。VoiceAttackは発生する可能性のあるループについて事前に知っているため、これは他のコマンドを実行する最も安全な方法です。

The second option (designated as an, 'advanced' feature) lets you select a command to execute by name (replacement tokens are supported). If a name of a command that exists is given, a simple loop check is done to make sure you will not potentially freeze up or crash VoiceAttack. If the referenced command does not exist (or if a replacement token is used) VoiceAttack will be unable to make the loop check, leaving you at risk for an infinite loop. Use this feature at your own peril :)
2番目のオプション(「高度な」機能として指定)では、名前で実行するコマンドを選択できます(置換トークンがサポートされています)。存在するコマンドの名前が指定されている場合、VoiceAttackがフリーズしたりクラッシュしたりしないように、単純なループチェックが行われます。参照されたコマンドが存在しない場合(または置換トークンが使用されている場合)、VoiceAttackはループチェックを行うことができず、無限ループの危険があります。あなた自身の危険でこの機能を使用してください:)

Note – If you are trying to execute a multipart/dynamic command by name, you must pick one of the commands that will be used (for instance, if you have a multipart command labeled, 'test;test [all;something]', you can just put, 'test' in the box.
注–マルチパート/ダイナミックコマンドを名前で実行しようとしている場合は、使用するコマンドの1つを選択する必要があります(たとえば、「test; test [all; something]」というラベルのマルチパートコマンドがある場合、ボックスに「テスト」と入力するだけです。

The, ‘Start the queue when this command is added’ option is a convenience feature that will allow you to start the execution of commands in the queue immediately after the current command is enqueued. This saves you the extra step of having to explicitly add a ‘Start’ queue action (see below).
[このコマンドが追加されたときにキューを開始する]オプションは、現在のコマンドがキューに登録された直後にキュー内のコマンドの実行を開始できる便利な機能です。これにより、「開始」キューアクションを明示的に追加する必要のある余分な手順を節約できます(以下を参照)。

'Command Queues – Queue Action'
「コマンドキュー–キューアクション」

This action will allow you to invoke the various functions of your command execution queues, such as starting, stopping and pausing. You can perform your queue action against a specified queue or all queues at once.
このアクションにより、開始、停止、一時停止など、コマンド実行キューのさまざまな機能を呼び出すことができます。指定したキューまたはすべてのキューに対して一度にキューアクションを実行できます。

The, ‘Queue’ option will allow you to indicate a specific queue in which to control. Simply type in the name of the queue that you would like, or select its name from the dropdown list. Note that this input box will accept any number of tokens to resolve a queue name. Selecting the, ‘All Queues’ option will indicate that you would like the action to be performed against all queues. For instance, maybe you want to stop all of the queues you have running all at the same time.
「キュー」オプションを使用すると、制御する特定のキューを指定できます。希望するキューの名前を入力するか、ドロップダウンリストからその名前を選択します。この入力ボックスは、キュー名を解決するために任意の数のトークンを受け入れることに注意してください。[すべてのキュー]オプションを選択すると、すべてのキューに対してアクションを実行することを示します。たとえば、実行中のすべてのキューをすべて同時に停止したい場合があります。

Next, you will want to choose the action to perform on your queue:
次に、キューで実行するアクションを選択します。

Start – This will make your queue start executing commands in the order that they were added. You can also start your queue by selecting the, 'Start queue when this command is added' option when you enqueue any command. Tip – You can pre-fill a queue with commands and then start the queue at any time.
開始–これにより、キューはコマンドが追加された順序で実行を開始します。コマンドをキューに追加するときに、[このコマンドが追加されたときにキューを開始する]オプションを選択して、キューを開始することもできます。ヒント–コマンドをキューに事前に入力して、いつでもキューを開始できます。

Pause – This will tell the queue to pause command execution once the currently- executing command has completed. Tip – You can add commands to the queue even when the queue is paused.
一時停止–これは、現在実行中のコマンドが完了すると、コマンドの実行を一時停止するようキューに指示します。ヒント–キューが一時停止している場合でも、キューにコマンドを追加できます。

Unpause – This will get the queue executing commands again after it has been paused.
一時停止解除–これにより、一時停止されたコマンドを再度実行するキューが取得されます。

Toggle pause/unpause – This will pause or unpause, depending on the pause state (that is, it will unpause a paused queue, and pause an unpaused queue).
一時停止/一時停止の切り替え–一時停止状態に応じて一時停止または一時停止解除します（つまり、一時停止したキューの一時停止を解除し、一時停止していないキューを一時停止します）。

Stop – This stops all queue processing. This will halt the currently-executing command and then clear out any remaining commands that are contained within the queue. Note that a 'Stop all commands' action will also act as a Stop on all queues. Tip – You can add commands to the queue even when the queue is stopped.
停止–すべてのキュー処理を停止します。これにより、現在実行中のコマンドが停止し、キューに含まれている残りのコマンドがすべてクリアされます。「すべてのコマンドを停止」アクションは、すべてのキューで停止としても機能することに注意してください。ヒント–キューが停止している場合でも、キューにコマンドを追加できます。

Stop, but allow current command to complete – This will do everything the Stop action will do, except the queue will allow the currently-executing command to complete.
停止しますが、現在のコマンドの完了を許可します-これは、キューが現在実行中のコマンドの完了を許可することを除いて、停止アクションが行うすべてを行います。

'Enable / Disable Hotkeys'
「ホットキーの有効化/無効化」

Selecting these will allow you to turn on and off the keyboard hotkey shortcuts.
これらを選択すると、キーボードのホットキーショートカットをオンまたはオフにできます。

'Enable / Disable Mouse Shortcuts'
「マウスショートカットの有効化/無効化」

Selecting these will allow you to turn on and off mouse button shortcuts.
これらを選択すると、マウスボタンのショートカットをオンまたはオフにできます。

'Enable / Disable Joysticks'
「ジョイスティックの有効化/無効化」

Selecting these will allow you to turn on and off joystick button detection.
これらを選択すると、ジョイスティックボタンの検出をオンまたはオフにできます。

'Stop Another Command'
「別のコマンドを停止」

This option will let you specify a certain command that needs to be stopped. You will want to use this in conjunction with commands that loop or commands that have long- running macros. In earlier versions of VoiceAttack, the only way to stop running commands was to hit the, 'Stop Commands' button. Now you can indicate specific commands. NOTE – all instances of a command will be stopped, so, if you have multiple instances of a looping, asynchronous command, calling this will stop ALL instances.
このオプションを使用すると、停止する必要がある特定のコマンドを指定できます。これは、ループするコマンドまたは長時間実行されるマクロを持つコマンドと組み合わせて使用する必要があります。VoiceAttackの以前のバージョンでは、コマンドの実行を停止する唯一の方法は、[コマンドの停止]ボタンを押すことでした。これで、特定のコマンドを指定できます。注 -コマンドのすべてのインスタンスが停止するため、ループする非同期コマンドのインスタンスが複数ある場合、これを呼び出すとすべてのインスタンスが停止します。

'Quick Input'
「クイック入力」

This action will allow you to indicate text that you want typed out in your application. This differs from the recorder screen, as it allows you to include text tokens for replacement (see the section on tokens further down in this document). Simply type the text you want typed out in the, 'Text' box. You can then specify how long to hold down your keys by indicating a value in the, 'Hold down keys for X seconds' box, as well as specify how long to wait between keys by indicating a value in the, 'Wait for X seconds between keys' box. Note that a value of zero for either of these delays is not recommended for DirectX games, as they tend to rely on polling for key state). In order to represent keys such as, 'Enter', 'Tab', 'F1', etc., you can use some special indicators enclosed in square brackets: [ ]. For instance, if you want to press the 'Enter' key, simply include [ENTER] in your text. Some keys, such as 'Shift', 'Alt, 'Ctrl' and 'Windows' need to be held down while you type other characters. There are some reserved indicators for this as well. As an example, for the, 'Shift' key, [SHIFTDOWN] and [SHIFTUP] are provided. If you need to specify a pause
このアクションにより、アプリケーションに入力するテキストを指定できます。これは、置換用のテキストトークンを含めることができるため、レコーダー画面とは異なります(このドキュメントのさらに下のトークンに関するセクションを参照してください)。[テキスト]ボックスに入力するテキストを入力します。次に、「X秒間キーを押したままにする」ボックスに値を示すことでキーを押し続ける時間を指定できます。また、「X秒間待機する」に値を示すことでキー間の待機時間を指定できますキー間ボックス。DirectXゲームでは、キー状態のポーリングに依存する傾向があるため、これらの遅延のいずれかの値をゼロにしないことをお勧めします。「Enter」、「Tab」、「F1」などのキーを表すために、角括弧で囲まれたいくつかの特別なインジケータを使用できます:[]。例えば、「Enter」キーを押したい場合は、単に[ENTER]をテキストに含めてください。「Shift」、「Alt」、「Ctrl」、「Windows」などの一部のキーは、他の文字を入力するときに押したままにする必要があります。これにはいくつかの予約済みのインジケータもあります。例として、「Shift」キーの場合、[SHIFTDOWN]と[SHIFTUP]が提供されます。一時停止を指定する必要がある場合

between keys, you can use the [PAUSE:seconds] indicator, where seconds is the number of seconds to pause (Ex: [PAUSE:0.5] will pause one half a second, and [PAUSE:2.5] will pause for two and a half seconds).
:キーの間は、[PAUSE使用でき秒インジケータ、】秒 2と一時停止されます半分秒、および[2.5 PAUSE]を一時停止する(実施例を一時停止する秒数である:[0.5 PAUSE]を半秒)。

Adding the following text to the Quick Input, 'Text' box:
次のテキストをクイック入力の「テキスト」ボックスに追加します。

Hello, out there![ENTER][ENTER]How are you? Produces the following output:
こんにちは。[ENTER] [ENTER]お元気ですか?次の出力を生成します。

Hello, out there!
こんにちは!

How are you?
お元気ですか?

The full list of Quick Input key indicators is near the end of this document.
クイック入力キーインジケータの完全なリストは、このドキュメントの終わり近くにあります。

Note: Key indicators are not case-sensitive.
注:キーインジケータは大文字と小文字を区別しません。

'Reset the Active Profile'
「アクティブなプロファイルをリセットする」

This action will reload the current profile. This is typically not something you'll want or even need to do for the most part, but is available for those that need it for some more advanced application (it's why the notes about this kinda wander off into the obscure o_O). When the profile is reloaded, any profile-scoped variables (variables prefixed with ONE '>') will be cleared. Persistent profile variables (variables prefixed with '>>') will be retained (see section below regarding variable scope if you're needing more info on that). All commands will be reloaded and any tokenized, 'when I say' phrases will be reevaluated. The speech engine will also be reloaded (of course), and any executing commands will be stopped. As you can see, this is whatever a, 'change profile' action does, just without actually changing ;)
このアクションにより、現在のプロファイルが再ロードされます。これは通常、ほとんどの部分で必要なことでも必要なことでもありませんが、より高度なアプリケーションで必要な場合に利用できます（このため、この種のメモはあいまいなo_Oに移動します）。プロファイルが再ロードされると、プロファイルスコープの変数（接頭辞が1つの '>'の変数）がクリアされます。永続的なプロファイル変数（「>>」で始まる変数）は保持されます（詳細については、変数スコープに関する以下のセクションを参照してください）。すべてのコマンドがリロードされ、トークン化された「私が言うとき」フレーズが再評価されます。スピーチエンジンも（もちろん）リロードされ、実行中のコマンドはすべて停止されます。ご覧のとおり、これは「プロファイルの変更」アクションが実行するものであり、

'Refresh Variable Hotkeys'
「変数ホットキーの更新」

This action works in conjunction with the, 'Use variable hotkeys' feature (see the, 'Command Screen' section for more information on setting up variable hotkeys). What this action does is refresh the hotkeys that VoiceAttack is monitoring based on the current state of the variables that the variable hotkeys are using. For instance, let's say we have a command using a variable hot key, and the variable that is being used is, 'myTextVariable'. If, 'myTextVariable's value changes, VoiceAttack will not be aware of this change until you execute this action. Note that this is an advanced feature.
このアクションは、「変数ホットキーを使用する」機能と連動します（変数ホットキーの設定の詳細については、「コマンド画面」セクションを参照してください）。このアクションは、VoiceAttackが監視しているホットキーを、変数ホットキーが使用している変数の現在の状態に基づいて更新します。たとえば、変数ホットキーを使用するコマンドがあり、使用されている変数が「myTextVariable」であるとします。「myTextVariable」の値が変更された場合、VoiceAttackはこのアクションを実行するまでこの変更を認識しません。これは高度な機能であることに注意してください。

Sounds
音

'Say Something with Text-To-Speech'.
「テキスト読み上げで何かを言う」。

Type in a phrase to be spoken by your built-in Text to Speech engine.
組み込みのテキスト読み上げエンジンで読み上げるフレーズを入力します。

Something fun you can do with this is input several phrases at once, separated by a semicolon and VoiceAttack will randomly pick a phrase to, 'say'. For example, you can input, 'Fire Weapons;Fire At Will;Destroy Them All' and VoiceAttack will see this as three random phrases for the same command.
これでできることは、セミコロンで区切られた複数のフレーズを一度に入力すると、VoiceAttackがランダムに「say」にフレーズを選択することです。たとえば、「Fire Weapons; Fire At Will; Destroy Them All」と入力すると、VoiceAttackはこれを同じコマンドの3つのランダムなフレーズとして表示します。

If you need further dynamic responses, you can include them by putting the responses between square brackets (just like dynamic spoken command phrases from the Command screen). Putting text between square brackets in text to speech is called a, 'dynamic response section'.
さらに動的な応答が必要な場合は、応答を角かっこで囲んで含めることができます（コマンド画面の動的な音声コマンドフレーズのように）。スピーチへのテキストの角括弧の間にテキストを置くことは、「動的応答セクション」と呼ばれます。

Dynamic response sections allow you to specify a part of your text to speech (TTS)
動的応答セクションでは、テキストの読み上げ（TTS）の一部を指定できます

that may vary. Sometimes you may want TTS to say, 'Hello captain' and sometimes you may want it to say, 'Greetings, captain'. To indicate that you want to use a dynamic response section, enclose the section in square brackets: [ ], with each element separated by a semicolon. Your TTS phrase may look something like this:
それは異なる場合があります。TTSに「ハローキャプテン」と言ってもらいたいときもあれば、「挨拶、キャプテン」と言ってほしいときもあります。動的応答セクションを使用することを示すには、セクションを角括弧[]で囲み、各要素をセミコロンで区切ります。TTSフレーズは次のようになります。

[Hello;Greetings]captain
[Hello; Greetings]キャプテン

This will result in either, 'Hello captain' or, 'Greetings captain'.
これにより、「Helloキャプテン」または「Greetingsキャプテン」が表示されます。

Note that you can still add phrases separated with a semicolon: [Greetings;Hello]captain;Hi
セミコロンで区切られたフレーズを追加できることに注意してください：[Greetings; Hello] captain; Hi

With this example, the result will be either 'Greetings captain', 'Hello captain' or 'Hi'.
この例では、結果は「Greetingsキャプテン」、「Helloキャプテン」または「こんにちは」のいずれかになります。

The dynamic response sections don't have to just be at the beginning. They can be anywhere in the phrase. Also, as a side-effect, if you put a semicolon on at the end of the selections, it makes the section optional:
動的応答セクションは、最初にある必要はありません。フレーズのどこにでも使用できます。また、副作用として、選択の最後にセミコロンを付けると、セクションがオプションになります。

[Greetings;Hello]captain[how are you;] This results in a response of: Greetings captain how are you
[Greetings; Hello] captain [how are you;]これにより、応答が返されます。

Hello captain how are you Greetings captain
こんにちはキャプテン、元気ですか？

Hello captain
こんにちはキャプテン

Note that there is a semicolon after 'how are you' to indicate that the entire section is optional.
「how are you」の後にセミコロンがあり、セクション全体がオプションであることを示すことに注意してください。

Something to consider when using this feature is that you can create a lot of permutations from very few words. Use with care :)
この機能を使用する際に考慮すべきことは、ごくわずかな単語から多くの順列を作成できることです。注意して使用してください:)

Dynamic response sections can also contain numeric ranges. To indicate a numeric range in a dynamic response section, just include the minimum and maximum values separated by an ellipsis (..). Not sure how many applications this may have, but it's there for you (it's available for dynamic commands… just thought I'd leave it in).
動的応答セクションには数値範囲を含めることもできます。動的応答セクションで数値範囲を示すには、最小値と最大値を省略記号(..)で区切って含めるだけです。これがどのくらいのアプリケーションを持っているかはわかりませんが、それはあなたのためにあります(動的コマンドで利用可能です...そのままにしておきたいと思っただけです)。

A bad example would be [Greetings;Hello]captain. I tried to call you [2..10] times today This will include responses that look like this:
悪い例は[Greetings; Hello] captainです。今日は[2..10]回お電話しようとしましたが、これには次のような応答が含まれます。

Hello captain. I tried to call you 5 times today Greetings captain. I tried to call you 10 times today.
こんにちはキャプテン。今日は5回お電話しようとしました。今日は10回電話しました。

Note that you can preview and set the volume and voice rate of your phrase from this panel.
このパネルからフレーズの音量と音声レートをプレビューして設定できることに注意してください。

The, ‘Voice’ drop down box will allow you to select the speaking voice that you would like to hear when your text is spoken. Note that you can also type freely into this box.
[音声]ドロップダウンボックスを使用すると、テキストが話されたときに聞きたい音声を選択できます。このボックスに自由に入力することもできます。

What can be typed in the box are text variable names, literal text and/or any combination of tokens. Note that whenever whatever was typed in is resolved, it must match an installed voice name exactly, otherwise the default voice will be used.
ボックスに入力できるのは、テキスト変数名、リテラルテキスト、および/またはトークンの任意の組み合わせです。入力されたものが解決される場合は常に、インストールされている音声名と正確に一致する必要があります。一致しない場合、デフォルトの音声が使用されます。

NEW (Advanced): If, ‘Default’ is selected as the, ‘Voice’ value (or, if a token or variable name does not resolve to a valid voice name as indicated above), the text-to-speech voice selected in Windows’ Control Panel will be used. You can override this voice by opening the ‘Profile Options’ screen and selecting a voice from the, ‘‘Default Text-to- speech voice’ option. See, ‘Profile Options’ screen for more details.
新規(高度):「音声」値として「デフォルト」が選択されている場合(または、トークンまたは変数名が上記のように有効な音声名に解決されない場合)、選択されたテキスト読み上げ音声Windowsのコントロールパネルが使用されます。「プロファイルオプション」画面を開き、「デフォルトのテキスト読み上げ音声」オプションから音声を選択すると、この音声を上書きできます。詳細については、「プロファイルオプション」画面を参照してください。

Speech Synthesis Markup Language (SSML) is supported if you want to do some more fancy stuff. Visit Microsoft’s site for more information on SSML: http://bit.ly/1PisKMD.
音声合成マークアップ言語(SSML)は、もっと凝ったことをしたい場合にサポートされています。SSMLの詳細については、Microsoftのサイト(http://bit.ly/1PisKMD)をご覧ください。

Certain tokens can be used with the Text-To-Speech phrases. See the section way down at the end titled, ’Text-To-Speech Tokens’.
特定のトークンは、音声合成フレーズで使用できます。「テキスト読み上げトークン」というタイトルの最後のセクションを参照してください。

Another advanced item is the Text-To-Speech Channel feature. If you are using the, ‘Integrated Components’ audio output type option (Options screen, audio tab), you will be given the ability to route the Text-To-Speech audio out of a selected audio playback channel. Simply select the audio channel from the dropdown list where you would like the audio to be routed. Selecting, ‘Default’ for this feature will not route the audio, and Text-To-Speech audio will be rendered through the default audio playback device as specified in Control Panel.
もう1つの高度な項目は、音声合成チャネル機能です。[統合コンポーネント]オーディオ出力タイプオプション([オプション]画面、[オーディオ]タブ)を使用している場合は、選択したオーディオ再生チャネルからテキスト読み上げオーディオをルーティングする機能が提供されます。オーディオをルーティングするオーディオチャンネルをドロップダウンリストから選択します。この機能の[デフォルト]を選択すると、オーディオはルーティングされず、コントロールパネルで指定されたデフォルトのオーディオ再生デバイスを介して音声合成テキストがレンダリングされます。

There are two options available for speech execution. Checking the, 'Wait until speech completes' option will hold up the executing command until the speech is finished.
スピーチの実行には2つのオプションがあります。[読み上げが完了するまで待機する]オプションをオンにすると、読み上げが終了するまで実行中のコマンドが保持されます。

Checking, 'This completes all other text-to-speech' will stop any other speech that is currently running. Note that it says, 'complete' rather than, 'stop' or, 'interrupt'. Any commands that are currently waiting on speech to finish will immediately resume (as their pending speech actions would then be, 'completed').
「これで他のすべてのテキスト読み上げが完了します」をチェックすると、現在実行中の他の読み上げが停止します。「停止」または「中断」ではなく、「完了」と表示されていることに注意してください。スピーチの終了を現在待機しているコマンドはすぐに再開されます（保留中のスピーチアクションが「完了」するため）。

The, 'Mass update' option is an advanced option that allows you to update all, 'Say Something with Text-to-Speech' actions in the current profile. Whatever voice that was originally selected will be updated to the newly-selected voice. So, if you change the voice from, 'Default' to 'Microsoft Hazel' and select the, 'Mass update voices' option, all actions in your profile that currently use the, 'Default' text-to-speech voice will be updated to, 'Microsoft Hazel'. Also, any action that contains the originally-selected volume and/or rate, will be updated to reflect the currently selected volume and/or rate. So, if you change the volume of, 'Default' from 100 to 90 and select, 'Mass Update', any, 'Say Something with Text-to-speech' actions that are using the, 'Default' voice and have a volume of 100 will be set to 90. All others will be ignored.
[一括更新]オプションは、現在のプロファイルの[テキスト読み上げで何かを言う]アクションをすべて更新できる高度なオプションです。最初に選択された音声はすべて、新しく選択された音声に更新されます。したがって、音声を「デフォルト」から「Microsoft Hazel」に変更し、「音声の一括更新」オプションを選択すると、現在「デフォルト」のテキスト読み上げ音声を使用しているプロファイル内のすべてのアクションが更新されます「Microsoft Hazel」に。また、最初に選択されたボリュームおよび/またはレートを含むアクションは、現在選択されているボリュームおよび/またはレートを反映するように更新されます。そのため、「デフォルト」の音量を100から90に変更し、「一括更新」、任意、「テキスト読み上げで何かを言う」アクション、「デフォルト」の音声を使用して音量を選択する場合100の値は90に設定されます。他のすべては無視されます。

Also note that hitting, 'Cancel' on this screen will not cancel a mass update. You will need to press, 'Cancel' on the Profile edit screen in order to cancel the operation, as the entire profile will be updated.
また、この画面で「キャンセル」を押しても、一括更新はキャンセルされません。プロファイル全体が更新されるため、操作をキャンセルするには、プロファイル編集画面で「キャンセル」を押す必要があります。

'Play a Sound'
「サウンドを再生する」

This feature simply plays a sound file that you choose. VoiceAttack has three audio output types that you can select from in order to play your sounds. The audio output type can be selected from the Options screen on the, 'Audio' tab (see, 'Audio Output' in the Options screen section later in this document for descriptions of each type). The reason this is important is because there are certain rules and various options to consider that are available for each output type.
この機能は、選択したサウンドファイルを再生するだけです。VoiceAttackには、サウンドを再生するために選択できる3つのオーディオ出力タイプがあります。オーディオ出力タイプは、「オーディオ」タブのオプション画面から選択できます（各タイプの説明については、このドキュメントで後述する「オプション画面」セクションの「オーディオ出力」を参照してください）。これが重要な理由は、特定のルールと考慮すべきさまざまなオプションがあり、各出力タイプで使用できるためです。

When selecting, ‘Legacy Audio’ as your audio output type, you can only play .wav files. That’s about it. You can’t set the volume, balance or channel, plus you will not be able to have VoiceAttack wait for the sound to complete (legacy audio is always played asynchronously in the executing command). Why is this even available? Well, it’s there in case you need it. This was the first method that VoiceAttack employed for playing audio way back at the beginning and was left in just in case the other output types just will not work for you (doesn’t hurt to leave it in, right?).
オーディオ出力タイプとして「レガシーオーディオ」を選択すると、.wavファイルのみを再生できます。それについてです。ボリューム、バランス、またはチャンネルを設定することはできません。また、VoiceAttackがサウンドの完了を待つことはできません（レガシーオーディオは常に実行コマンドで非同期に再生されます）。なぜこれも利用できるのですか？必要な場合に備えてあります。これは、VoiceAttackが最初にオーディオを再生するために採用した最初の方法であり、他の出力タイプがうまく機能しない場合に備えて残されました（そのままにしておいても問題ありませんか？）。

If, ‘Windows Media Components’ is selected as the audio output type, you have a wide variety of file types that you can play. You can also adjust the volume or set the start and end positions. This was the second audio output type added to VoiceAttack, and was left in not only to ensure backward-compatibility, but also because the components work extremely well with files that just won’t seem to play otherwise.
オーディオ出力タイプとして「Windows Media Components」が選択されている場合、再生できるファイルタイプは多種多様です。ボリュームを調整したり、開始位置と終了位置を設定することもできます。これはVoiceAttackに追加された2番目のオーディオ出力タイプであり、下位互換性を確保するためだけでなく、コンポーネントが他の方法では再生されないように見えるファイルでも非常にうまく機能するために残されました。

When, ‘Integrated Components’ is selected as the audio output type, you will have access to all of the available, ‘Play a Sound’ options, as well as a wide variety of file types to play. The only drawback is that this mode may be a little bit pickier about the audio files that it plays, so make sure you preview your sound to make sure it’s going to work for you.
オーディオ出力タイプとして「統合コンポーネント」を選択すると、利用可能なすべての「サウンドの再生」オプションに加えて、再生するさまざまなファイルタイプにアクセスできます。唯一の欠点は、このモードが再生するオーディオファイルについて少しうるさいことです。そのため、サウンドをプレビューして、動作することを確認してください。

The, ‘Play a Sound’ feature works much like the 'Run an application' feature above. Click the file browser ('…') button to select a sound file (note again that legacy audio mode is restricted to .wav files, and that ‘Integrated Components’ or ‘Windows Media Components’ allow you to play .wav, .wma and .mp3 (also .ogg, .flac, .m4a and .aac if you have the proper codecs installed) files). Note that this input box will also accept any combination of replacement tokens (See, the ‘Text (and Text-to-Speech) Tokens’ later in this document for more details).
「サウンドの再生」機能は、上記の「アプリケーションの実行」機能とよく似ています。ファイルブラウザー（「…」）ボタンをクリックしてサウンドファイルを選択します（レガシーオーディオモードは.wavファイルに制限され、「統合コンポーネント」または「Windows Mediaコンポーネント」を使用すると.wav、.wmaを再生できます。および.mp3（適切なコーデックがインストールされている場合は.ogg、.flac、.m4a、および.aac）ファイル）。この入力ボックスは、置換トークンの任意の組み合わせも受け入れることに注意してください（詳細については、このドキュメントで後述する「テキスト（およびテキスト読み上げ）トークン」を参照してください）。

You can preview your sound file by clicking the, 'Preview' button.
[プレビュー]ボタンをクリックして、サウンドファイルをプレビューできます。

The options available for both ‘Integrated Components’ and ‘Windows Media Components’ audio output types include being able to select the volume of the sound you are playing. You also have several additional options. The first is, 'Wait until sound completes before continuing'. This will hold up the containing command until the audio finishes. The next option is, 'This completes all other sounds'. This will stop any other sound that is currently playing. Note that it says, 'complete' rather than, 'stop' or, 'interrupt'. Any commands that are currently waiting on sounds to finish will immediately resume (as their pending sound actions would then be, 'completed'). The third option is the, ‘Begin at position’ option. This will allow you to start the sound playback at a certain number of seconds, expressed as a decimal value. Note that this
「統合コンポーネント」と「Windows Mediaコンポーネント」の両方のオーディオ出力タイプで利用可能なオプションには、再生しているサウンドの音量を選択できることが含まれます。また、いくつかの追加オプションがあります。最初は、「続行する前に音が完了するまで待機する」です。これは、オーディオが終了するまで、包含コマンドを保持します。次のオプションは、「これは他のすべてのサウンドを完了します」です。これにより、現在再生中の他のサウンドが停止します。「停止」または「中断」ではなく、「完了」と表示されていることに注意してください。現在、サウンドの終了を待機しているコマンドはすぐに再開されます（保留中のサウンドアクションが「完了」するため）。3番目のオプションは、’ 位置から開始’オプション。これにより、特定の秒数でサウンドの再生を開始できます（10進数値で表されます）。これに注意してください

box can accept a decimal variable name as well as a token that resolves to a decimal value. See also the, ‘{STATE_AUDIOCOUNT}’ and, ‘{STATE_AUDIOPOSITION}’
boxは、10進数の変数名と、10進数の値に解決されるトークンを受け入れることができます。
「{STATE_AUDIOCOUNT}」および「{STATE_AUDIOPOSITION}」も参照してください

tokens later in this document. Also note that, ‘Begin at position’ is not available when using Legacy Audio as your selected output type. If you choose to start your audio at a certain position, the, ‘Fade in’ option will become available. Fading in may help make the audio sound better when started in positions that are loud. Just a very minor enhancement… no big deal ;) The fourth option is, ‘End at position’. This works exactly like the, ‘Begin at position’ option above, except this option marks where you want the audio to end within the playing sound (also, you have a, ‘fade out’ option instead of, ‘fade in’).
このドキュメントの後半のトークン。また、選択した出力タイプとしてレガシーオーディオを使用している場合、「位置で開始」は使用できません。特定の位置でオーディオを開始することを選択した場合、「フェードイン」オプションが利用可能になります。フェードインは、大きな位置で開始したときにオーディオの音を良くするのに役立ちます。ほんの少しの機能強化…大したことはありません;）4番目のオプションは、「位置で終了」です。これは、上記の「Begin at position」オプションとまったく同じように機能しますが、このオプションは再生サウンド内でオーディオを終了する場所をマークします（また、「fade in」ではなく「fade out」オプションがあります）。

The options available only to the, ‘Integrated Components’ audio output type is Balance and Channel. Balance allows you to adjust the playback audio to your left or right speaker. For example, sliding to the left will increase the volume in the left speaker, and decrease the volume in the right. Channel allows you to choose the device on which your audio will be played. So, if you would like a certain sound only played back through your desktop speakers, you can choose to do that
「統合コンポーネント」オーディオ出力タイプでのみ使用可能なオプションは、バランスとチャンネルです。バランスを使用すると、左または右のスピーカーの再生オーディオを調整できます。たとえば、左にスライドすると左スピーカーの音量が上がり、右スピーカーの音量が下がります。チャンネルを使用すると、オーディオを再生するデバイスを選択できます。そのため、特定のサウンドをデスクトップスピーカーからのみ再生したい場合は、それを選択できます。

here. Choosing, 'Default' will play the audio back through the default playback device specified by Windows.
ここに。「デフォルト」を選択すると、Windowsで指定されたデフォルトの再生デバイスでオーディオが再生されます。

The, ‘Variable Volume’ feature of this screen is available as an advanced option to allow you to specify a variable that sets the volume for the played sound. If the value in this box can be resolved to an integer value, that value will be used to override what is set by using the volume slider higher up on the screen. The value must be between 0 and 100, with zero being no sound and 100 being full volume. The input box can take a literal value, a variable name or any combination of tokens that will resolve to a valid value. Note that since this option is command-dependent, ‘Preview’ will not resolve this value from this screen.
この画面の「可変音量」機能は、再生音の音量を設定する変数を指定できる詳細オプションとして利用できます。このボックスの値を整数値に解決できる場合、その値は、画面上のボリュームスライダーを使用して設定された値を上書きするために使用されます。値は0から100の間である必要があります。ゼロは無音で、100は最大音量です。入力ボックスには、リテラル値、変数名、または有効な値に解決されるトークンの任意の組み合わせを指定できます。このオプションはコマンド依存であるため、「プレビュー」はこの画面からこの値を解決しないことに注意してください。

The, ‘Variable Balance’ feature of this screen is available as another advanced option to allow you to specify a variable to set the balance for the played sound. If the value in this box can be resolved to an integer value, that value will be used to override what is set by using the balance slider higher up on the screen. The value must be between
この画面の「変数バランス」機能は、再生音のバランスを設定する変数を指定できる別の高度なオプションとして利用できます。このボックスの値を整数値に解決できる場合、その値を使用して、画面上部のバランススライダーを使用して設定された値を上書きします。値は

-100 and 100, with -100 being full left, 100 being full right and zero being center (full left and right). The input box can take a literal value, a variable name or any combination of tokens that will resolve to a valid value. Note that since this option is command-dependent, ‘Preview’ will not resolve this value from this screen.
-100および100。-100は左いっぱい、100は右いっぱい、ゼロは中央（左いっぱいおよび右いっぱい）です。入力ボックスには、リテラル値、変数名、または有効な値に解決されるトークンの任意の組み合わせを指定できます。このオプションはコマンド依存であるため、「プレビュー」はこの画面からこの値を解決しないことに注意してください。

Note: If you are unable to hear your sound file, you can always go into the Options screen and try out a different audio output type from the, ‘Audio’ tab.
注：サウンドファイルを聞くことができない場合は、いつでも[オプション]画面に移動して、[オーディオ]タブから別のオーディオ出力タイプを試すことができます。

'Play a Random Sound'
「ランダムな音を再生する」

This action will allow you to make a selection of sounds that will play randomly. You can choose to select individual files or entire directories of files.
このアクションにより、ランダムに再生されるサウンドを選択できます。個々のファイルまたはファイルのディレクトリ全体を選択することができます。

To add individual sound files to the list, click on the radio button titled, 'Play a random sound file from a list' then click the, 'Add New' button. You will then be presented with the same interface as found in the, 'Play a Sound' action (see above). For every sound
個々のサウンドファイルをリストに追加するには、「リストからランダムなサウンドファイルを再生する」というタイトルのラジオボタンをクリックしてから、「新規追加」ボタンをクリックします。その後、「サウンドの再生」アクションで見られるのと同じインターフェースが表示されます（上記を参照）。すべての音について

file selected, you will be able to choose its volume and whether or not it holds up the macro until it completes or stops other sounds from playing. To edit a sound, just click on the, 'Edit' button (or double-click the item in the list). To remove a sound, click the, 'Remove' button. Note that you can select multiple files at once for adding,
ファイルを選択すると、そのボリュームと、他のサウンドの再生が完了するか停止するまでマクロを保持するかどうかを選択できます。サウンドを編集するには、[編集]ボタンをクリックするか、リスト内のアイテムをダブルクリックします。サウンドを削除するには、[削除]ボタンをクリックします。追加する複数のファイルを一度に選択できることに注意してください。

editing or removing.
編集または削除。

To play a random sound out of a directory of sounds, choose the option labeled, 'Play a random sound from a directory. You will then be able to choose the desired directory, plus set the options for all the files that will be played (note that the options apply to all files in the directory. If your files need to have their own attributes, you will need to use the first option above). All supported files (.wav, .mp3, .ogg, .flc, .m4a, .aac) will be used out of the selected directory as well as shortcut files (.lnk) that are for sound files.
サウンドのディレクトリからランダムなサウンドを再生するには、「ディレクトリからランダムなサウンドを再生します。次に、目的のディレクトリを選択し、再生するすべてのファイルのオプションを設定できます（オプションはディレクトリ内のすべてのファイルに適用されることに注意してください。ファイルに独自の属性が必要な場合は、上記の最初のオプションを使用してください）。サポートされているすべてのファイル（.wav、.mp3、.ogg、.flc、.m4a、.aac）は、選択したディレクトリと、サウンドファイル用のショートカットファイル（.lnk）から使用されます。

As with the, 'Play a Sound' action up above, you can choose the audio channel that your randomly-played sound will be rendered. Just select the desired output channel from the dropdown list. See the, 'Play a Sound' action for more info on selecting an audio channel.
上記の「サウンドの再生」アクションと同様に、ランダムに再生されるサウンドがレンダリングされるオーディオチャンネルを選択できます。ドロップダウンリストから目的の出力チャネルを選択するだけです。オーディオチャンネルの選択の詳細については、「サウンドの再生」アクションを参照してください。

To minimize the chance of constantly repeating sounds, select the, 'Suppress Repeat' option at the bottom.
音が絶えず繰り返される可能性を最小限に抑えるには、下部にある[繰り返しの抑制]オプションを選択します。

'Captured Audio'
「キャプチャされたオーディオ」

For mostly fun, or to possibly diagnose a mic problem, there is the, ‘Captured Audio’ feature. What this does is play back or save the audio that VoiceAttack receives as input. There are several types of audio you can play back (selected from the, ‘Type’ list):
主に楽しみのために、またはマイクの問題を診断するために、「Captured Audio」機能があります。これは、VoiceAttackが入力として受信するオーディオを再生または保存することです。再生できるオーディオにはいくつかのタイプがあります（「タイプ」リストから選択）。

Recognized Audio – This is the current recognized audio. So, if you say, ‘fire weapons’ and this action is in your command, you will hear your very own voice say, ‘fire weapons’. Yup… you sound that bad, for reals.
認識された音声–これは現在認識されている音声です。したがって、「武器を発射する」と言い、このアクションがあなたの指揮下にある場合、あなた自身の声が「武器を発射する」と言うのが聞こえます。うん…あなたは本当の、それは悪い音。

Previous Recognized Audio – This is the recognized audio prior to the current recognized audio. This is so you can issue a voice command and hear the horrible thing you just said (and not the current horrible thing you just said).
以前に認識された音声–これは、現在認識されている音声の前に認識された音声です。これは、音声コマンドを発行して、あなたが今言った恐ろしいことを聞くことができるようにするためです（あなたがちょうど今言った恐ろしいことではありません）。

Unrecognized Audio – This is input that was considered by VoiceAttack to be, ‘Unrecognized’. For me this seems to always be a lot of seemingly aggressive typing sounds.
認識されないオーディオ–これは、VoiceAttackによって「認識されない」と見なされた入力です。私にとって、これは常に一見攻撃的なタイピングサウンドのようです。

Captured Audio – This is the latest non-dictation input that is recognized or unrecognized.
キャプチャされたオーディオ–これは、認識または認識されない最新のディクテーションなしの入力です。

Previous Captured Audio – This is the second-to-latest audio input. Again, just in case you want to get this audio using a voice command.
以前にキャプチャしたオーディオ–これは、2番目から最後のオーディオ入力です。繰り返しますが、音声コマンドを使用してこの音声を取得する場合に備えてください。

Dictation Audio – This is the audio captured when dictation mode is enabled.
ディクテーションオーディオ–ディクテーションモードが有効になっているときにキャプチャされるオーディオです。

There are some options available when playing back the captured audio when using, ‘Integrated Components’ (Options screen). You are able to choose the volume and output channel, as well as be able to have the command wait until the audio is finished. When the audio is played, you have to option to complete (stop) all other audio as well. The last option for this feature is to be able to save the captured audio to a file. Select the, ‘Save the captured audio’ option and then indicate the path where you would like the file saved. Note that this box accepts any combination of text and tokens. This
「統合コンポーネント」（オプション画面）を使用しているときに、キャプチャしたオーディオを再生するときに使用できるオプションがいくつかあります。音量と出力チャンネルを選択できます。また、音声が終了するまでコマンドを待機させることもできます。オーディオが再生されると、他のすべてのオーディオも同様に完了（停止）するオプションが必要です。この機能の最後のオプションは、キャプチャしたオーディオをファイルに保存できるようにすることです。[キャプチャしたオーディオを保存する]オプションを選択し、ファイルを保存するパスを指定します。このボックスは、テキストとトークンの任意の組み合わせを受け入れることに注意してください。この

feature will NOT overwrite a file that exists. If you want to save with unique file names, make sure to check out the {TIMESTAMP} and {GUID} tokens later in this document.
この機能は、存在するファイルを上書きしません。一意のファイル名で保存する場合は、このドキュメントの後半で{TIMESTAMP}および{GUID}トークンを必ず確認してください。

Note also that the file that is saved is in .wav format only.
また、保存されるファイルは.wav形式のみであることにも注意してください。

Windows
窓

'Run an Application'
「アプリケーションを実行する」

Select this option if you want VoiceAttack to launch a program. There are a few extra input boxes with this action:
VoiceAttackでプログラムを起動する場合は、このオプションを選択します。このアクションには、追加の入力ボックスがいくつかあります。

Path of the program to be run
実行するプログラムのパス

This is what program will be launched by VoiceAttack. You can click on the file browser button (labeled, '...') to browse for a file.
これは、VoiceAttackによって起動されるプログラムです。ファイルブラウザボタン(「...」というラベル)をクリックして、ファイルを参照できます。

Note: You can drag and drop files to this input box (edit: maybe... depends on the security of your system).
注:この入力ボックスにファイルをドラッグアンドドロップできます(編集:システムのセキュリティに依存する場合があります)。

Note: I won't go into much detail, but, you can also execute shell commands with Windows Vista and later (type in a value like, 'shell:MyComputerFolder'). Check out the web for more details on shell commands (very handy).
注:詳細は説明しませんが、Windows Vista以降でシェルコマンドを実行することもできます(「shell:MyComputerFolder」などの値を入力します)。シェルコマンドの詳細については、Webをご覧ください(非常に便利です)。

File browser ('...') button
ファイルブラウザ('...')ボタン

Use this button to select a file to launch. This doesn't have to be just executable (.exe) files for games and apps. This can also be files that Windows have associated with other applications (for example, launching, 'helloWorld.txt' would launch the Windows Notepad.exe program and load the, 'helloWorld.txt' file. Kinda handy....
このボタンを使用して、起動するファイルを選択します。これは、ゲームやアプリの実行可能(.exe)ファイルである必要はありません。これは、Windowsが他のアプリケーションに関連付けたファイルでもあります(たとえば、起動すると、「helloWorld.txt」はWindows Notepad.exeプログラムを起動し、「helloWorld.txt」ファイルをロードします。

'With these parameters' box
「これらのパラメーターを使用」ボックス

If your launched program needs extra run-time parameters, you can specify them here. The example shows us launching Notepad with 'c:¥mytext.txt' as the parameter. When this action is activated, VoiceAttack launches Notepad and displays the contents of, 'mytext.txt'.
起動したプログラムに追加のランタイムパラメーターが必要な場合は、ここで指定できます。この例では、パラメーターとして「c:¥ mytext.txt」を使用してメモ帳を起動しています。このアクションがアクティブになると、VoiceAttackはメモ帳を起動し、「mytext.txt」の内容を表示します。

'In this working directory' box
「この作業ディレクトリ内」ボックス

Specify the working directory for your launched application here.
ここで、起動したアプリケーションの作業ディレクトリを指定します。

'Window Style' selection
「ウィンドウスタイル」の選択

Indicate how your launched application will behave when it is launched. Note: that the target application must support the attribute specified.
起動したアプリケーションの起動時の動作を示します。注:ターゲットアプリケーションは、指定された属性をサポートする必要があります。

Advanced
高度な

This section is for advanced use of, 'Run an Application'.
このセクションは、「アプリケーションの実行」の高度な使用を目的としています。

'Do not wait for launched application' option – This option allows the command to continue on without waiting for the launched app (this is the default option).
「起動したアプリケーションを待たない」オプション–このオプションにより、起動したアプリを待たずにコマンドを続行できます（これはデフォルトのオプションです）。

'Wait until launched application is started before continuing' option – This option tells VoiceAttack to wait until the launched application is started and is ready to receive input before continuing to the next action. You can specify how long to wait before giving up by selecting the, 'Only wait up to X seconds' option and filling in the box (see below). If the application does not launch, or, the time indicated is exceeded, you can
「起動前にアプリケーションが起動されるまで待機する」オプション–このオプションは、起動されたアプリケーションが起動し、次のアクションに進む前に入力を受け取る準備ができるまで待機するようVoiceAttackに指示します。「X秒までのみ待機する」オプションを選択し、ボックスに入力することで、あきらめるまでの待機時間を指定できます（以下を参照）。アプリケーションが起動しない場合、または指定された時間が経過した場合、次のことができます。

exit the command completely if you select the, 'Exit command if launch has failed or wait time exceeded' option. This option also has its own option called, 'Set command to target launched application'. This option will make the launched application the process target for the remaining duration of the command and all
「起動に失敗した場合、またはコマンドが待機時間を超えた場合にコマンドを終了する」オプションを選択した場合は、コマンドを完全に終了してください。このオプションには、「起動したアプリケーションをターゲットにするコマンドを設定」という独自のオプションもあります。このオプションは、コマンドの残りの期間、起動されたアプリケーションをプロセスターゲットにします。

subsequent input operations will be directed to it (this overrides the currently selected process target). See the guide titled, 'Application Focus (Process Target) Guide' later in this document.
後続の入力操作はそれに向けられます（これは現在選択されているプロセスターゲットをオーバーライドします）。このドキュメントで後述する「Application Focus（Process Target）Guide」というタイトルのガイドを参照してください。

'Wait until the launched application exits before continuing' option – This option will keep the command from continuing to the next action until the launched application closes. There are two options that go along with this feature:
「起動するアプリケーションが終了するまで待機する」オプション–このオプションは、起動したアプリケーションが閉じるまで、コマンドが次のアクションに継続しないようにします。この機能には2つのオプションがあります。

'Capture STDOUT to a text variable' and 'Capture Exit Code to an integer variable'. 'Capture STDOUT to a text variable' allows you to specify a text variable to hold the STDOUT generated by the launched app. The text variable can then be used as you wish (for instance, in a condition block or as a plugin parameter). This option is enabled by simply indicating the variable. Leave this blank to leave this option turned off.
「STDOUTをテキスト変数にキャプチャ」および「終了コードを整数変数にキャプチャ」。「STDOUTをテキスト変数にキャプチャする」を使用すると、起動されたアプリによって生成されたSTDOUTを保持するテキスト変数を指定できます。テキスト変数は、必要に応じて使用できます（たとえば、条件ブロック内またはプラグインパラメーターとして）。このオプションは、変数を指定するだけで有効になります。このオプションをオフのままにするには、空白のままにします。

'Capture Exit Code to an integer variable' works exactly like the STDOUT feature, except you are capturing the app's exit code an integer variable. Again, leave this field blank to leave this option turned off.
「終了コードを整数変数にキャプチャする」は、アプリの終了コードを整数変数にキャプチャすることを除いて、STDOUT機能とまったく同じように機能します。このオプションもオフのままにするには、このフィールドを空白のままにします。

You can specify how long to wait before giving up by selecting the, 'Only wait up to X seconds' option and filling in the box (see below). If the application fails to launch, or, the time indicated is exceeded, you can exit the command completely if you select the, 'Exit command if launch has failed or wait time exceeded' option.
「X秒までのみ待機する」オプションを選択し、ボックスに入力することで、あきらめるまでの待機時間を指定できます（以下を参照）。アプリケーションの起動に失敗した場合、または指定された時間が経過した場合、「起動に失敗した場合はコマンドを終了するか待機時間が経過した」オプションを選択すると、コマンドを完全に終了できます。

'Only wait up to X seconds for launched application' option – This option is available for both of the wait features listed above. Specify how many seconds to wait for the application to either launch or exit. If the time is exceeded, the command will continue on to the next step.
「起動されたアプリケーションを最大X秒まで待機する」オプション–このオプションは、上記の両方の待機機能で使用できます。アプリケーションが起動または終了するまで待機する秒数を指定します。時間を超過した場合、コマンドは次のステップに進みます。

'Exit command if launch has failed or wait time exceeded' option – This option will cause the command to exit if the application fails to launch or exceeds the time indicated in the, 'Only wait up to X seconds…' option.
「起動に失敗した場合、または待機時間を超えた場合にコマンドを終了する」オプション–このオプションは、アプリケーションが起動に失敗した場合、または「X秒まで待機する」オプションで示された時間を超えた場合にコマンドを終了します

'Stop a Process by Name'
「名前によるプロセスの停止」

Select this if you want to terminate a running process. The drop down list shows all running processes, as indicated by Windows. If you know the name of the process that you might like to terminate, you can select it here. Note that you can type into this box and that tokens can also be used. Use this functionality with great care.
実行中のプロセスを終了する場合、これを選択します。ドロップダウンリストには、Windowsが示すように、実行中のすべてのプロセスが表示されます。終了したいプロセスの名前がわかっている場合は、ここで選択できます。このボックスに入力でき、トークンも使用できることに注意してください。この機能は慎重に使用してください。

'Set a text value to the Windows clipboard'
「テキスト値をWindowsクリップボードに設定します」

This action will allow you to put a text value in the Windows clipboard. The value can also contain text tokens or condition tokens.
このアクションにより、Windowsクリップボードにテキスト値を配置できます。値には、テキストトークンまたは条件トークンを含めることもできます。

To clear the clipboard, leave the value in the box blank.
クリップボードをクリアするには、ボックスの値を空白のままにします。

The value that is stored in the clipboard can be accessed with the, '{CLIP}' token (see section on Tokens near the end of this document).
クリップボードに保存されている値には、「{CLIP}」トークンを使用してアクセスできます（このドキュメントの終わり近くにあるトークンに関するセクションを参照）。

Note: Pasting from the clipboard will require a command that executes a CTRL + V action (or whatever your system supports).
注：クリップボードから貼り付けるには、CTRL + Vアクション（またはシステムがサポートするもの）を実行するコマンドが必要です。

'Perform a Window Function'
「ウィンドウ関数の実行」

This action will let you target a particular window/process (main window of a process) and perform a particular action on it. To select a window of an application that is currently running, just drop down the box labeled, 'Window Title'. You can also choose the currently active, focused window by selecting, '[Active Window]' from the list. Note that this is a free input box and you can modify your selection (as detailed below). This box also accepts text variable names as well as any combination of tokens if needed.
このアクションにより、特定のウィンドウ/プロセス（プロセスのメインウィンドウ）をターゲットにして、特定のアクションを実行できます。現在実行中のアプリケーションのウィンドウを選択するには、「ウィンドウタイトル」というラベルの付いたボックスをドロップダウンします。リストから「[Active Window]」を選択して、現在アクティブなフォーカスされたウィンドウを選択することもできます。これは無料の入力ボックスであり、選択を変更できることに注意してください（詳細は以下を参照）。このボックスは、テキスト変数名と、必要に応じてトークンの任意の組み合わせも受け入れます。

The value in the drop-down box can contain wildcards indicated by asterisks (*). This is handy when the title of the window changes. To indicate that the window title contains the value in the box, put an asterisk on each end. For instance, if you want to target any window that contains, 'Notepad' in the title, put, '*Notepad*' (without quotes) in the box. To indicate that the window title starts with the value in the box, put an asterisk at the end: 'Notepad*'. To indicate that the window title ends with the value in the box, put an asterisk at the beginning: '*Notepad'. The values are not case- sensitive. The first window found to match the criteria indicated will be selected.
ドロップダウンボックスの値には、アスタリスク（*）で示されるワイルドカードを含めることができます。これは、ウィンドウのタイトルが変更されたときに便利です。ウィンドウのタイトルにボックスの値が含まれていることを示すには、両端にアスタリスクを付けます。たとえば、タイトルに「メモ帳」と入力し、ボックスに「*メモ帳*」（引用符なし）を含むウィンドウをターゲットにしたい場合。ウィンドウのタイトルがボックス内の値で始まることを示すには、最後にアスタリスク「Notepad *」を付けます。ウィンドウのタイトルがボックス内の値で終わることを示すには、先頭にアスタリスク「* Notepad」を付けます。値は大文字と小文字を区別しません。示された基準に一致することが検出された最初のウィンドウが選択されます。

Advanced: Note that you can also use process names as they appear in the Windows Task Manager. You can use wildcards the same as you do with the window titles.
詳細：Windowsタスクマネージャーに表示されるプロセス名を使用することもできます。ウィンドウのタイトルと同じようにワイルドカードを使用できます。

Window titles are checked first, and then the process names. More advanced: If the window is still not located, the window is then searched for by window class name and also by process id if the value in the box (which can be a text variable/token) resolves to an integer value. If needed, you can select, ‘[Active Window]’ from the dropdown list (instead of using tokens the old-fashioned way ;)). Note also that you can click on the, ‘Title’ or ‘Process’ links near the bottom to fill in this box (see, ‘Active Window Details’ section below).
最初にウィンドウのタイトルがチェックされ、次にプロセス名がチェックされます。より高度な：ウィンドウがまだ見つからない場合、ウィンドウはウィンドウクラス名で検索され、ボックス内の値（テキスト変数/トークンの場合もある）が整数値に解決される場合はプロセスIDでも検索されます。必要に応じて、ドロップダウンリストから「[Active Window]」を選択できます（従来の方法でトークンを使用する代わりに;））。また、下部の[タイトル]または[処理]リンクをクリックして、このボックスに入力できることに注意してください（以下の「アクティブウィンドウの詳細」セクションを参照）。

VoiceAttack will also allow the action to pause for a specified amount of time while trying to acquire the window and make sure it is receiving messages. If the pause expires, nothing will happen. If the window is found, processing will continue immediately. To set the maximum amount of pause time, just check the box labeled, 'Pause up to' and set the number of seconds (or fractions of seconds) to the desired value.
また、VoiceAttackを使用すると、ウィンドウを取得してメッセージを受信していることを確認しながら、指定した時間だけアクションを一時停止できます。一時停止が期限切れになると、何も起こりません。ウィンドウが見つかった場合、処理はすぐに続行されます。最大一時停止時間を設定するには、「一時停止まで」というラベルの付いたボックスをオンにして、秒数（または秒の小数部）を目的の値に設定します。

If the wait time is exceeded, or, if the window/process are not available, you can set the option, 'Exit command immediately if window/process not available' to jump out of the command instead of allowing the command to continue processing.
待機時間を超えた場合、またはウィンドウ/プロセスが使用できない場合は、オプションを設定して、「ウィンドウ/プロセスが使用できない場合はすぐにコマンドを終了する」コマンドを処理し続ける代わりに、コマンドから飛び出すことができます。

Once we have a handle on the window, there are several things that can be done to it:
ウィンドウのハンドルを取得したら、次のことができます。

'Display' - This will let you show/minimize/maximize/hide/etc. your targeted window. Just drop down the list and choose one of the following (Note: Yeah, this seems a little confusing. You will probably need to try out various methods to see which one works for you. The list could have been pared down to just a few simple items (show/minimize/maximize), but the underlying feature allows for more flexibility. In
「表示」- これにより、表示/最小化/最大化/非表示などを行うことができます。ターゲットウィンドウ。リストをドロップダウンして、次のいずれかを選択してください（注：うん、これは少し混乱しているようです。おそらく、どれが自分に合っているかを確認するためにさまざまな方法を試してみる必要があります。シンプルなアイテム（表示/最小化/最大化）ですが、基礎となる機能により柔軟性が高まります。

order to provide this flexibility, all the functions have been exposed to the user, both in name and description. So, basically, these will work differently based on situation, and, quite frankly, I'm not entirely sure if they are all even necessary for most (just left in for your use, if you can use 'em)) :
この柔軟性を提供するために、すべての機能が名前と説明の両方でユーザーに公開されています。したがって、基本的に、これらは状況に応じて異なる方法で動作し、率直に言って、ほとんどすべてに必要であるかどうかは完全にはわかりません（「em」を使用できる場合は使用のために残してください）：

Normal – Activates and displays a window. If the window is minimized or maximized, the system restores it to its original size and position.
通常–ウィンドウをアクティブにして表示します。ウィンドウが最小化または最大化されている場合、システムは元のサイズと位置に復元します。

Minimize – Minimizes the specified window and activates the next top–level window in the Z order.
最小化–指定されたウィンドウを最小化し、Zオーダーで次の最上位ウィンドウをアクティブにします。

Maximize – Maximizes the specified window.
最大化–指定されたウィンドウを最大化します。

Show Minimized – Activates the window and displays it as a minimized window.
Show Minimized– ウィンドウをアクティブにし、最小化されたウィンドウとして表示します。

Show Maximized – Activates the window and displays it as a maximized window.
最大化を表示–ウィンドウをアクティブにし、最大化されたウィンドウとして表示します。

Show Minimized No Activate – Displays the window as a minimized window. This value is similar to 'Show Minimized', except the window is not activated.
最小化されたアクティブ化なしを表示–ウィンドウを最小化されたウィンドウとして表示します。この値は、ウィンドウがアクティブ化されていないことを除いて、「Show Minimized」に似ています。

Force Minimized – Minimizes a window, even if the thread that owns the window is not responding.
Force Minimized– ウィンドウを所有するスレッドが応答しない場合でも、ウィンドウを最小化します。

Normal No Activate – Displays a window in its most recent size and position. This value is similar to 'Normal', except the window is not activated.
通常のアクティブ化なし–ウィンドウを最新のサイズと位置で表示します。この値は、ウィンドウがアクティブ化されていないことを除いて、「標準」に似ています。

Show – Activates the window and displays it in its current size and position.
表示–ウィンドウをアクティブにし、現在のサイズと位置で表示します。

Show No Activate – Displays the window in its current size and position. This value is similar to 'Show', except the window is not activated.
アクティブ化なしを表示–ウィンドウを現在のサイズと位置で表示します。この値は「表示」に似ていますが、ウィンドウがアクティブ化されていません。

Restore – Activates and displays the window. If the window is minimized or maximized, the system restores it to its original size and position. You should specify this flag when restoring a minimized window.
復元–ウィンドウをアクティブにして表示します。ウィンドウが最小化または最大化されている場合、システムは元のサイズと位置に復元します。最小化されたウィンドウを復元するときに、このフラグを指定する必要があります。

Show Default – Sets the show state based on the state of the window when it was created.
デフォルトの表示–作成時のウィンドウの状態に基づいて表示状態を設定します。

Hide – Hides the window and activates another window. Use with caution.
非表示–ウィンドウを非表示にして、別のウィンドウをアクティブにします。注意して使用してください。

The 'Display' feature also has an additional option, 'Set command to target this window'. This option will make the displayed window the process target for the rest of the command. This will override what you have set as the process target at the command, profile or global level. This is useful if you need to send input to a different application while you are in the middle of a command. There's a lot of extra detail in the section titled, 'Application Focus (Process Target) Guide' later in this document.
「表示」機能には、「このウィンドウをターゲットにするコマンドを設定」という追加オプションもあります。このオプションは、表示されたウィンドウを残りのコマンドのプロセスターゲットにします。これにより、コマンド、プロファイル、またはグローバルレベルでプロセスターゲットとして設定したものが上書きされます。これは、コマンドの途中で別のアプリケーションに入力を送信する必要がある場合に便利です。このドキュメントで後述する「アプリケーションフォーカス（プロセスターゲット）ガイド」というタイトルのセクションには、さらに多くの詳細があります。

'Close Window' – Closes the targeted window (surprise!).
「ウィンドウを閉じる」– ターゲットウィンドウを閉じます（驚き！）。

'Move Window' – This will allow you to move the targeted window to a specific X, Y coordinate on your screen. Yup... you can move it right off the screen, so be careful. Note that you can click on the, 'Location' link near the bottom to fill in these boxes (see, 'Active Window Details' section below).
「ウィンドウの移動」–これにより、ターゲットウィンドウを画面上の特定のX、Y座標に移動できます。うん...あなたはそれを画面の外に動かすことができるので、注意してください。下部の[場所]リンクをクリックしてこれらのボックスに入力できることに注意してください（以下の「アクティブウィンドウの詳細」セクションを参照）。

'Resize Window' – This will allow you to resize the targeted window to whatever width and height you like. Note that you can click on the, 'Size' link near the bottom to fill in these boxes (see, 'Active Window Details' section below).
'Resize Window'– これにより、ターゲットウィンドウの幅と高さを自由に変更できます。下部にある[サイズ]リンクをクリックして、これらのボックスを埋めることができます（以下の[アクティブウィンドウの詳細]セクションを参照）。

'Change Title' – This will allow you to change the title text of the target window. If you find yourself with multiple instances of the same application open, this can be handy when it comes to targeting each instance. Note that this box respects replacement tokens.
「タイトルの変更」–これにより、ターゲットウィンドウのタイトルテキストを変更できます。同じアプリケーションの複数のインスタンスを開いている場合は、各インスタンスを対象とするときに便利です。このボックスは置換トークンを尊重することに注意してください。

The, 'Active Window Details' helper section near the bottom will display information about the active window that you currently have selected, if you have selected a window outside of VoiceAttack itself. This section indicates the title of the active window as well as its process name, size and location. Clicking on the, 'Title' or, 'Process' links will copy the title text or process text value into the, 'Window Title' dropdown box up near the top. Clicking on the, 'Location' link will copy the location of the active window ((X, Y) coordinates) into the, 'Move Window' boxes. Clicking on the, 'Size' link will copy the size of the active window (width, height) into the, 'Resize Window' boxes.
VoiceAttackの外部のウィンドウを選択した場合、下部にある「アクティブウィンドウの詳細」ヘルパーセクションには、現在選択しているアクティブウィンドウに関する情報が表示されます。このセクションには、アクティブなウィンドウのタイトルとそのプロセス名、サイズ、場所が表示されます。[タイトル]または[処理]リンクをクリックすると、タイトルテキストまたはプロセステキスト値が、上部近くの[ウィンドウタイトル]ドロップダウンボックスにコピーされます。[場所]リンクをクリックすると、アクティブなウィンドウの場所（（X、Y）座標）が[ウィンドウの移動]ボックスにコピーされます。[サイズ]リンクをクリックすると、アクティブウィンドウのサイズ（幅、高さ）が[ウィンドウのサイズ変更]ボックスにコピーされます。

'Change Default Audio Devices'
「デフォルトのオーディオデバイスの変更」

This action will let you change Windows' default multimedia and communications playback and/or default recording devices.
このアクションにより、Windowsのデフォルトのマルチメディアおよび通信の再生および/またはデフォルトの録音デバイスを変更できます。

VoiceAttack uses Windows Media components and the Windows speech engine to do its thing. Windows Media components will only work with the default playback device, and the Windows speech engine can be set (and is set by default) to the default recording device. Sometimes you may want to change these devices when you are using VoiceAttack. For instance, you may want to switch over from a webcam mic / speaker setup over to a headset. This feature may save you a step (or several) while using VoiceAttack. To use this feature, simply select the devices you would like to use by checking the appropriate boxes and selecting the devices you would to use from the lists (the list only shows your currently-available devices). Now for lots of NOTES:
VoiceAttackは、Windows MediaコンポーネントとWindowsスピーチエンジンを使用してその処理を行います。Windows Mediaコンポーネントはデフォルトの再生デバイスでのみ機能し、Windowsの音声エンジンはデフォルトの録音デバイスに設定できます（デフォルトで設定されます）。VoiceAttackを使用しているときに、これらのデバイスを変更したい場合があります。たとえば、ウェブカメラのマイク/スピーカー設定からヘッドセットに切り替えることができます。この機能により、VoiceAttackの使用中に1つ（または複数）の作業を節約できます。この機能を使用するには、適切なボックスをチェックし、リストから使用するデバイスを選択して、使用するデバイスを選択します（リストには現在使用可能なデバイスのみが表示されます）。今、多くの注意事項について：

Note: If the Windows speech engine is set up to be something different than the default recording device (set up through the control panel), changing the default audio device will have no effect on the Windows speech engine, which means it will have no effect on VoiceAttack.
注：Windowsスピーチエンジンがデフォルトの録音デバイス（コントロールパネルから設定）とは異なるものに設定されている場合、デフォルトのオーディオデバイスを変更してもWindowsスピーチエンジンに影響はありません。つまり、効果はありません。 VoiceAttackで。

Note: This feature is only available for Windows 7 and later.
注：この機能は、Windows 7以降でのみ使用できます。

Note: If the device's underlying identifier is changed (driver update, Windows update,
注：デバイスの基盤となる識別子が変更された場合（ドライバーの更新、Windowsの更新、

crash, etc.) and this action is run, VoiceAttack will attempt to resolve the device by its last-known device name. If you see a log message containing, 'Resolved by device name', VoiceAttack has successfully made the change, but you may want to update this action as it is not automatically saved.
クラッシュなど）、このアクションが実行されると、VoiceAttackは最後に認識されたデバイス名でデバイスを解決しようとします。「Resolved by device name」というログメッセージが表示された場合、VoiceAttackは変更を正常に行いましたが、このアクションは自動的に保存されないため、更新することができます。

Note: There are other ways to change these devices. One way is through VoiceAttack startup in the Options screen. Another way is through command-line variables such as
注：これらのデバイスを変更する方法は他にもあります。1つの方法は、オプション画面でVoiceAttackを起動することです。別の方法は、次のようなコマンドライン変数を使用することです

-input, -output, etc (see the section on command-line variables later in this document). There are also corresponding tokens '{STATE_DEFAULTPLAYBACK}' and
-input、-outputなど（このドキュメントで後述するコマンドライン変数のセクションを参照）。また、対応するトークン「{STATE_DEFAULTPLAYBACK}」と

{STATE_DEFAULTRECORDING}' (see the token reference later in this document).
{STATE_DEFAULTRECORDING}'（このドキュメントで後述するトークンリファレンスを参照）。

<u>WARNING: This is not a VoiceAttack setting, rather a Windows device setting and can (and probably will) cause other applications that depend on the changed devices to appear to malfunction. It's not THAT big of a deal, but it will definitely throw you off when your Skype or TeamSpeak is not working how you left them.
警告：これはVoiceAttackの設定ではなく、Windowsデバイスの設定であり、変更されたデバイスに依存する他のアプリケーションが誤動作するように見える可能性があります。それほど大したことではありませんが、SkypeまたはTeamSpeakがあなたがそれらを残したように機能していない場合、間違いなくあなたを失望させます。</u>

'Windows Miscellaneous Functions'
「Windowsのその他の機能」

Given proper version and user rights, VoiceAttack will attempt to execute any of the following Windows functions (they're all mostly self-explanatory and/or probably don't need explanation):
適切なバージョンとユーザー権限が与えられた場合、VoiceAttackは次のWindows機能のいずれかを実行しようとします（これらはほとんど自明であり、おそらく説明を必要としません）。

'Toggle Desktop' - Toggles the desktop view in case the boss is coming. You'll need to come up with a creative command name here so he/she will think you are actually doing work.
'Toggle Desktop'- 上司が来ている場合にデスクトップビューを切り替えます。あなたが実際に仕事をしていると思うように、ここで創造的なコマンド名を考え出す必要があります。

'Lock Workstation' - Locks your PC so your roommate can't mess with your settings. Also, works to hide what you are doing from your boss ;)
「ワークステーションのロック」-PCをロックして、ルームメイトが設定を変更できないようにします。また、上司からあなたがしていることを隠す働きをします;)

'Log Off Current User' - Logs you out when you are done for the day.
「現在のユーザーをログオフ」-その日の作業が終了したらログアウトします。

'Sleep (Standby) PC' - Put your PC to sleep to save some power while you go and do stuff.
「スリープ（スタンバイ）PC」-PCをスリープ状態にして、作業中の電力を節約します。

'Force Sleep (Standby) PC' - This forces your PC into sleep mode. What this does is it tells Windows to not alert any apps that it's going into standby mode. Take that, apps!
「強制スリープ（スタンバイ）PC」-これにより、PCが強制的にスリープモードになります。これにより、Windowsがスタンバイモードになることをアプリに警告しないように指示されます。それを取ります、アプリ！

'Hibernate PC' - Hibernate your PC and save even more power.
'Hibernate PC' -PCを休止状態にして、さらに電力を節約します。

'Force Hibernate PC' - Again, this tells Windows to dis all of the apps by not telling them that it's going into hibernation.
'Force Hibernate PC'- 繰り返しになりますが、これは、休止状態になることを通知しないことで、すべてのアプリを無効にするようWindowsに指示します。

'Restart PC' – This will reboot your PC. This generally gives you a chance to save any unsaved documents
「PCの再起動」–これにより、PCが再起動します。通常、これにより未保存のドキュメントを保存することができます

'Force Restart PC' – This will not only reboot your PC, but it will NOT give you a chance to save any unsaved documents. Use with care.
「PCを強制的に再起動する」–これはPCを再起動するだけでなく、保存されていないドキュメントを保存する機会を与えません 。注意して使用してください。

'Shutdown PC' – Shuts down your PC and will probably give you the chance to save your unsaved documents.
「PCのシャットダウン」–PCをシャットダウンし、保存していないドキュメントを保存する機会を与えます。

'Force Shutdown PC' – Just like the restart option above, only the PC just stays off. You will LOSE any unsaved work. Use with care.
「強制シャットダウンPC」–上記の再起動オプションと同様に、PCのみがオフのままです。保存されていない作業は失われます。注意して使用してください。

'Power Off PC' – Works just like shutdown, except, where supported, will act like you pulled the plug out of the wall. This may have adverse effects on your system.
「PCの電源オフ」–シャットダウンのように動作しますが、サポートされている場合は、プラグを壁から引き抜いたように動作します。これは、システムに悪影響を与える可能性があります。

'Force Power Off PC' – Forces the power off of your pc. Just like all the other, 'force' options, you WILL LOSE unsaved data.
「Force Power Off PC」–PCの電源を強制的にオフにします。ただ、他のすべての、「力」オプションのように、あなたが失うことになる保存されていないデータを。

'Minimize All' – Minimizes all open windows.
「すべて最小化」–開いているすべてのウィンドウを最小化します。

'Undo Minimize All' – An undo for the, 'minimize all' you just did.
「すべて最小化を元に戻す」–直前に行った「すべて最小化」を元に戻します。

'Open Run Dialog' – Opens the run dialog so you can look like you know what you are doing… maybe even be productive.
「実行ダイアログを開く」–実行ダイアログを開くので、自分が何をしているのかを知っているように見えます。

'Open Search Dialog' – Opens the Windows search dialog so you can find those cat videos easily.
[検索ダイアログを開く]-Windowsの検索ダイアログを開くと、これらの猫のビデオを簡単に見つけることができます。

'Open Window Switcher' – This will arrange all open windows in a tiled format for you to choose from.
'Open Window Switcher'- これにより、開いているすべてのウィンドウがタイル形式で配置され、選択できるようになります。

'Run Screen saver' – Simply runs your screen saver if you have one that is
「スクリーンセーバーの実行」-スクリーンセーバーがあれば実行します

active. "Why did you just shove this in here?" It was just one more option, might as well include it, right? Easier than adding it later. Coming soon: Espresso Maker
アクティブ。「なぜこれをここに押し込んだのですか？」これはもう1つのオプションでした。後で追加するよりも簡単です。近日提供予定：エスプレッソメーカー

'Hide Task Bar', 'Show Task Bar' – Hides and shows the Windows task bar.
「タスクバーを非表示」、「タスクバーを表示」-Windowsタスクバーを非表示および表示します。

'Set Audio Level'
「オーディオレベルの設定」

This screen will allow you to add an action that will change a specified audio endpoint's volume. This is useful if you want to voice control or hotkey various volume controls.
この画面では、指定したオーディオエンドポイントの音量を変更するアクションを追加できます。これは、音声コントロールをしたり、さまざまな音量コントロールをホットキーしたい場合に便利です。

You can choose to set the overall system volume, the volume of specific recording or playback devices (microphones, speakers, headphones) or the volume of various applications as they relate to the System Volume Mixer and the overall system volume (you can find the user interface of the System Volume Mixer in the system tray, or by right-clicking on the VoiceAttack icon in the task bar and selecting, 'System Volume Mixer').
システム全体の音量、特定の録音または再生デバイス（マイク、スピーカー、ヘッドフォン）の音量、またはシステムボリュームミキサーとシステム全体の音量に関連するさまざまなアプリケーションの音量（ユーザーを見つけることができます）を設定できます。システムトレイのシステムボリュームミキサーのインターフェイス、またはタスクバーのVoiceAttackアイコンを右クリックして[システムボリュームミキサー]を選択します）。

To change the overall audio volume of your computer (most common), select, ‘System’. To change the volume level of a specific playback device (like speakers or headphones), select the, ‘Playback Device’ option and select the device you would like to modify. Choosing the, ‘Default’ device will always select the default playback device as indicated by Windows. Note that selecting the default playback device is the same as choosing, ‘System’ (this is left in for backward-compatibility).
コンピューターの全体的な音量を変更するには（最も一般的）、[システム]を選択します。特定の再生デバイス（スピーカーやヘッドフォンなど）の音量レベルを変更するには、[再生デバイス]オプションを選択し、変更するデバイスを選択します。「デフォルト」デバイスを選択すると、Windowsの指示に従ってデフォルトの再生デバイスが常に選択されます。デフォルトの再生デバイスを選択することは、「システム」を選択することと同じであることに注意してください（これは下位互換性のために残されています）。

To change the volume level of a specific recording device (microphone), select the, ‘Recording Device’ option and select the device you would like to modify. Choosing the, ‘Default’ device will always select the default playback device as indicated by Windows. If you want to change the volume of whatever device your speech engine is currently using, select, ‘Speech Engine Recording Device’ from the list.
特定の録音デバイス（マイク）の音量レベルを変更するには、[録音デバイス]オプションを選択し、変更するデバイスを選択します。「デフォルト」デバイスを選択すると、Windowsの指示に従ってデフォルトの再生デバイスが常に選択されます。音声エンジンが現在使用しているデバイスの音量を変更する場合は、リストから「音声エンジン録音デバイス」を選択します。

To change the audio level of an application, select the application from the dropdown list. Note that this is a free input box and you can modify your selection (as detailed below). This box also respects tokens if needed.
アプリケーションの音声レベルを変更するには、ドロップダウンリストからアプリケーションを選択します。これは無料の入力ボックスであり、選択を変更できることに注意してください（詳細は以下を参照）。このボックスは、必要に応じてトークンも尊重します。

The value in the drop-down box can contain wildcards indicated by asterisks (*). This is handy when the title of the window changes. To indicate that the window title contains the value in the box, put an asterisk on each end. For instance, if you want to target any window that contains, 'VLC' in the title, put, '*VLC*' (without quotes) in the box. To indicate that the window title starts with the value in the box, put an asterisk at the end: 'VLC*'. To indicate that the window title ends with the value in the box, put an asterisk at the beginning: '*VLC'. The values are not case-sensitive. The first window found to match the criteria indicated will be selected. Advanced: Note that you can also use process names as they appear in the Windows Task Manager.
ドロップダウンボックスの値には、アスタリスク（*）で示されるワイルドカードを含めることができます。これは、ウィンドウのタイトルが変更されたときに便利です。ウィンドウのタイトルにボックスの値が含まれていることを示すには、両端にアスタリスクを付けます。たとえば、タイトルに「VLC」と入力し、ボックスに「* VLC *」（引用符なし）を含むウィンドウをターゲットにしたい場合。ウィンドウのタイトルがボックス内の値で始まることを示すには、末尾にアスタリスク「VLC *」を付けます。ウィンドウのタイトルがボックス内の値で終わることを示すには、先頭にアスタリスク「* VLC」を付けます。値は大文字と小文字が区別されません。示された基準に一致することが検出された最初のウィンドウが選択されます。高度な：Windowsタスクマネージャーに表示されるプロセス名を使用することもできます。

You can use wildcards the same as you do with the window titles. Window titles are checked first, and then the process names. More advanced: If a window cannot be found by title or process name, the window class names will then be checked.
ウィンドウのタイトルと同じようにワイルドカードを使用できます。最初にウィンドウのタイトルがチェックされ、次にプロセス名がチェックされます。より高度：タイトルまたはプロセス名でウィンドウが見つからない場合、ウィンドウクラス名がチェックされます。

Wildcards apply if you need them. Note that this will be the class name of the window itself and not the class name of a child control. Again, this is an advanced feature that you may never ever use.
ワイルドカードは必要な場合に適用されます。これはウィンドウ自体のクラス名であり、子コントロールのクラス名ではないことに注意してください。繰り返しになりますが、これは使用することのない高度な機能です。

Once you select the type of audio that you want to control, you can then set the audio's level using several methods. You can mute and unmute the audio by selecting
制御するオーディオの種類を選択したら、いくつかの方法を使用してオーディオのレベルを設定できます。を選択すると、音声をミュートおよびミュート解除できます

either, 'Mute' or 'Unmute', or you and toggle the mute on and by selecting, 'Toggle Mute'. You can also set the level of the audio to a specific value by selecting, 'Level' and inputting a value. This value must resolve to a whole number from 0 to 100. So, if you want to set your volume to 50%, simply enter, '50' (without quotes) into the box.
「ミュート」または「ミュート解除」のいずれか、または「ミュートの切り替え」を選択して、ミュートを切り替えます。「レベル」を選択して値を入力することにより、オーディオのレベルを特定の値に設定することもできます。この値は、0～100の整数に解決する必要があります。したがって、ボリュームを50%に設定する場合は、ボックスに「50」（引用符なし）と入力します。

Advanced: Note that this box will also accept an integer variable name or a token that resolves into either a number or even an integer variable name.
詳細：このボックスは、整数変数名または数値または整数変数名に解決されるトークンも受け入れることに注意してください。

To increase/decrease the current volume by a certain amount, select the, 'Offset' option and input the value by which you would like to increase or decrease the volume. For example, if you want to increase the volume by 10, input 10 in the input box. If you want to decrease the volume by 10, enter -10 in the input box. Advanced: Note that this box will also accept an integer variable name or a token that resolves into either a number or even an integer variable name.
現在の音量を特定の量だけ増減するには、「オフセット」オプションを選択し、音量を増減する値を入力します。たとえば、ボリュームを10増やしたい場合は、入力ボックスに10と入力します。ボリュームを10減らしたい場合は、入力ボックスに-10を入力します。詳細：このボックスは、整数変数名または数値または整数変数名に解決されるトークンも受け入れることに注意してください。

Note: The application volume is not the actual volume that is controlled by the selected application. For instance, there is a volume slider bar on Windows Media Player that controls the volume. VoiceAttack does NOT control that slider. VoiceAttack will control the volume for WMP as it is reflected in the System Volume Mixer. That means that if you set WMP's audio level value to 50, the slider in the System Volume Mixer will adjust the volume for WMP to 50% of whatever the system volume is currently set.
注：アプリケーションのボリュームは、選択したアプリケーションによって制御される実際のボリュームではありません。たとえば、Windows Media Playerには音量を制御する音量スライダーバーがあります。VoiceAttackはそのスライダーを制御しません。VoiceAttackは、システムボリュームミキサーに反映されるWMPの音量を制御します。つまり、WMPのオーディオレベル値を50に設定すると、システムボリュームミキサーのスライダーは、WMPのボリュームを現在設定されているシステムボリュームの50%に調整します。

'Block/Unblock Keyboard Input'
「キーボード入力のブロック/ブロック解除」

This action works like a gaming aid and will allow you to have VoiceAttack attempt to block, unblock or toggle the blocking of keyboard key presses that you choose (you know… like when that annoying Windows key gets pressed all the time). ‘Attempt’ is indicated, as this will depend on your system. Factors such as other software running and the order by which you execute that software in regards to VoiceAttack can alter your experience. Also, if your game or app relies on input that cannot be blocked, VoiceAttack will not be able to block it. So, in a nutshell, ‘Your mileage may vary’.
このアクションはゲーミングエイドのように機能し、VoiceAttackに、選択したキーボードのキー押下のブロック、ブロック解除、または切り替えを試行させることができます（ご存知のように…迷惑なWindowsキーが常に押されるときのように）。これはシステムによって異なるため、「試行」が示されます。実行中の他のソフトウェアや、VoiceAttackに関してそのソフトウェアを実行する順序などの要因によって、エクスペリエンスが変わる場合があります。また、ゲームまたはアプリがブロックできない入力に依存している場合、VoiceAttackはそれをブロックできません。したがって、一言で言えば、「あなたの走行距離は異なる場合があります」。

Ok, back to it… There are three actions that you can perform:
さて、戻って…実行できる3つのアクションがあります。

Block – this blocks input from occurring from the selected keys.
ブロック–選択したキーからの入力をブロックします。

Unblock – this will remove blocks that were previously put in place by a VoiceAttack, ‘Block’ action.
ブロック解除–これは、VoiceAttackの「ブロック」アクションによって以前に配置されたブロックを削除します。

Toggle – this will either block unblocked input or unblock blocked input (it’s a toggle, lol).
Toggle– これは、ブロックされていない入力をブロックするか、ブロックされた入力をブロック解除します（トグルです、笑）。

The scope of blocking depends on the next option. If you want to just specify a certain number of keys, you can select the, ‘Only the keys listed below’ option. Then, simply press the keyboard keys that you want to block. The keys that are selected will appear in the box below the selection. Note that you can remove the keys by clicking on them or by clicking, ‘Clear’. When this action is executed, only the selected keys will be blocked. If you want to block ALL input except for certain keys, select the, ‘All but the keys listed below’ option. Note that indicating no keys with this option will block all keys.
ブロックの範囲は次のオプションに依存します。特定の数のキーのみを指定する場合は、「以下にリストされているキーのみ」オプションを選択できます。次に、ブロックするキーボードキーを押します。選択されたキーは、選択の下のボックスに表示されます。キーをクリックするか、「クリア」をクリックして、キーを削除できることに注意してください。このアクションが実行されると、選択されたキーのみがブロックされます。特定のキー以外のすべての入力をブロックする場合は、「以下にリストされているキーを除くすべて」オプションを選択します。このオプションでキーを指定しないと、すべてのキーがブロックされることに注意してください。

Each, ‘Block Keyboard Input’ action is cumulative. What that means is that if you block, say, key ‘X’ in one action and block key, ‘Y’ in a second action, both, ‘X’ and ‘Y’ keys will be blocked at that point.
各「キーボード入力をブロック」アクションは累積的です。つまり、あるアクションで「X」キーをブロックし、2番目のアクションで「Y」キーをブロックすると、その時点で「X」キーと「Y」キーがブロックされます。

Note: Although key presses will continue to be executed by VoiceAttack actions, VoiceAttack will not respond to key presses from the keyboard. Use with care.
注：キーの押下はVoiceAttackアクションによって引き続き実行されますが、VoiceAttackはキーボードからのキーの押下に応答しません。注意して使用してください。

'Block/Unblock Mouse Input'
「マウス入力のブロック/ブロック解除」

This action works like a gaming aid and will allow you to have VoiceAttack attempt to block, unblock or toggle the blocking of various mouse actions that you choose. Again, just like in, 'Block/Unblock Keyboard Input', 'attempt' is indicated, as this will depend on your system (see previous section for rant ;)).
このアクションはゲーミングエイドのように機能し、VoiceAttackに、選択したさまざまなマウスアクションのブロック、ブロック解除、またはブロックの切り替えを試行させることができます。繰り返しますが、「キーボード入力のブロック/ブロック解除」と同様に、「試行」が示されます。これはシステムに依存するためです（暴言については前のセクションを参照してください;））。

There are three actions that you can perform:
実行できるアクションは3つあります。

Block – this blocks input from occurring from the selected mouse actions.
ブロック–選択したマウスアクションからの入力をブロックします。

Unblock – this will remove blocks that were previously put in place by a VoiceAttack, 'Block' action.
ブロック解除–これは、VoiceAttackの「ブロック」アクションによって以前に配置されたブロックを削除します。

Toggle – this will either block unblocked input or unblock blocked input.
トグル–ブロックされていない入力をブロックするか、ブロックされた入力をブロック解除します。

The scope of blocking depends on the next option. If you want to just specify certain actions, you can select the, 'Only the actions indicated below' option. Then, just click on the items you want to block. You'll see all five standard mouse buttons (left, right, middle, back, forward) as well as four different scrolling actions (scroll forward, back, (tilt) left, (tilt) right). Note that you can remove the actions by clicking on them again or by clicking, 'Clear'. When this action is executed, only the selected actions will be blocked. If you want to block ALL mouse actions except for certain ones, select the, 'All but the actions indicated below' option. Note that indicating no actions will block all selectable mouse actions.
ブロックの範囲は次のオプションに依存します。特定のアクションのみを指定する場合は、[以下に示すアクションのみ]オプションを選択できます。次に、ブロックするアイテムをクリックします。5つの標準マウスボタン（左、右、中央、後ろ、前）のすべてと、4つの異なるスクロールアクション（前、後ろ、（傾き）左、（傾き）右）が表示されます。アクションを削除するには、もう一度クリックするか、「クリア」をクリックします。このアクションが実行されると、選択されたアクションのみがブロックされます。特定のアクション以外のすべてのマウスアクションをブロックする場合は、[以下に示すアクションを除くすべて]オプションを選択します。アクションを指定しないと、選択可能なすべてのマウスアクションがブロックされることに注意してください。

Each, ‘Block Mouse Input’ action is cumulative. What that means is that if you block, say, the middle mouse button in one action and block the right mouse button in a second action, both the middle and right buttons will be blocked at that point.
各「マウス入力をブロック」アクションは累積的です。つまり、たとえば、あるアクションでマウスの中央ボタンをブロックし、2番目のアクションでマウスの右ボタンをブロックすると、その時点で中央ボタンと右ボタンの両方がブロックされます。

Note: Although mouse actions will continue to be executed by VoiceAttack, VoiceAttack itself will not respond to your physical mouse actions. Use with care.
注：VoiceAttackは引き続きマウスアクションを実行しますが、VoiceAttack自体は物理的なマウスアクションに応答しません。注意して使用してください。

Note: Extended buttons, like the ones found on macro-enabled mice, can ONLY be blocked via the software that comes with those mice.
注：マクロ対応マウスにあるような拡張ボタンは、それらのマウスに付属のソフトウェアでのみブロックできます。

'Restrict Mouse Movement'
「マウスの動きを制限する」

This action works like a gaming aid and will allow you to have VoiceAttack attempt to reduce the speed of the mouse. Again, just like in, ‘Block/Unblock Keyboard/Mouse Input’, ‘attempt’ is indicated, as this will depend on your system. Simply choose the level of movement restriction by using the slider. Choosing 0% clears any previously- invoked restriction, 10% would indicate only a small amount of movement restriction, while 100% would indicate a full block (the cursor will not move, so use with care).
このアクションはゲーミングエイドのように機能し、VoiceAttackにマウスの速度を低下させようとします。繰り返しますが、「キーボード/マウス入力のブロック/ブロック解除」のように、「試行」が示されます。これはシステムによって異なります。スライダーを使用して、移動制限のレベルを選択するだけです。0%を選択すると、以前に呼び出された制限がクリアされ、10%はわずかな移動制限のみを示し、100%は完全なブロックを示します（カーソルは移動しないため、注意して使用してください）。

Dictation (Speech to Text)
ディクテーション（音声読み上げ）

In order to be as flexible as possible, 'dictation' in VoiceAttack requires multiple parts.
可能な限り柔軟にするために、VoiceAttackの「ディクテーション」には複数の部分が必要です。

Since VoiceAttack is already, 'listening' for your commands, you will need to be able to turn dictation on and off. There are two new actions to do this for you: Start Dictation Mode, and Stop Dictation Mode.
VoiceAttackは既にコマンドを「聞いている」ため、ディクテーションをオンまたはオフにできる必要があります。これを行うための2つの新しいアクションがあります。ディクテーションモードの開始とディクテーションモードの停止です。

To turn on dictation mode, just add the action to one of your commands. You can initialize variables, play a sound, clear the dictation buffer (more on that later) or whatever you want before or after the Start Dictation action. For example, you can have a command called, 'Open Quote'. In that command, you can play a sound to notify you that dictation is on and, 'listening', followed by the, 'Start Dictation' action.
ディクテーションモードをオンにするには、コマンドの1つにアクションを追加するだけです。変数の初期化、サウンドの再生、ディクテーションバッファのクリア（詳細は後ほど）、またはディクテーションの開始アクションの前後に必要なものをクリアできます。たとえば、「Open Quote」というコマンドを使用できます。そのコマンドでは、音声を再生して、ディクテーションがオンになっていることを通知し、「聞き取り」、続いて「ディクテーションの開始」アクションを実行できます。

To turn off dictation mode, just add the Stop Dictation Mode action to a command you specify. Again, you can paste the dictation buffer, play a sound, clear variables or whatever in the same command.
ディクテーションモードをオフにするには、指定したコマンドにディクテーションモードの停止アクションを追加します。繰り返しますが、ディクテーションバッファを貼り付けたり、サウンドを再生したり、変数などを同じコマンドでクリアしたりできます。

Note: When dictation mode is on, you can still issue normal VoiceAttack commands. The commands will be processed and not included in the dictation buffer. That way, you can still, 'fire all weapons' when you're in the middle of messaging your wingmen ;)
注：ディクテーションモードがオンの場合でも、通常のVoiceAttackコマンドを発行できます。コマンドは処理され、ディクテーションバッファには含まれません。そのようにして、あなたはまだ、あなたがあなたの翼兵にメッセージを送る最中にいるとき、「すべての武器を発射する」ことができます;)

The main part of the dictation feature is what is referred to as the speech buffer (or, 'the buffer', for short).
ディクテーション機能の主要部分は、音声バッファ（または略して「バッファ」）と呼ばれるものです。

The speech buffer holds all the words that you are speaking. Every time you speak and then pause, the speech is converted to text and is added to the buffer. The buffer will continue to grow until you clear it. To clear the speech buffer, the, 'Clear Dictation Buffer' action is provided. Simply add the action to your specified command at the place where you want to clear the buffer. As an option in this action, you can limit what is cleared to the last statement in the buffer. Also, in the Start and Stop Dictation actions, you can optionally clear what is in the buffer.
スピーチバッファには、話しているすべての単語が保持されます。発言してから一時停止するたびに、スピーチはテキストに変換され、バッファに追加されます。バッファは、クリアするまで拡大し続けます。音声バッファをクリアするには、「ディクテーションバッファのクリア」アクションが提供されます。バッファをクリアしたい場所で、指定したコマンドにアクションを追加するだけです。このアクションのオプションとして、クリアするものをバッファー内の最後のステートメントに制限できます。また、ディクテーションの開始アクションと停止アクションでは、オプションでバッファーの内容をクリアできます。

The value in the dictation buffer is accessed in VoiceAttack in places that accept tokens (see, 'Tokens' in the VoiceAttack documentation for more info on how to use them... seriously... check it out because you can do a lot with tokens).
ディクテーションバッファーの値は、トークンを受け入れる場所でVoiceAttackでアクセスされます（トークンの使用方法の詳細については、VoiceAttackドキュメントの「Tokens」を参照してください...真剣に...トークン）。

The tokens for the dictation buffer is {DICTATION} and {DICTATION:options}. For example, you can pop that token into the Text to Speech output box, into the Quick Input value box or into the Set Windows Clipboard value box.
ディクテーションバッファのトークンは{DICTATION}および{DICTATION：options }です。たとえば、そのトークンを[音声入力]テキストボックス、[クイック入力]値ボックス、または[Windowsクリップボード値の設定]ボックスにポップできます。

VoiceAttack will convert the {DICTATION} token into what is contained in the buffer at the point that it is used.
VoiceAttackは、{DICTATION}トークンを、使用された時点でバッファーに含まれているトークンに変換します。

Some things you might want to do with what is in the speech buffer:
スピーチバッファの内容を使用して、次のことを実行できます。

-output to some application (messaging teammates)
-いくつかのアプリケーションへの出力（メッセージングチームメイト）

-output to text to speech (feedback for you)
-テキストを音声に出力（フィードバック）

-output to a plugin, the VoiceAttack log, the Windows clipboard, etc.
-プラグイン、VoiceAttackログ、Windowsクリップボードなどへの出力

There are two ways to get what is in the speech buffer out of VoiceAttack and into another application. One way is to output the text one character at a time using VoiceAttack's Quick
VoiceAttackから音声バッファにあるものを別のアプリケーションに取り込むには、2つの方法があります。1つの方法は、VoiceAttackのQuickを使用して、一度に1文字ずつテキストを出力することです

Input feature. This is a fairly reliable way to get your text out, however, it is one character at a time. That means that there will be a delay getting the text out. Since there is a delay, there are opportunities to accidentally hit other keys while the text is being output to your application.
入力機能。これはテキストを出力するためのかなり信頼できる方法ですが、一度に1文字です。つまり、テキストの出力が遅れることになります。遅延があるため、テキストがアプリケーションに出力されている間に誤って他のキーを押す機会があります。

The preferred way to get the buffered text out is by using the Windows clipboard. This way, all the text is output in one shot, reducing the chance for error. To add what is in the buffer to the Windows clipboard, simply add a, 'Set a Text Value to the Windows Clipboard' action to your command. In the, 'Value' box, just add the, '{DICTATION}' or '{DICTATION:options}' token (again, see, 'Tokens' in the VoiceAttack documentation for more info on how to use them). To paste from the Windows clipboard, simply issue your preferred method of pasting (CTRL + V or SHIFT + INSERT with appropriate delays).
バッファリングされたテキストを取得する好ましい方法は、Windowsクリップボードを使用することです。これにより、すべてのテキストが1ショットで出力されるため、エラーが発生する可能性が低くなります。バッファーの内容をWindowsクリップボードに追加するには、コマンドに「テキスト値をWindowsクリップボードに設定」アクションを追加するだけです。[値]ボックスに、「{DICTATION}」または「{DICTATION：options }」トークンを追加します（これらの使用方法の詳細については、VoiceAttackドキュメントの「Tokens」を参照してください）。Windowsのクリップボードから貼り付けるには、好みの貼り付け方法（CTRL + VまたはSHIFT + INSERTと適切な遅延）を発行するだけです。

Note: If you do not have access to the Windows clipboard (within games), outputting your text one character at a time may be your only option.
注：Windowsクリップボード（ゲーム内）にアクセスできない場合は、一度に1文字ずつテキストを出力することが唯一のオプションです。

It also needs to be noted that dictation is only as reliable as your speech engine's training and/or your hardware and/or your configuration and drivers. It's not going to be nearly as accurate as your spoken commands. For some, dictation may be near flawless while others may find it hit or miss or even frustrating. Apologies in advance, however, this is a first go- round with dictation and is provided as-is (sorry!). Over time, I'm hoping that this will get even better.
また、ディクテーションは、スピーチエンジンのトレーニング、ハードウェア、構成、ドライバと同程度にしか信頼できないことに注意する必要があります。音声コマンドほど正確ではありません。一部の人にとっては、口述は完璧に近いかもしれませんが、他の人はそれをヒットまたはミス、あるいはイライラさえするかもしれません。ただし、事前におApび申し上げますが、これは口述による最初の試みであり、現状のまま提供されます（ごめんなさい！）。時間が経つにつれて、私はこれがさらに良くなることを望んでいます。

'Start Dictation Mode'
「ディクテーションモードを開始する」

This action will turn on VoiceAttack's, 'dictation' mode. Any recognized speech while this mode is active will be added to the dictation buffer. Recognized commands that are spoken on their own will still be processed first and not added to the dictation buffer (just in case you are dictating a message to your team and then you get attacked...
このアクションにより、VoiceAttackの「ディクテーション」モードがオンになります。このモードがアクティブな間に認識された音声は、ディクテーションバッファに追加されます。自分で話された認識されたコマンドは、最初に処理され、ディクテーションバッファに追加されません（チームへのメッセージをディクテーションしてから攻撃を受ける場合に備えて...

lol). Note that dictation can be stopped and started and the dictation buffer will be maintained until you clear it.
笑）。ディクテーションは停止および開始でき、ディクテーションバッファはクリアするまで維持されます。

The value in the dictation buffer can be accessed by using the {DICTATION} and
ディクテーションバッファの値には、{DICTATION}と

{DICTATION:options} tokens. See more about this token in the Token section near the end of this document.
{DICTATION：options }トークン。このドキュメントの終わり近くにあるトークンセクションで、このトークンの詳細を参照してください。

The, 'Clear dictation buffer before starting dictation mode' does just that. It clears the entire buffer once you start this action (for convenience... saves a step if you really need it).
「ディクテーションモードを開始する前にディクテーションバッファをクリアする」がまさにそれを行います。このアクションを開始すると、バッファ全体がクリアされます（便宜上...本当に必要な場合はステップを保存します）。

'Stop Dictation Mode'
「ディクテーションモードの停止」

This action will turn off VoiceAttack's, 'dictation' mode. Note that the dictation buffer is still maintained while you start and stop dictation mode. That is, unless you select the, 'Clear dictation buffer after stopping dictation mode' option. Using this option will clear the dictation buffer immediately after stopping dictation mode. Again, more of a convenience feature to save steps.
このアクションは、VoiceAttackの「ディクテーション」モードをオフにします。ディクテーションモードを開始および停止している間も、ディクテーションバッファは維持されます。つまり、「ディクテーションモードの停止後にディクテーションバッファをクリアする」オプションを選択しない限りです。このオプションを使用すると、ディクテーションモードを停止した直後にディクテーションバッファがクリアされます。繰り返しますが、ステップを節約するための便利な機能が増えています。

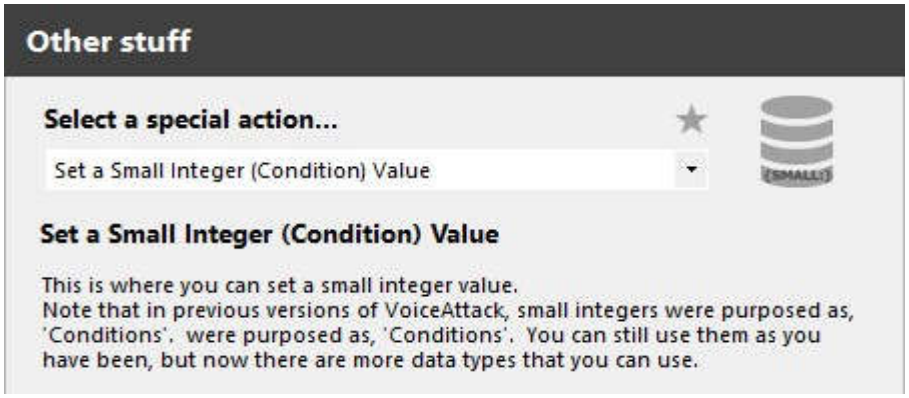
'Clear Dictation Buffer'
「ディクテーションバッファのクリア」

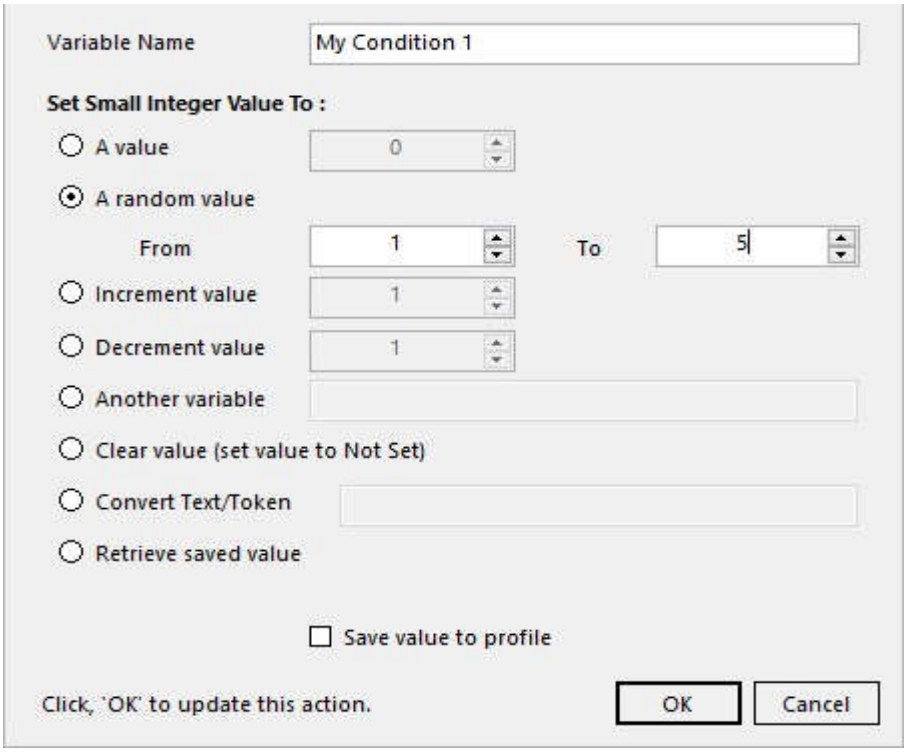
This action will clear the dictation buffer. The option, 'clear only the last statement' is handy if you botch the last thing you said (which will happen a lot, given the current state of the speech engine).
このアクションにより、ディクテーションバッファがクリアされます。「最後のステートメントのみをクリアする」オプションは、最後に言ったことをやめる場合に便利です（これは、音声エンジンの現在の状態を考えると、頻繁に発生します）。

Advanced
高度な

The next few command actions are considered, 'advanced' and are for use within VoiceAttack to allow users to make things a bit more flexible. There is a good chance that you will never use these features or use them very little (since they are a little bit outside of what you probably need to make VoiceAttack work for you). Have fun!
次のいくつかのコマンドアクションは「高度」とみなされ、VoiceAttack内で使用して、ユーザーが物事をもう少し柔軟にできるようにします。これらの機能を使用したり、ごくわずかしか使用したりしない可能性が十分にあります（VoiceAttackを機能させるためにおそらく必要なものから少し外れているため）。楽しむ！

'Set a Small Integer (Condition) Value'
「小さな整数（条件）値を設定する」





NOTE: Due to the expansion of VoiceAttack's available variable types, what used to be called a, 'Condition' has been renamed, 'Small Integer'. The Small Integer type is going to be left in for backward-compatibility, however, you may find the Integer type a little bit more useful. Make sure to check out the Integer, Boolean and Decimal variable types (and the accompanying new features) outlined below.
注：VoiceAttackの利用可能な変数タイプの拡張により、「Condition」と呼ばれていたものは「Small Integer」と名前が変更されました。Small Integer型は後方互換性のために残されますが、Integer型の方が少し便利かもしれません。以下に概説する整数、ブール、および10進数の変数タイプ（および付随する新機能）を必ずチェックアウトしてください。

This command action will allow you to set the value of a small integer variable. These
このコマンドアクションを使用すると、短整数変数の値を設定できます。これら

variables are used in conjunction with Conditional blocks ('If' statements) to control the flow of your command actions (more on conditional blocks below).
変数は、条件付きブロック（「If」ステートメント）と組み合わせて使用して、コマンドアクションのフローを制御します（以下の条件付きブロックの詳細）。

The Small Integer name can be whatever name you want. The variable names are not case-sensitive and they must not contain semicolons or colons (variable names can only contain colons if contained within a token… this would be a good place to indicate that this input box also processes tokens).
Small Integer名には、任意の名前を付けることができます。変数名は大文字と小文字を区別せず、セミコロンまたはコロンを含めることはできません（変数名には、トークンに含まれる場合にのみコロンを含めることができます。これは、この入力ボックスがトークンも処理することを示す適切な場所です）。

The value that you set to your small integer variable can be explicit (you set an exact value), random (from a low value to a high value), incremental, decremental, to another small integer variable's value that you have defined, or to the value of text/tokens. The value can be no more than 32,767 and no less than -32,768. If you try to set your small integer variable to a value outside of this range, your small integer variable's value will be set to, 'Not Set'.
スモール整数変数に設定する値は、明示的（正確な値を設定）、ランダム（低値から高値）、増分、減分、定義した別のスモール整数変数の値、またはテキスト/トークンの値。値は32,767以下、-32,768以上にすることができます。小整数変数をこの範囲外の値に設定しようとすると、小整数変数の値は「未設定」に設定されます。

If you want your small integer variable to be saved with the active profile, select the, 'Save value to profile' option. This will allow you to access the value between application sessions (VoiceAttack application is closed and then run again).
小さな整数変数をアクティブなプロファイルと共に保存する場合は、[プロファイルに値を保存する]オプションを選択します。これにより、アプリケーションセッション間の値にアクセスできるようになります（VoiceAttackアプリケーションを閉じてから再実行します）。

To retrieve the small integer variable saved with the profile, select the, 'Retrieve saved value' option.
プロファイルで保存された短整数変数を取得するには、「保存された値を取得」オプションを選択します。

Note: To clear all previously-saved small integer variables, see the, 'Clear Saved Values from Profile' action.
注：以前に保存された小さな整数変数をすべてクリアするには、「プロファイルから保存された値をクリア」アクションを参照してください。

Note: You can define as many values as you want, however, the values that you define are not persisted. That is, they are not saved to disk. These values will be reset every time you restart VoiceAttack. If you want your values saved to disk for use between application sessions, select the 'Save value to profile' option.
注：必要な数の値を定義できますが、定義した値は保持されません。つまり、ディスクには保存されません。これらの値は、VoiceAttackを再起動するたびにリセットされます。アプリケーションセッション間で使用するために値をディスクに保存する場合は、[プロファイルに値を保存する]オプションを選択します。

Advanced: Variables can be scoped at the command-level, at the profile-level and globally. Most will use the globally-scoped variables (for ease of use), however, for those that need a finer level of control, make sure to check out the, 'Advanced Variable Control (Scope)' section later in this document.
高度：変数は、コマンドレベル、プロファイルレベル、およびグローバルでスコープできます。ほとんどはグローバルスコープの変数を使用します（使いやすさのため）が、より細かいレベルの制御が必要な場合は、このドキュメントで後述する「高度な変数制御（スコープ）」セクションを確認してください。

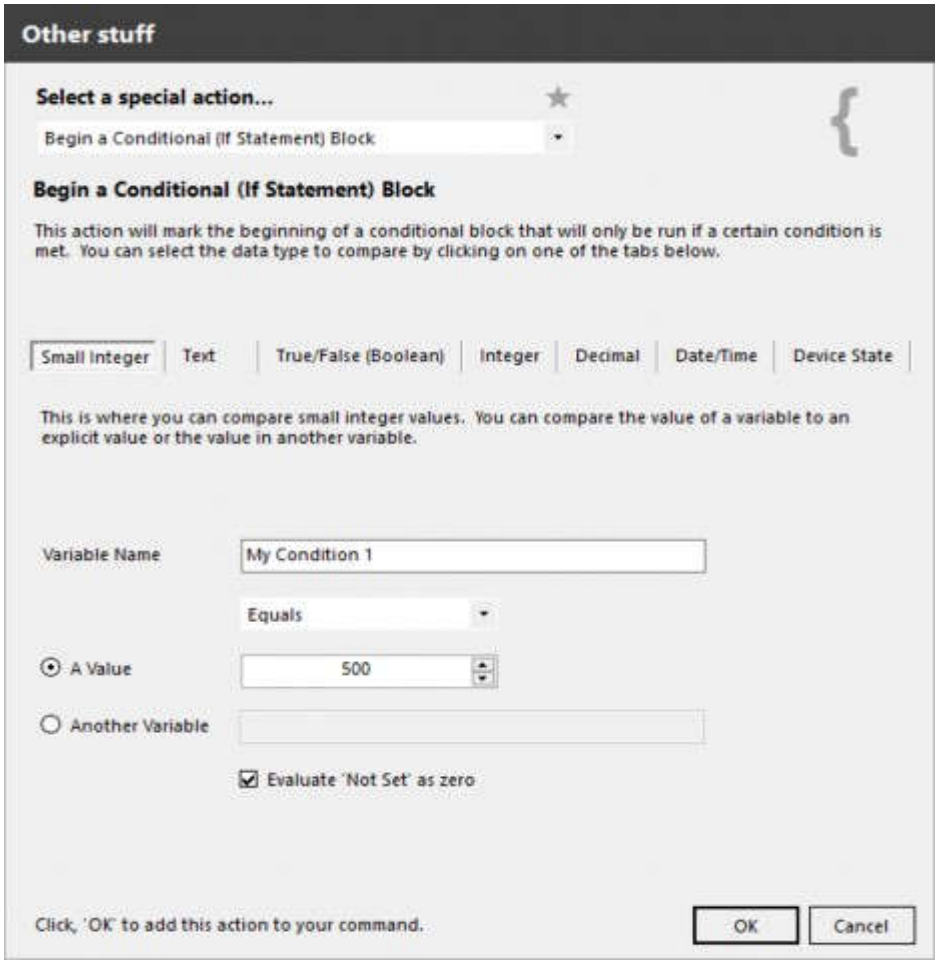
'Begin a Conditional (If Statement) Block'
「条件付き（Ifステートメント）ブロックの開始」

This command action is what you will use to begin a conditional block. Think of a conditional block as a simple, 'IF' statement. This block MUST be used with a corresponding End Conditional Block action (below).
このコマンドアクションは、条件ブロックを開始するために使用するものです。条件付きブロックは、単純な「IF」ステートメントと考えてください。このブロックは、対応するEnd Conditional Blockアクション（下記）とともに使用する必要があります。

Basically, the command actions that occur between the Begin and End Conditional Blocks will ONLY be executed if the comparison you define meets the criteria that you indicate in the Begin Conditional Block.
基本的に、開始条件ブロックと終了条件ブロックの間で発生するコマンドアクションは、定義した比較が開始条件ブロックで指定した基準を満たす場合にのみ実行されます。

You will need to first indicate what type of comparison you are going to make. If you are going to compare small integers, select the, 'Small Integer Tab'. If you are going to compare the text in a text variable, select the, 'Text' tab (and so on for the other data types).
まず、どのタイプの比較を行うのかを示す必要があります。小さい整数を比較する場合は、「小さい整数タブ」を選択します。テキスト変数のテキストを比較する場合は、[テキスト]タブを選択します（他のデータ型についても同様です）。

Small Integer (Condition) Value Comparison
小整数（条件）値の比較



If you are wanting to compare small integer (formerly called, 'conditions') values and chose this tab, the next step is to indicate what variable you are going to check by typing its name in the Variable Name field (you would have set this value in the, 'Set a
小さな整数（以前は「条件」と呼ばれていました）値を比較してこのタブを選択したい場合、次のステップは、変数名フィールドに名前を入力して、チェックする変数を指定することです（これを設定します）の値、 'Set a

Small Integer (Condition) Value' action... see above).
小さい整数（条件）値のアクション...上記を参照）。

Next, you will need to establish how the variable is going to be compared, by dropping down the, 'operator' box. You can choose Equals, Does Not Equal, Greater Than, Less Than, Is Set and Is Not Set (a variable is, 'Set' if a value has actually been
次に、「演算子」ボックスをドロップダウンして、変数の比較方法を確立する必要があります。「等しい」、「等しくない」、「より大きい」、「より小さい」、「設定」および「設定しない」を選択できます（値が実際に設定されている場合、変数は「設定」です

assigned to the variable. In programming-speak, the value would be considered null or not null).
変数に割り当てられます。プログラミング言語では、値はnullまたはnullでないと見なされます）。

The last thing you will need to do is to indicate the value that you are going to compare the variable to. This can be an explicit value (by selecting, 'A Value' and filling in the box), or the value of another small integer variable that you had set up (by selecting, 'Another Variable' and typing the name of that variable in the box provided).
最後に行う必要があるのは、変数と比較する値を示すことです。これは、明示的な値（「A Value」を選択してボックスに入力する）、または設定した別の小さな整数変数の値（「Another Variable」を選択して、その変数の名前を提供されたボックス）。

Optionally, you can select the, “Evaluate, ‘Not Set’ as zero” feature that will automatically evaluate the included variables (variables indicated in both, ‘Variable Name’ as well as, ‘Another Variable’) as zero when comparing. If you want to compare variables as null (‘Not Set’) when they are null (‘Not Set’), make sure to uncheck this option.
必要に応じて、「評価、ゼロとして設定しない」機能を選択して、比較するときに含まれる変数(「変数名」と「別の変数」の両方で示される変数)をゼロとして自動的に評価できます。変数がヌル(「未設定」)である場合にヌル(「未設定」)として変数を比較する場合は、このオプションをオフにしてください。

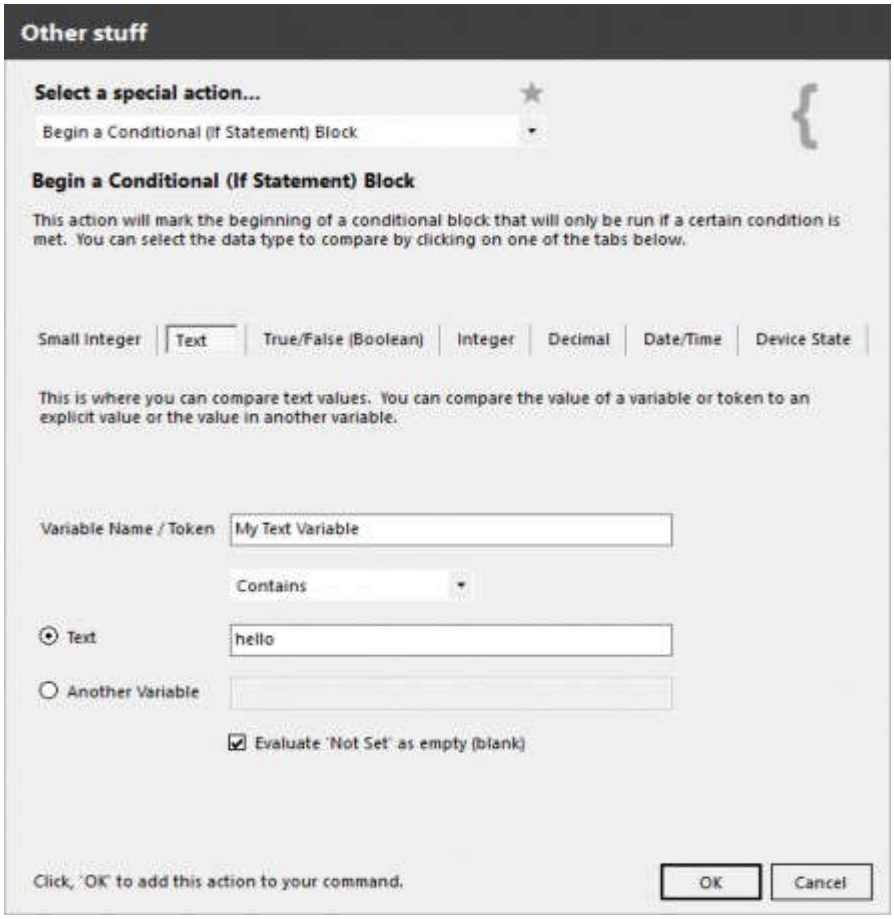
If the comparison is made and the condition is met, the command action immediately following the Begin a Conditional Block action will be executed. If the condition is NOT met, the command action immediately following a corresponding Else or the corresponding End Conditional Block will be executed (this is why the End Conditional Block is required).
比較が行われ、条件が満たされた場合、条件付きブロックの開始アクションの直後のコマンドアクションが実行されます。条件が満たされない場合、対応するElseまたは対応する終了条件ブロックの直後のコマンドアクションが実行されます(これが終了条件ブロックが必要な理由です)。

Note: If a variable that is being compared is NOT SET, the comparison will always result as false. So, if 'My Condition 1' is not set, and you try to compare it to 0 by setting the operator as, 'not equal to', the result will be false and the code will continue after the end block (see information above regarding evaluating, ‘Not Set’ as empty (blank)).
注:比較される変数が設定されていない場合、比較は常にfalseになります。したがって、「My Condition 1」が設定されておらず、演算子を「等しくない」に設定して0と比較しようとすると、結果はfalseになり、コードは終了ブロックの後に続きます(上記の情報を参照)評価に関して、「未設定」は空(空白)として)。

Note: To create a simple, single condition, ‘If’ statement, choose the, ‘Single Condition’ option from the menu. To create a compound, ‘If’ statement (that is, an ‘If’ statement that contains multiple conditions (‘and’ and ‘or’), select the, ‘Compound Condition Builder’ option from the menu. For more information regarding compound conditions, see the section titled, ‘Using the Condition Builder’ later in this document.
注:単純な単一条件「If」ステートメントを作成するには、メニューから「単一条件」オプションを選択します。複合の「If」ステートメント(つまり、複数の条件(「and」および「or」)を含む「If」ステートメントを作成するには、メニューから「Compound Condition Builder」オプションを選択します。条件については、このドキュメントで後述する「条件ビルダーの使用」というタイトルのセクションを参照してください。

Also note that you can convert a single conditional statement to a compound statement by right clicking on the action on the command screen and choosing the, ‘Edit with condition builder’ option.
また、コマンド画面でアクションを右クリックし、「条件ビルダーで編集」オプションを選択すると、単一の条件ステートメントを複合ステートメントに変換できることに注意してください。

Text Value Comparison
テキスト値の比較



If you are wanting to compare a text variable and chose this tab, the next step is to indicate what text variable you are going to check by typing its name in the, 'Variable Name / Token' field (you would have set this value in the, 'Set a Text Value'
テキスト変数を比較してこのタブを選択する場合、次のステップは、「変数名/トークン」フィールドに名前を入力して、チェックするテキスト変数を示すことです（この値を設定します） the、「テキスト値の設定」

action… see way below). Note that this box can accept tokens as well. This way, you can compare either variable values or the value of a rendered text token.
アクション...以下の方法を参照）。このボックスはトークンも受け入れることができることに注意してください。これにより、変数値またはレンダリングされたテキストトークンの値を比較できます。

Next, you will need to establish how the variable is going to be compared, by dropping down the, 'operator' box. You can choose Equals, Does Not Equal, Starts With, Does Not Start With, Ends With, Does Not End With, Contains, Does Not Contain, Is Set and Is Not Set (a text variable is, 'Set' if a value has actually been assigned to the variable. If you are a programmer, you would refer to this as the variable being null or not null).
次に、「演算子」ボックスをドロップダウンして、変数の比較方法を確立する必要があります。「等しい」、「等しくない」、「で始まる」、「で始まらない」、「で終わる」、「で終わる」、「含む」、「含まない」、「設定されている」および「設定されていない」を選択できます（値が実際に変数に割り当てられています。プログラマーである場合は、これを変数がnullであるか、nullでないと呼びます）。

The last thing you will need to do is to indicate the value that you are going to compare the text variable to. This can be an explicit value (by selecting, 'Text' and filling in
最後に行う必要があるのは、テキスト変数と比較する値を示すことです。これは明示的な値にすることができます（「テキスト」を選択し、

the box), or the value of another text variable that you had set up (by selecting, 'Another Variable' and typing the name of that variable in the box provided).
ボックス)、または設定した別のテキスト変数の値(「別の変数」を選択し、表示されたボックスにその変数の名前を入力します)。

Optionally, you can select the, "Evaluate, 'Not Set' as empty (blank)" feature that will automatically evaluate the included variables/tokens (variables indicated in both, 'Variable Name/Token' and 'Another Variable', as well as the converted value of what is
オプションで、含まれる変数/トークン(「変数名/トークン」と「別の変数」の両方で示される変数)を自動的に評価する、「評価、「空(空白)として設定しない」」機能を選択できます。何の変換値として

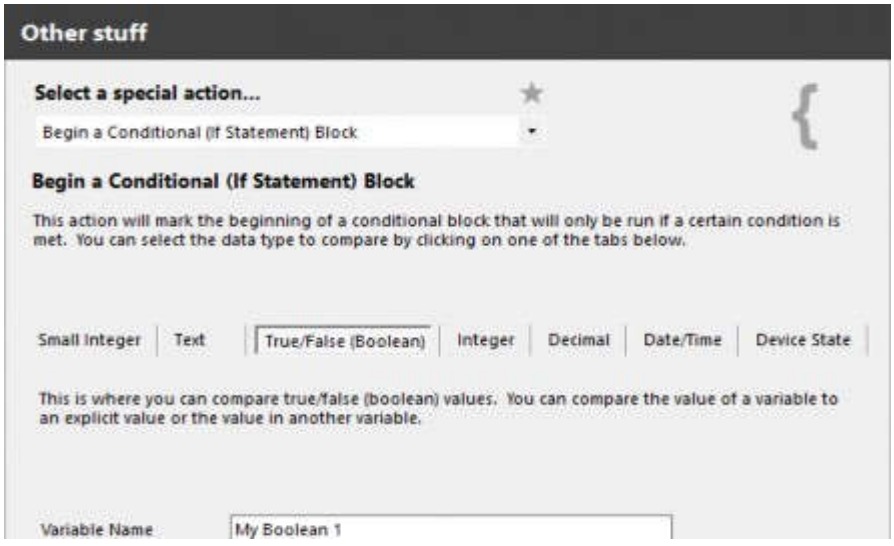
placed in the Token field) as empty (blank) when comparing. If you want to compare variables as null ('Not Set') when they are null ('Not Set'), make sure to uncheck this option.
比較するとき、空(空白)としてトークンフィールドに配置されます。変数がヌル(「未設定」)である場合にヌル(「未設定」)として変数を比較する場合は、このオプションをオフにしてください。

If the comparison succeeds, the command action immediately following the Begin a Conditional Block action will be executed. If the comparison fails, the command action immediately following a corresponding Else or the corresponding End Conditional Block will be executed (this is why the End Conditional Block is required).
比較が成功すると、条件ブロックの開始アクションの直後のコマンドアクションが実行されます。比較が失敗した場合、対応するElseまたは対応する終了条件ブロックの直後のコマンドアクションが実行されます(これが終了条件ブロックが必要な理由です)。

Note: If a text value variable that is being compared is NOT SET, the comparison will always result as false (see information above regarding evaluating, 'Not Set' as empty (blank)).
注:比較されているテキスト値変数がNOT SETの場合、比較は常にfalseになります(評価に関する上記の情報を参照してください、「Not Set」を空(空白)として)

Note: The 'Text' option also accepts tokens.
注:「テキスト」オプションもトークンを受け入れます。

True/False (Boolean) Comparison
真/偽(ブール)比較



If you are wanting to compare a true/false (Boolean) variable and chose this tab, the next step is to indicate what Boolean variable you are going to check by typing its name in the, 'Variable Name' field (you would have set this value in the, 'Set a True/False (Boolean) Value' action… see way below).
真/偽(ブール)変数を比較してこのタブを選択する場合、次のステップは、「変数名」フィールドに名前を入力して、チェックするブール変数を示すことです(設定します)この値は、「True / False(ブール)値の設定」アクションで…次の方法を参照してください)。

Next, you will need to establish how the variable is going to be compared, by dropping down the, 'operator' box. You can choose Equals, Does Not Equal, Is Set
次に、「演算子」ボックスをドロップダウンして、変数の比較方法を確立する必要があります。「等しい」、「等しくない」、「設定」を選択できます

and Is Not Set (a Boolean variable is, 'Set' if a value has actually been assigned to the variable. If you are a programmer, you would refer to this as the variable being null or not null).
およびIs Not Set(ブール変数は、値が実際に変数に割り当てられている場合は「Set」です。プログラマーの場合、これは変数がnullまたはnullでないことを指します)。

The last thing you will need to do is to indicate the value that you are going to compare the Boolean variable to. This can be an explicit value (by selecting, True or False from the 'Value' field), or the value of another Boolean variable that you had set up (by selecting, 'Another Variable' and typing the name of that variable in the box provided).
最後に行う必要があるのは、ブール変数と比較する値を示すことです。これは、明示的な値(「値」フィールドからTrueまたはFalseを選択することにより)、または設定した別のブール変数の値(「別の変数」を選択して、その変数の名前をボックスが提供されます)。

Optionally, you can select the, "Evaluate, 'Not Set' as false" feature that will automatically evaluate the included variables (variables indicated in both, 'Variable Name' as well as, 'Another Variable') as false when comparing. If you want to compare variables as null ('Not Set') when they are null ('Not Set'), make sure to uncheck this option.
オプションで、含まれる変数(「変数名」と「別の変数」の両方で示される変数)を比較するときにfalseとして自動的に評価する「評価、「設定しない」をfalseにする」機能を選択できます。変数がヌル(「未設定」)である場合にヌル(「未設定」)として変数を比較する場合は、このオプションをオフにしてください。

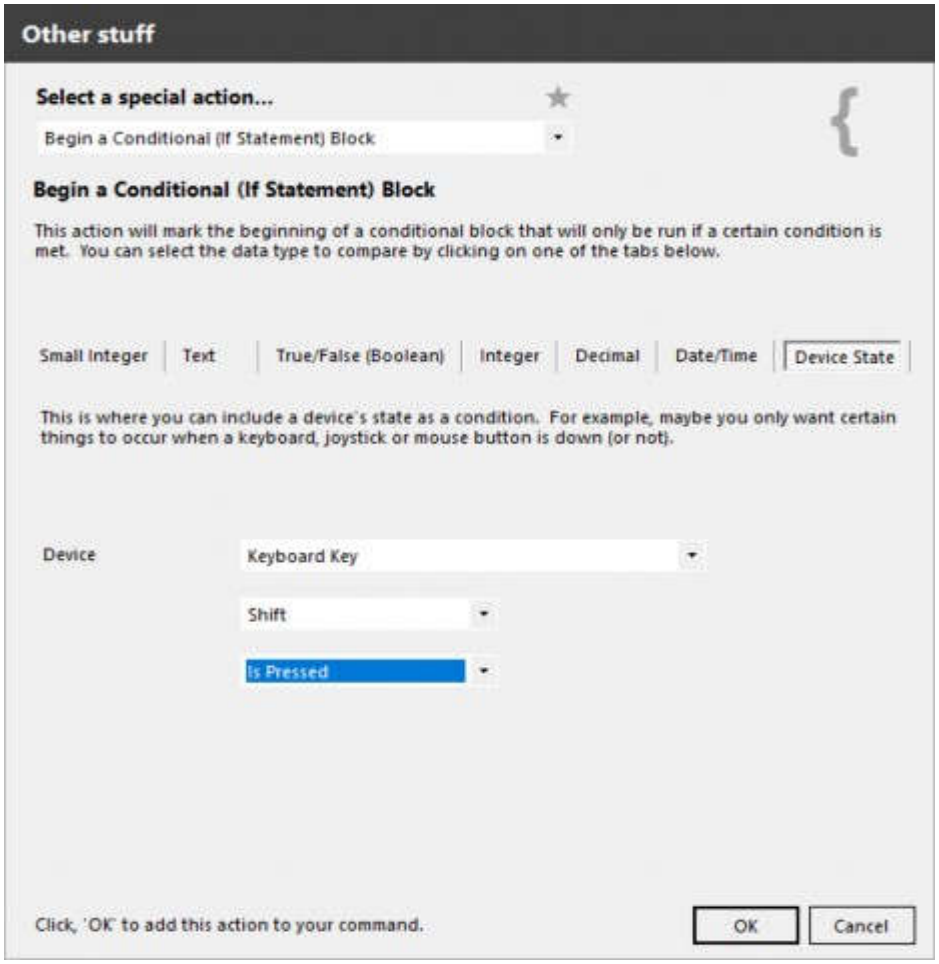
If the comparison succeeds, the command action immediately following the Begin a Conditional Block action will be executed. If the comparison fails, the command action immediately following a corresponding Else or the corresponding End Conditional Block will be executed (this is why the End Conditional Block is required).
比較が成功すると、条件ブロックの開始アクションの直後のコマンドアクションが実行されます。比較が失敗した場合、対応するElseまたは対応する終了条件ブロックの直後のコマンドアクションが実行されます(これが終了条件ブロックが必要な理由です)。

Note: If a True/False (Boolean) variable that is being compared is NOT SET, the comparison will always result as false (see information above regarding evaluating, ‘Not Set’ as empty (blank)).
注：比較されるTrue / False（ブール）変数がNOT SETの場合、比較は常にfalseになります（評価に関する上記の情報を参照してください、「Not Set」は空（空白）として）。

Integer, Decimal and Date/Time Value Comparison
整数、10進数、および日付/時刻値の比較

If you would like to compare the values of integers, decimal and date/time variables, simply select the appropriate tab. The functionality of these three options are basically the same as the Small Integer comparisons indicated above (so, we'll save a tree and not duplicate even more stuff... lol.).
整数、10進数、および日付/時刻変数の値を比較する場合は、適切なタブを選択するだけです。これらの3つのオプションの機能は、上記のSmall Integer比較と基本的に同じです（したがって、ツリーを保存し、さらに多くのものを複製しません...笑）。

Device State
デバイス状態



The Device State condition allows you to check the state of your keyboard keys,
デバイス状態の状態では、キーボードのキーの状態を確認できます。

mouse buttons and joystick buttons and have your command be able to react to these states. For instance, you may want to check if a certain key is down when you say, ‘Fire Weapons’. Your command could do something differently if the, ‘X’ key is down rather than if it is not. Also, you can check the state of your mouse and/or joystick buttons. So, your, ‘Fire Weapons’ command could behave totally differently if Shift, Right Mouse Button and Joystick Button 7 are down ;) Simply choose the device you would like to check, then select the key, button or position. Next, select whether you want to check if the button/key is pressed or not pressed. Note that you can also check to see if the key/button you are checking is the ONLY key/button pressed, by selecting the, ‘Only key/button pressed’ option. To do a device-wide check to see if any key/button is pressed at all, select the, ‘<Any key/button>’. To do a device-wide check to see if no keys/buttons are pressed, select the, ‘<No key/button>’ option.
マウスボタンとジョイスティックボタンを使用し、コマンドがこれらの状態に反応できるようにします。たとえば、「Fire Weapons」と言ったときに特定のキーがダウンしているかどうかを確認できます。「X」キーが押されていない場合ではなく、押されている場合、コマンドの動作が異なる可能性があります。また、マウスやジョイスティックのボタンの状態を確認できます。そのため、Shift、右マウスボタン、ジョイスティックボタン7がダウンしている場合、「Fire Weapons」コマンドの動作はまったく異なる可能性があります;)チェックするデバイスを選択し、キー、ボタン、または位置を選択します。次に、ボタン/キーが押されているかどうかを確認するかどうかを選択します。[キー/ボタンのみを押す]オプションを選択して、チェックするキー/ボタンが押されたキー/ボタンのみであるかどうかを確認することもできます。キー/ボタンがまったく押されていないかどうかをデバイス全体で確認するには、「<Any key / button>」を選択します。キー/ボタンが押されていないかどうかをデバイス全体で確認するには、「<キー/ボタンなし>」オプションを選択します。

Mouse and joystick positions can also be checked. So, if your mouse enters an area of the screen, or if your joystick is pushed to the limit, VoiceAttack can be set up to react. Simply choose the device and position aspect you would like to check and indicate the value in the box at the bottom. Note that this box can contain integer values, variable names of variables that resolve to an integer, tokens that resolve to an integer or tokens that resolve to variable names that resolve to an integer (whew).
マウスとジョイスティックの位置も確認できます。そのため、マウスが画面の領域に入った場合、またはジョイスティックが限界まで押された場合、VoiceAttackは反応するように設定できます。デバイスを選択し、チェックしたいアスペクトを配置し、下部のボックスに値を指定します。このボックスには、整数値、整数に解決される変数の変数名、整数に解決されるトークン、または整数に解決される変数名に解決されるトークン（whew）を含めることができます。

Note that the mouse has two different aspects: screen and app/window. Use, ‘screen’ when you want your coordinates to relate to the entire screen ((0,0) would be the top- left corner of the screen). Use, ‘app/window’ when you want the coordinates to relate to the active window ((0,0) would be the top-left corner of the active window). Also note that the value of the joystick positions will be -1 if the joystick position cannot be accessed.
マウスには、画面とアプリ/ウィンドウの2つの異なる側面があることに注意してください。座標を画面全体に関連付ける場合は、「screen」を使用します（(0,0)は画面の左上隅になります）。座標をアクティブウィンドウに関連付ける場合は、「app / window」を使用します（(0,0)はアクティブウィンドウの左上隅になります）。また、ジョイスティックの位置にアクセスできない場合、ジョイスティックの位置の値は-1になります。

'Add Else If to a Conditional Block'
「条件ブロックにElse Ifを追加」

This action will allow you to add an, 'Else If' to your condition block. That is, if the result of the beginning, 'If' statement is false (condition not met), you can use an Else If block to do another comparison. You can have as many, 'Else If' blocks as you wish between the Begin and End Conditional block actions. The options for the, 'Else If' are the same as what you find in the 'Begin a conditional (if statement) block'. If all the 'Else If' blocks do not have their conditions met, the command will then go to an existing, 'Else' action or to the End (if there are no, 'Else' actions for the containing conditional block).
このアクションにより、条件ブロックに「Else If」を追加できます。つまり、先頭の「If」ステートメントの結果がfalse（条件が満たされない）の場合、Else Ifブロックを使用して別の比較を実行できます。条件ブロックの開始アクションと終了ブロックアクションの間に、必要な数の「Else If」ブロックを含めることができます。「Else If」のオプションは、「Begin a condition（if statement）block」で見つけるものと同じです。すべての「Else If」ブロックの条件が満たされていない場合、コマンドは既存の「Else」アクションまたは最後に移動します（存在しない場合は、包含条件ブロックの「Else」アクション）。

To create a simple, single condition, ‘Else If’ statement, choose the, ‘Single Condition’ option from the menu. To create a compound, ‘Else If’ statement (that is, an, ‘Else If’ statement that contains multiple conditions), select the, ‘Compound Condition Builder’ option from the menu. For more information regarding compound conditions, see the section titled, ‘Using the Condition Builder’ later in this document.
単純な単一条件の「Else If」ステートメントを作成するには、メニューから「単一条件」オプションを選択します。複合の「Else If」ステートメント（つまり、複数の条件を含む「Else If」ステートメント）を作成するには、メニューから「Compound Condition Builder」オプションを選択します。複合条件の詳細については、このドキュメントで後述する「条件ビルダーの使用」というタイトルのセクションを参照してください。

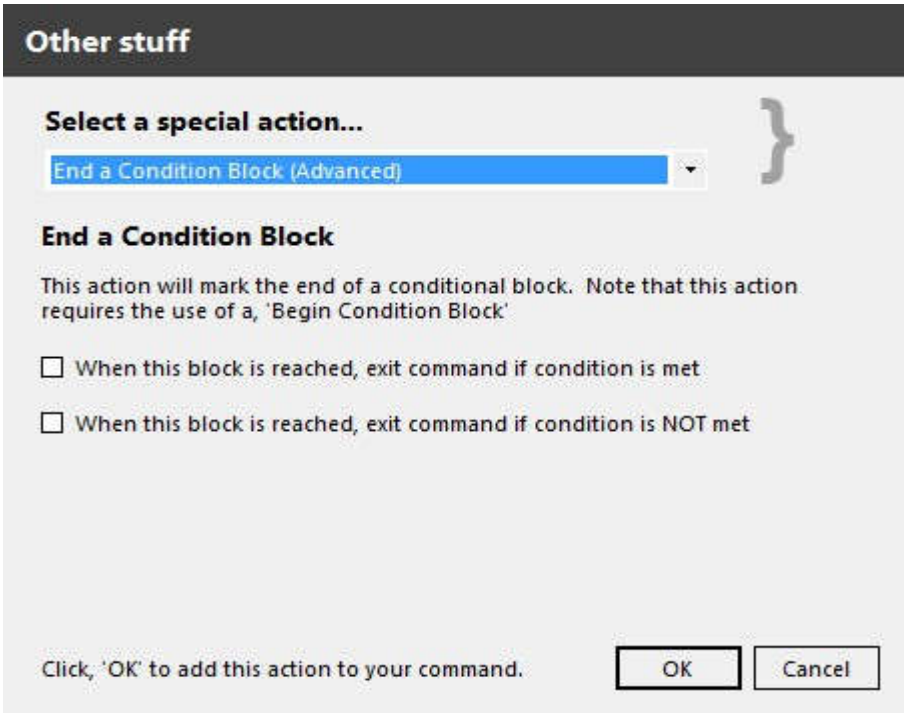
'Add Else to a Conditional Block'
「条件ブロックにその他を追加する」

You can add an 'Else' action to direct the flow of your condition block. If the result of the conditional block is true, all actions in the conditional block ABOVE the, 'Else' will be executed. When the actions between the start of the conditional block and the, 'Else' are finished, the command will jump down to (and execute) the, 'End Conditional
「Else」アクションを追加して、条件ブロックのフローを指示できます。条件ブロックの結果がtrueの場合、条件ブロック内のすべてのアクション、「Else」が実行されます。条件ブロックの開始と「Else」の間のアクションが終了すると、コマンドは「End Conditional」にジャンプ（および実行）します。

Block' action.
ブロック」アクション。

When a conditional block does not meet the requirements (result is false), all actions between the, 'Else' action, up to and including the 'End Conditional Block' action will be executed.
条件ブロックが要件を満たさない場合（結果はfalse）、「Else」アクションから「条件ブロックの終了」アクションまでのすべてのアクションが実行されます。

'End a Conditional Block'
「条件ブロックの終了」



This command action is what you will use to end a condition block. This MUST be used in conjunction with a Begin Condition Block action (see above).
このコマンドアクションは、条件ブロックを終了するために使用するものです。これは、Begin Condition Blockアクションと組み合わせて使用する必要があります（上記を参照）。

When a Condition Start Block action is executed, a value is checked to see if a condition is met. If the condition is met, all the actions between the Condition Start Block and the End Condition block are executed. If the condition is not met, the code following the End Condition Block is executed (everything between the Condition Start and Condition End blocks is skipped).
条件開始ブロックアクションが実行されると、値がチェックされ、条件が満たされているかどうかが確認されます。条件が満たされると、条件開始ブロックと終了条件ブロックの間のすべてのアクションが実行されます。条件が満たされない場合、End Conditionブロックに続くコードが実行されます（Condition StartブロックとCondition Endブロックの間のすべてがスキップされます）。

If you select either of the options indicated in the End Condition block, you can have one more check before processing resumes after the block. You have the option of completely exiting the command if the condition is or is not met. One thing this is useful for is to bypass a long list of condition checks.
End Conditionブロックに示されているオプションのいずれかを選択すると、ブロックの後に処理を再開する前にもう1つのチェックを行うことができます。条件が満たされているかどうかに応じて、コマンドを完全に終了するオプションがあります。これが役立つのは、条件チェックの長いリストをバイパスすることです。

A sample conditional block with Else If and Else actions is below: Set 'myTextVariable' to 'howdy'
Else IfおよびElseアクションを含むサンプルの条件ブロックは以下のとおりです。「myTextVariable」を「howdy」に設定します

Begin Text Compare 'myTextVariable' equals, 'hello'
テキスト比較を開始 'myTextVariable'が等しい、 'hello'

Say, 'You said hello'
「こんにちは」と言います

Else If 'myTextVariable' equals, 'greetings'
それ以外の場合、「myTextVariable」が等しい場合、「greetings」

Say, 'You said greetings'
「挨拶を言った」と言う

Else If 'myTextVariable' contains, 'howdy'
その他「myTextVariable」に「howdy」が含まれる場合

Begin Text Compare 'myTextVariable' contains 'partner'
テキスト比較を開始 'myTextVariable'に 'partner'が含まれる

Say, 'You said howdy and I'm not your partner'
「あなたはハウディと言ったが、私はあなたのパートナーではない」と言う

Else
その他

Say, 'You said howdy'
「あなたはハウディと言った」と言う

End Condition Else
その他の終了条件

Say, 'You didn't say anything I recognize'
「あなたは私が認識したことを何も言わなかった」と言う

End Condition
終了条件

In this example, TTS would say, 'You said howdy'. If, 'myTextVariable' was set to 'hola', TTS would say, 'You didn't say anything I recognize'. Note that the conditional blocks can contain other conditional blocks (nested).
この例では、TTSは「あなたはハウディと言いました」と言います。「myTextVariable」が「hola」に設定されている場合、TTSは「あなたは私が認識していることを何も言わなかった」と言うでしょう。条件ブロックには、他の条件ブロック（ネスト）を含めることができます。

'Add a Loop Start'
「ループスタートの追加」

There are a few options when adding loops: Single-condition, 'while' loops, compound- condition, 'while' loops, and loops that repeat a certain number of times.
ループを追加する場合、単一条件、「while」ループ、複合条件、「while」ループ、特定の回数繰り返すループなどのオプションがあります。

- Loop Start - Single Condition (While Loop)
- ループ開始-単一条件(ループ中)

This action indicates the beginning of a looping block that will continue looping as long as the indicated condition is met. The options for the loop condition are identical to the ones found in 'Begin a Conditional (If Statement) Block' (see above), so, they will not be repeated here (there's a lot of options (lol)). All actions between the Loop Start and Loop End will be repeated. Once the condition is not met, the command flow will go to the command action immediately following the Loop End (see below). Note that loops can contain other loops (nested loops).
このアクションは、示された条件が満たされる限りループを継続するループブロックの開始を示します。ループ条件のオプションは、「条件付き(Ifステートメント)ブロックの開始」(上記を参照)にあるオプションと同じであるため、ここでは繰り返されません(多くのオプションがあります(笑))。ループ開始とループ終了の間のすべてのアクションが繰り返されます。条件が満たされない場合、コマンドフローはループエンドの直後のコマンドアクションに移動します(以下を参照)。ループには他のループ(ネストされたループ)を含めることができます。

- Loop Start - Compound Condition (While Loop)
- ループスタート-複合条件(ループ中)

This action works similarly to the previously-mentioned, 'Begin a Conditional (If Statement) Block' screens, with the difference being that multiple (compound) conditions can be used to build a Loop Start. There is a lot to this screen, so in order to keep this description short, please see the section, 'Using the Condition Builder' section later in this document.
このアクションは、前述の「条件(Ifステートメント)ブロックの開始」画面と同様に機能しますが、複数の(複合)条件を使用してループスタートを構築できる点が異なります。この画面には多くの機能があるため、この説明を短くするために、このドキュメントで後述する「条件ビルダーの使用」セクションを参照してください。

- Loop Start - Repeat a Certain Number of Times (For Loop)
- ループスタート-特定の回数を繰り返す(Forループ)

This action has two different options. The first and easiest option is to repeat a block a specified number of times. The second, more advanced option is to repeat a block a number of times using a range (and optional indexer).
このアクションには2つの異なるオプションがあります。最初の最も簡単なオプションは、ブロックを指定された回数だけ繰り返すことです。2番目のより高度なオプションは、範囲(およびオプションのインデクサー)を使用してブロックを何度も繰り返すことです。

Using the first option labeled, ‘Number of times’, you can simply specify how many times you would like a block of actions repeated. In the input box, you can specify the whole number to indicate the number of times to repeat the loop block. This box also accepts integer (large) variables. When the loop start is first encountered, the value in the variable is resolved and if it is valid, it is used to indicate how many times the block will be repeated. This box also accepts any combination of tokens. If the tokens render into a valid whole number, that value will be used. If the tokens render into a valid variable name, the variable will be resolved and its value used. Note: The value can be a negative number and still be valid. So, if the value is -3, the loop will repeat 3 times. Hint: If you
「回数」というラベルの付いた最初のオプションを使用すると、アクションのブロックを繰り返す回数を簡単に指定できます。入力ボックスでは、整数を指定して、ループブロックを繰り返す回数を指定できます。このボックスは、整数（大）変数も受け入れます。ループの開始が最初に検出されると、変数の値が解決され、有効な場合、ブロックが繰り返される回数を示すために使用されます。このボックスは、トークンの任意の組み合わせも受け入れます。トークンが有効な整数にレンダリングされる場合、その値が使用されます。トークンが有効な変数名にレンダリングされる場合、変数は解決され、その値が使用されます。注意：値は負の数でも有効です。したがって、値が-3の場合、ループは3回繰り返されます。ヒント：あなたが

need to bail out of this type of loop at any time, try using, ‘Jumps’ and ‘Jump Markers’.
いつでもこのタイプのループから脱出する必要がある場合は、「ジャンプ」と「ジャンプマーカー」を使用してみてください。

The second option is labeled as, ‘Range (For Loop)’. This option most resembles what is known as a, ‘for’ loop in programmer-speak. The loop will repeat from the value indicated in the, 'From' box (inclusive) to the value indicated the 'To' box (inclusive) (Note that these two boxes are also overloaded to work just like the ‘Number of times’ box (above)). So, if you put a value that resolves to 3 in the, ‘From’ box and a value that resolves to 7 in the, ‘To’ box, the block will repeat 5 times (remember, the values are inclusive). Note that either of these boxes can be positive or negative, so, having a, ‘From’ value as -3 and a, ‘To’ value as -5, the block will repeat 3 times. The, ‘Range’ feature also allows you to include an optional integer (large) variable to include as an indexer. The indexer variable will hold the current value of the loop’s index.
2番目のオプションには、「範囲（Forループ）」というラベルが付いています。このオプションは、プログラマーに言えば「for」ループと呼ばれるものに最も似ています。ループは、「From」ボックス（包括的）に示されている値から「To」ボックス（包括的）に示されている値まで繰り返されます（これら2つのボックスも、「回数」ボックスと同様に機能するようにオーバーロードされていることに注意してください）（上記））。したがって、「From」ボックスに3に解決される値と「To」ボックスに7に解決される値を入力すると、ブロックは5回繰り返されます（値は包括的であることに注意してください）。これらのボックスのいずれかが正または負になる可能性があるため、「From」値が-3で、「To」値が-5の場合、ブロックは3回繰り返されます。「範囲」機能では、インデクサーとして含めるオプションの整数（大）変数を含めることもできます。。インデクサー変数は、ループのインデックスの現在の値を保持します。

The loop’s index is incremented by 1 on each iteration of the loop. So, if you have 1 in the, ‘From’ box and ‘5’ in the, ‘To’ box, the indexer variable’s value the first time through the loop will be 1. The index then increments by 1, so the second time through the indexer variable’s value will be 2. The third time through it will be 3, then 4, then 5.
ループのインデックスは、ループの各反復で1ずつ増加します。したがって、「From」ボックスに1があり、「To」ボックスに「5」がある場合、ループの最初のインデクサー変数の値は1になります。その後、インデックスは1ずつ増加します。インデクサー変数の値は2になります。3回目は3、4、5の順になります。

Spoiler alert – Advanced (if it wasn't enough already): Since the indexer variable's value can also be altered to indicate the loop's index, you can reset or, 'step' the index if you need to. If an indexer variable's value is altered, the loop's index will be set to match the indexer variable's value on the next iteration of the loop. Here are some examples. Let's say you have 1 in the, 'From' box and 10 in the, 'To' box. If you have a condition within your loop where you need to start the loop over, you can set the indexer variable's value to 1. On the next iteration of the loop, the index will now be 1 and the loop will act like it has started over. If you need to bail out of the loop (and don't want to use jumps/jump markers), you can set the value to 11 and the loop will exit before the next iteration starts (since the index will be greater than the value of 10 in the, 'To' box). If you need to, 'step' by a certain number, simply add that number to the indexer variable's value. So, if you have 2 in the, 'From' box and 10 in the, 'To' box and you want to just index by even numbers, simply add 2 to the indexer variable on each iteration. Hope I didn't just scare you off. Come visit everybody in the VoiceAttack User Forum and maybe we can clear all this up ;)
スポイラーアラート–詳細(まだ十分でない場合):インデクサー変数の値もループのインデックスを示すように変更できるため、必要に応じて、インデックスをリセットまたは「ステップ実行」できます。インデクサー変数の値が変更された場合、ループのインデックスは、ループの次の反復でインデクサー変数の値と一致するように設定されます。下記は用例です。「From」ボックスに1つ、「To」ボックスに10個あるとします。ループ内にループを開始する必要がある条件がある場合、インデクサー変数の値を1に設定できます。ループの次の反復で、インデックスは1になり、ループは開始したように動作します以上。ループから脱出する必要がある場合(ジャンプ/ジャンプマーカーを使用したくない場合)、値を11に設定すると、次の反復が始まる前にループが終了します(インデックスが値より大きいため)「To」ボックスに10を入力)。特定の番号で「ステップ」する必要がある場合は、その番号をインデクサー変数の値に追加するだけです。したがって、「From」ボックスに2個、「To」ボックスに10個あり、偶数でインデックスを作成する場合は、各反復でインデクサー変数に2を追加するだけです。私はあなたを追い払うだけではなかったと思います。VoiceAttackユーザーフォーラムの全員にアクセスしてください。これをすべてクリアできるかもしれません;)

Note the, 'Indexer' box can also contain any number of tokens that can resolve to a variable name.
「インデクサー」ボックスには、変数名に解決できるトークンをいくつでも含めることができます。

'Add a Loop End'
「ループエンドの追加」

Use in conjunction with a, 'Loop Start' action to indicate the end of the block that requires looping. Command flow will go to the command action immediately following this action when the loop condition is not met.
「ループ開始」アクションと組み合わせて使用して、ループを必要とするブロックの終了を示します。ループ条件が満たされない場合、コマンドフローはこのアクションの直後のコマンドアクションに移動します。

'Add a Loop Break'
「ループブレークの追加」

If a, 'Loop Break' action is encountered within a looping section, control flow is then moved to the end of the containing loop.
ループセクション内で「ループブレーク」アクションが発生した場合、制御フローはループを含むループの最後に移動します。

'Add a Jump Marker'
「ジャンプマーカーを追加する」

This command action lets you place a marker within your command that will indicate a location that can be, 'jumped' to (using a 'Jump' command action).
このコマンドアクションを使用すると、コマンド内にマーカーを配置して、「ジャンプ」可能な場所を示すことができます（「ジャンプ」コマンドアクションを使用）。

Jump markers can be named whatever you want, but must be uniquely named within the command (if markers happen to be named the same in a prefix/suffix situation, the first marker will be the target of the jump). See, 'Add a Jump' (below) for more info.
ジャンプマーカーには任意の名前を付けることができますが、コマンド内で一意の名前を付ける必要があります（プレフィックス/サフィックスの状況でマーカーに同じ名前が付けられている場合、最初のマーカーがジャンプのターゲットになります）。詳細については、「ジャンプの追加」（下）を参照してください。

'Compound Condition Builder'
「複合条件ビルダー」

This action works similarly to the previously-mentioned, 'Begin a Conditional (If Statement) Block' screens, with the difference being that multiple (compound) conditions can be used to build a Conditional (If) Statement Block or Loop Start. There is a lot to this screen, so in order to keep this description short, please see the section, 'Using the Condition Builder' section later in this document.
このアクションは、前述の「条件付き（Ifステートメント）ブロックの開始」画面と同様に機能しますが、違いは、複数の（複合）条件を使用して条件付き（If）ステートメントブロックまたはループスタートを構築できることです。この画面には多くの機能があるため、この説明を短くするために、このドキュメントで後述する「条件ビルダーの使用」セクションを参照してください。

'Add a Jump'
「ジャンプを追加」

This action will instruct your running command to jump to another place within in your command's actions. You'll notice that you can jump to three different places: To a marker, to the exit of the command and to the start of the command. If you choose to jump to a marker, you can choose an existing marker from the dropdown list, or just type in a name (in case you haven't created the marker yet, or, if the marker exists in a corresponding prefix or suffix command and is not available). Note that this input box also supports the use of tokens (see token reference near the end of this document).
このアクションは、実行中のコマンドに、コマンドのアクション内の別の場所にジャンプするよう指示します。マーカー、コマンドの終了、コマンドの開始の3つの異なる場所にジャンプできることがわかります。マーカーにジャンプすることを選択した場合、ドロップダウンリストから既存のマーカーを選択するか、名前を入力するだけです（マーカーをまだ作成していない場合、またはマーカーが対応するプレフィックスまたはサフィックスに存在する場合）コマンドおよび利用できません）。この入力ボックスは、トークンの使用もサポートしていることに注意してください（このドキュメントの終わり近くにあるトークンリファレンスを参照）。

If you choose to jump to the exit, the command will do just that... exit. Note that any sub-commands that are currently executing will continue to execute (this is not a kill switch for the command).
出口にジャンプすることを選択した場合、コマンドはそれだけを行います...終了します。現在実行中のサブコマンドは引き続き実行されることに注意してください（これはコマンドのキルスイッチではありません）。

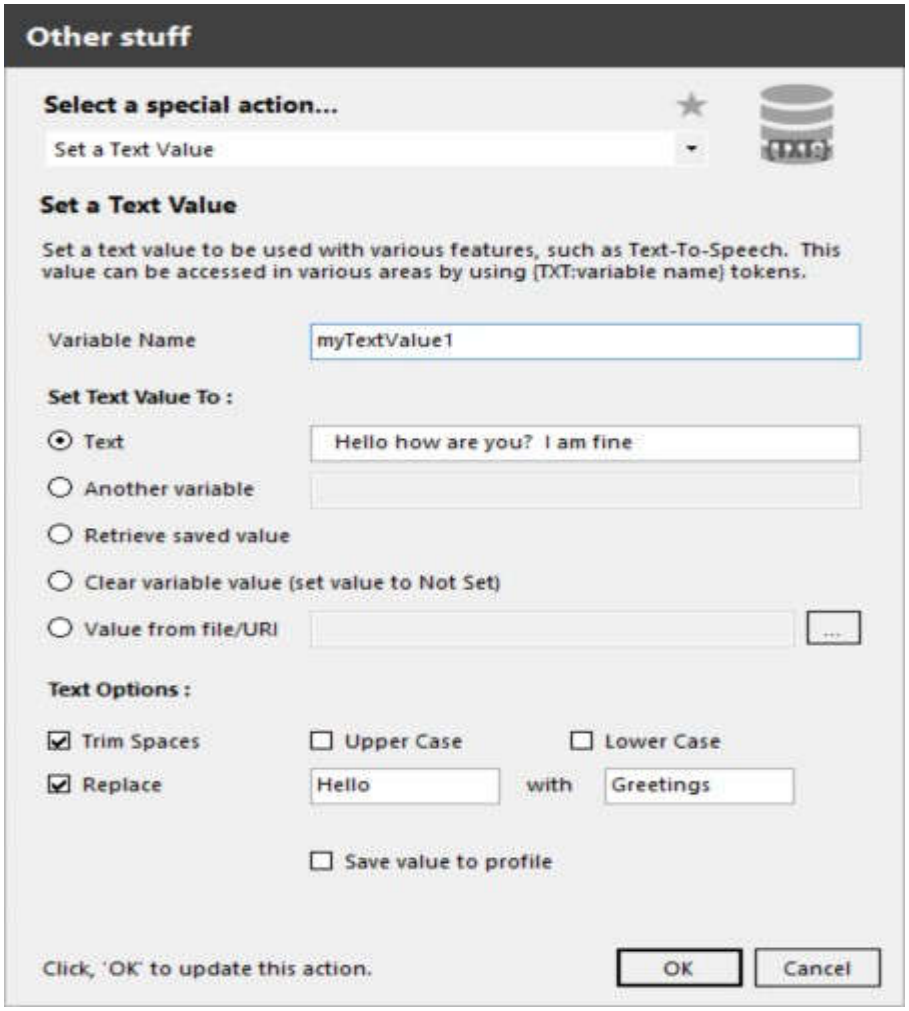
If you choose to jump to the start of the command, the command will continue processing from the start of the command.
コマンドの先頭にジャンプすることを選択した場合、コマンドはコマンドの先頭から処理を続行します。

Note that all jump types will work in prefix/suffix commands when they are executed together as a composite command.
すべてのジャンプタイプは、複合コマンドとして一緒に実行される場合、プレフィックス/サフィックスコマンドで機能することに注意してください。

'Exit Command'
「終了コマンド」

This action allows you to simply exit the command. This will not stop any executing sub-commands, as this is just a simple exit (not a kill command). Note: This is just a more obvious implementation of what was previously only available as an option in a Jump (see above).
このアクションにより、コマンドを単純に終了できます。これは単純な終了（killコマンドではない）であるため、実行中のサブコマンドは停止しません。注：これは、以前はジャンプのオプションとしてのみ使用可能であったものの、より明白な実装です（上記を参照）。

'Set a Text Value'
「テキスト値を設定」



This command action will allow you to set a text value to a named variable. These values can be accessed by special tokens in various places within VoiceAttack (such as Text-To-Speech, Launch Application paths/parameters/working directories, etc).
このコマンドアクションを使用すると、テキスト値を名前付き変数に設定できます。これらの値には、VoiceAttack内のさまざまな場所にある特別なトークン（Text-To-Speech、Launch Applicationパス/パラメーター/作業ディレクトリなど）からアクセスできます。

The Text Value Name can be whatever name you want. This value will be stored at the application level (shared among all profiles), so care must be made in order to make your name is unique enough so you don't overwrite your values accidentally. The text value names are not case-sensitive and they must not contain semicolons or colons.
テキスト値の名前は、任意の名前にすることができます。この値はアプリケーションレベル（すべてのプロファイルで共有）に保存されるため、名前が十分に一意になるように注意して、誤って値を上書きしないようにする必要があります。テキスト値の名前は大文字と小文字が区別されず、セミコロンまたはコロンを含めることはできません。

You can choose to set your text value explicitly, by typing in a value and selecting the, 'Text' option. This value can also contain other tokens that are replaced out when the command runs (see the section regarding tokens near the end of this manual).
値を入力して「テキスト」オプションを選択することにより、テキスト値を明示的に設定することを選択できます。この値には、コマンドの実行時に置き換えられる他のトークンを含めることもできます（このマニュアルの終わり近くにあるトークンに関するセクションを参照してください）。

You can also choose to set your text value to another text value variable. To do this, just select 'A variable' option and type the name of the target text value variable in the box provided. Note that the variable can be the same as the one you are setting (for use if you just want to use the text options without assigning to another variable first).
テキスト値を別のテキスト値変数に設定することもできます。これを行うには、「A変数」オプションを選択し、表示されたボックスにターゲットテキスト値変数の名前を入力します。変数は設定しているものと同じであることに注意してください（最初に別の変数に割り当てずにテキストオプションを使用する場合に使用します）。

To save the text variable value with the current profile, check the, 'Save value to profile' box. The value will be available even if the VoiceAttack application is closed and opened again. This is a simple way to save your text value variable to disk.
現在のプロファイルでテキスト変数値を保存するには、[プロファイルに値を保存]ボックスをオンにします。VoiceAttackアプリケーションを閉じて再度開いた場合でも、値は使用可能になります。これは、テキスト値変数をディスクに保存する簡単な方法です。

To retrieve the saved value, select the, 'Retrieve saved value' option.
保存された値を取得するには、「保存された値を取得する」オプションを選択します。

To set the text value variable to an unset state, select the, 'Clear value' option.
テキスト値変数を未設定状態に設定するには、「値をクリア」オプションを選択します。

If you want to get the value for your text value variable from a file or URL, select the, 'Value from file/URI' option. To browse for a file, click the button with the ellipsis ('…'). To get the value from a URL, just type the address. For example, you can try, 'http://www.voiceattack.com/test.htm' (without the quotes). VoiceAttack will attempt to get the text from the response. Note: VoiceAttack reads data from these sources using UTF-8 encoding.
ファイルまたはURLからテキスト値変数の値を取得する場合は、[ファイル/ URIからの値]オプションを選択します。ファイルを参照するには、省略記号（ '…'）が付いたボタンをクリックします。URLから値を取得するには、アドレスを入力するだけです。たとえば、「http://www.voiceattack.com/test.htm」（引用符なし）を試すことができます。VoiceAttackは、応答からテキストを取得しようとします。注：VoiceAttackは、UTF-8エンコードを使用してこれらのソースからデータを読み取ります。

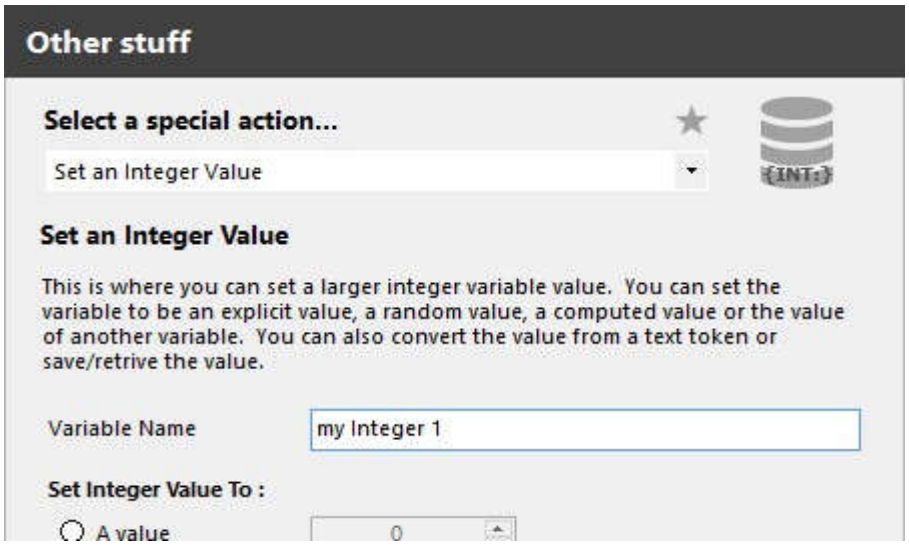
To access the values stored as text values, you will use the {TXT:valueName} token (see the section about tokens near the end of this manual).
テキスト値として保存された値にアクセスするには、{TXT：valueName}トークンを使用します（このマニュアルの最後にあるトークンに関するセクションを参照してください）。

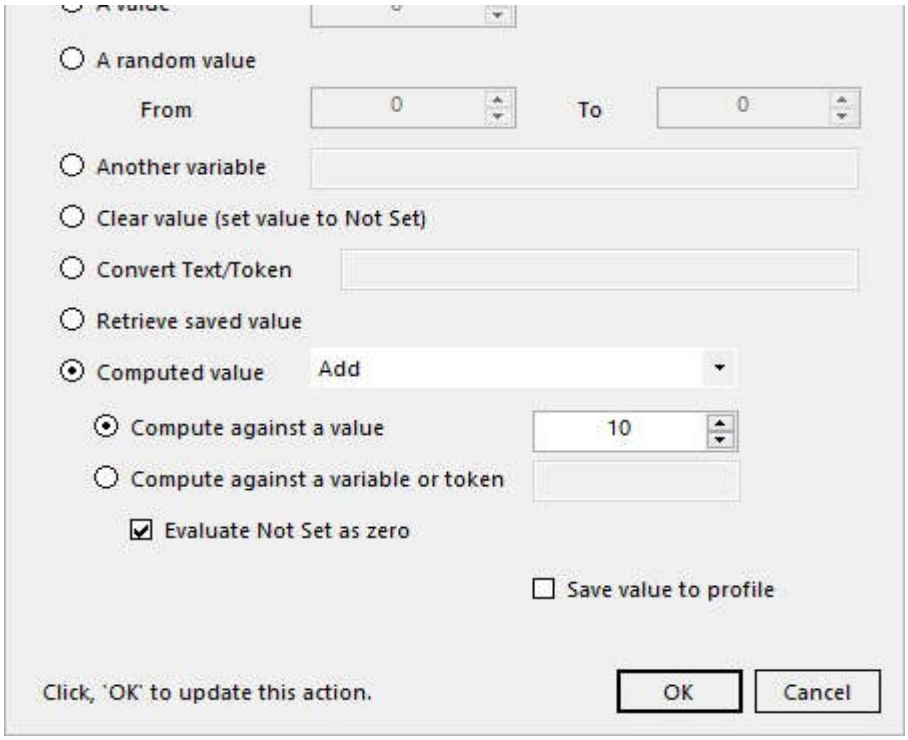
There are a few additional options available to you when you set a text variable. The options are Trim Spaces, Upper Case, Lower Case and Replace. These are applied after the variable is set. Trim Spaces will remove any spaces/whitespace from either end of the text value. If your text value is, ' Hello, how are you? ', using the Trim Spaces option will update the value to be, 'Hello, how are you?'. The Upper Case and Lower Case options will convert the text values to either all upper or all lower case characters. Using the Replace option will allow you to replace a portion of the text value with another value. For instance, you can replace, 'Hello' with 'Greetings' by putting 'Hello' in the first box and 'Greetings' in the second box. Any instance of the word, 'Hello' in the text value will be replaced with, 'Greetings'. Note that, 'Replace' is case-sensitive, and both of the 'Replace' input boxes can accept tokens.
テキスト変数を設定するときに使用できる追加のオプションがいくつかあります。オプションは、スペースのトリミング、大文字、小文字、および置換です。これらは、変数が設定された後に適用されます。[スペースのトリミング]は、テキスト値の両端からスペース/空白を削除します。テキスト値が 'Hello、お元気ですか？'、Trim Spacesオプションを使用すると、値が' Hello、how are you？ 'に更新されます。[大文字]オプションと[小文字]オプションは、テキスト値をすべて大文字またはすべて小文字に変換します。[置換]オプションを使用すると、テキスト値の一部を別の値に置き換えることができます。たとえば、最初のボックスに「Hello」、2番目のボックスに「Greetings」と入力すると、「Hello」を「Greetings」に置き換えることができます。「こんにちは」という単語のインスタンス テキストの値は「Greetings」に置き換えられます。「置換」では大文字と小文字が区別され、「置換」入力ボックスは両方ともトークンを受け入れることができます。

Note: You can define as many values as you want, however, the values that you define are not persisted. That is, they are not saved to disk (unless you check the, 'save values to profile' option). These values will be reset every time you restart VoiceAttack.
注：必要な数の値を定義できますが、定義した値は保持されません。つまり、それらはディスクに保存されません（「プロファイルに値を保存する」オプションをチェックしない限り）。これらの値は、VoiceAttackを再起動するたびにリセットされます。

Advanced: Variables can be scoped at the command-level, at the profile-level and globally. Most will use the globally-scoped variables (for good reason), however, for those that need a finer level of control, make sure to check out the, 'Advanced Variable Control (Scope)' section later in this document.
高度：変数は、コマンドレベル、プロファイルレベル、およびグローバルでスコープできます。大部分はグローバルなスコープの変数を使用します（正当な理由によります）が、より細かいレベルの制御が必要な場合は、このドキュメントで後述する「高度な変数制御（スコープ）」セクションを確認してください。

'Set an Integer Value'
「整数値を設定する」





This command action will allow you to set the value of an integer variable. These variables can be used in conjunction with Conditional blocks ('If' statements) to control the flow of your command actions, provide feedback through things like text-to-speech as well as do other stuff like provide information to plugins.
このコマンドアクションにより、整数変数の値を設定できます。これらの変数を条件ブロック(「If」ステートメント)と組み合わせて使用して、コマンドアクションのフローを制御したり、音声合成などのフィードバックを提供したり、プラグインに情報を提供するなどの他のことを実行したりできます。

The Integer Variable Name can be whatever name you want. This value will be stored at the application level (shared among all profiles), so care must be made in order to make your name is unique enough so you don't overwrite your values accidentally. The variable names are not case-sensitive and they must not contain semicolons or
整数変数名には、任意の名前を付けることができます。この値はアプリケーションレベル(すべてのプロファイルで共有)に保存されるため、名前が十分に一意になるように注意して、誤って値を上書きしないようにする必要があります。変数名は大文字と小文字が区別されず、セミコロンまたは

colons (variable names can only contain colons if contained within a token… this would be a good place to indicate that this input box also processes tokens).
コロン(変数名には、トークン内に含まれる場合にのみコロンを含めることができます。これは、この入力ボックスもトークンを処理することを示すのに適した場所です)。

The purpose of this screen is to set the value of the variable, and you do that by selecting one of several different ways. The first way is to set an exact value (such as 500). Just select the option labeled, 'A value' and type the value in the box.
この画面の目的は変数の値を設定することであり、いくつかの異なる方法のいずれかを選択することでそれを行います。最初の方法は、正確な値(500など)を設定することです。「A value」というラベルの付いたオプションを選択し、ボックスに値を入力するだけです。

Another way to set an integer value is to give it a random value. Just select, 'A random value' and provide a minimum and maximum value and the variable will have a random number chosen within that range.
整数値を設定する別の方法は、ランダムな値を与えることです。「ランダム値」を選択し、最小値と最大値を指定するだけで、変数はその範囲内で選択された乱数を持ちます。

You can set your variable to the same value as another variable. Select, 'Another variable' and type the name of the variable with the value that you want copied in the box provided.
変数を別の変数と同じ値に設定できます。[別の変数]を選択し、コピーする値を持つ変数の名前を表示されたボックスに入力します。

To clear the value of your variable (make the value be, 'Not Set' (programmers will call this, 'null'), select the, 'Clear value' option.
変数の値をクリアするには（値を「設定しない」にします（プログラマーはこれを「null」と呼びます）、「値のクリア」オプションを選択します。

If you have a value in text or in a token, you can attempt to convert the value to an integer by selecting, 'Convert Text/Token' and type the text and/or tokens into the box provided. If the value cannot be converted, the variable value will be, 'Not Set'. This is kind of advanced, and you may never even use this option.
テキストまたはトークンに値がある場合は、「テキスト/トークンの変換」を選択してテキストまたはトークンを入力ボックスに入力することにより、値を整数に変換することができます。値を変換できない場合、変数値は「未設定」になります。これは一種の高度な方法であり、このオプションを使用することはできません。

If you want your integer variable to be saved with the active profile, select the, 'Save value to profile' option (check box at the bottom). This will allow you to access the value between application sessions (VoiceAttack application is closed and then launched again).
整数変数をアクティブなプロファイルとともに保存する場合は、[プロファイルに値を保存する]オプションを選択します（下部のチェックボックス）。これにより、アプリケーションセッション間の値にアクセスできるようになります（VoiceAttackアプリケーションが閉じられ、再度起動されます）。

To retrieve the integer variable saved with the profile, select the, 'Retrieve saved value' option. If the value was previously saved (as indicated above), the value will be set. If no value is available, the variable value will be, 'Not Set'.
プロファイルで保存された整数変数を取得するには、「保存された値を取得」オプションを選択します。値が以前に保存されている場合（上記のとおり）、値が設定されます。値が使用できない場合、変数値は「未設定」になります。

Note: To clear all previously-saved integer variables, see the, 'Clear Saved Values from Profile' action.
注：以前に保存されたすべての整数変数をクリアするには、「プロファイルから保存された値をクリア」アクションを参照してください。

If you want your integer variable's value to be computed for you, there are some simple math functions available to you. First, select the, 'Computed value' option. Next, select the appropriate function. You can add, subtract, multiply and divide (with integer division… 7 divided by 3 is 2, for example), or get the remainder (Modulus)… 7 Mod 3 is 1, for example). The next thing you will want to do is indicate what you would like to compute against… that can be an explicit value (such as 2) or another variable or even a converted token. To select an explicit value, select, 'Compute against a value' and provide a value. To select another variable, choose, 'compute against a variable or token' and provide the variable name. To compute against a token, select this same option and indicate the token in the box provided. If the token cannot be converted, or the computed value falls outside of valid range (that is, between -2147483648 and 2147483647) the value of computation will be, 'Not Set'.
整数変数の値を計算する場合は、いくつかの簡単な数学関数を使用できます。最初に、「計算値」オプションを選択します。次に、適切な機能を選択します。加算、減算、乗算、除算(整数除算で…7を3で除算するなど)、または剰余を取得(モジュラス)... 7 Mod 3が1などです。次に行うことは、何に対して計算するかを示すことです。これは、明示的な値(2など)、別の変数、または変換されたトークンでもかまいません。明示的な値を選択するには、「値に対して計算」を選択して値を指定します。別の変数を選択するには、「変数またはトークンに対して計算」を選択し、変数名を指定します。トークンに対して計算するには、この同じオプションを選択し、提供されたボックスにトークンを示します。

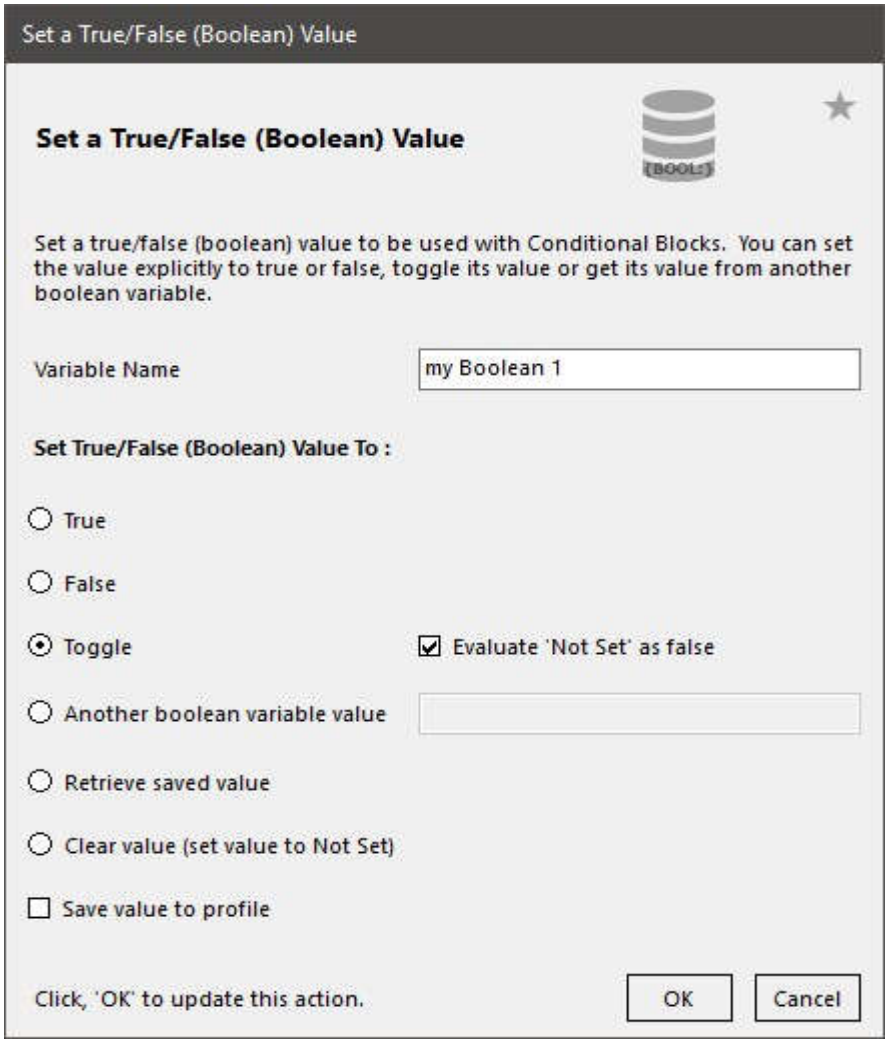
As a convenience feature, there is a check box labeled, 'Evaluate Not Set as zero'. This will allow you to initialize your variables as zero if they are not set when computing values. This is merely to save an initialization step (yes, another advanced bit you may
便利な機能として、「ゼロに設定しないで評価する」というラベルの付いたチェックボックスがあります。これにより、値を計算するときに変数が設定されていない場合、変数をゼロとして初期化できます。これは単に初期化手順を保存するためです(はい、もう1つの高度なビットを

never use).
使用しないでください)。

Note: You can define as many values as you want, however, the values that you define are not persisted. That is, they are not saved to disk. These values will be reset every time you restart VoiceAttack. If you want your values saved to disk for use between application sessions, select the 'Save value to profile' option.
注:必要な数の値を定義できますが、定義した値は保持されません。つまり、ディスクには保存されません。これらの値は、VoiceAttackを再起動するたびにリセットされます。アプリケーションセッション間で使用するために値をディスクに保存する場合は、[プロファイルに値を保存する]オプションを選択します。

Advanced: Variables can be scoped at the command-level, at the profile-level and globally. Most will use the globally-scoped variables (for good reason), however, for those that need a finer level of control, make sure to check out the, 'Advanced Variable Control (Scope)' section later in this document.
高度:変数は、コマンドレベル、プロファイルレベル、およびグローバルでスコープできます。大部分はグローバルなスコープの変数を使用します(正当な理由によります)が、より細かいレベルの制御が必要な場合は、このドキュメントで後述する「高度な変数制御(スコープ)」セクションを確認してください。

'Set a True/False (Boolean) Value'
「真/偽(ブール)値を設定する」



This command action will allow you to set the value of True/False (Boolean) variable. These variables can be used in conjunction with Conditional blocks ('If' statements) to control the flow of your command actions, provide feedback through things like text-to-
このコマンドアクションでは、True / False(ブール)変数の値を設定できます。これらの変数を条件ブロック(「If」ステートメント)と組み合わせて使用して、コマンドアクションのフローを制御し、テキストから

speech as well as do other stuff like provide information to plugins.
音声だけでなく、プラグインに情報を提供するような他のことを行います。

The Variable Name can be whatever name you want. This value will be stored at the application level (shared among all profiles), so care must be made in order to make your name is unique enough so you don't overwrite your values accidentally. The variable names are not case-sensitive and they must not contain semicolons or
変数名には任意の名前を指定できます。この値はアプリケーションレベル(すべてのプロファイルで共有)に保存されるため、名前が十分に一意になるように注意して、誤って値を上書きしないようにする必要があります。変数名は大文字と小文字が区別されず、セミコロンまたは

colons (variable names can only contain colons if contained within a token… this would be a good place to indicate that this input box also processes tokens).
コロン（変数名には、トークン内に含まれる場合にのみコロンを含めることができます。これは、この入力ボックスもトークンを処理することを示すのに適した場所です）。

The purpose of this screen is to set the value of the variable, and you do that by selecting one of several different ways. The first way is to set an explicit value (such as either True or False). Just select the option labeled, 'True' or the option labeled, 'False' to set your variable accordingly.
この画面の目的は変数の値を設定することであり、いくつかの異なる方法のいずれかを選択することでそれを行います。最初の方法は、明示的な値（TrueまたはFalseなど）を設定することです。「True」というラベルのオプションまたは「False」というラベルのオプションを選択するだけで、それに応じて変数を設定できます。

Another way to set a Boolean variable is to toggle its value. So, if a variable's value is True, it will be set to False. If it is False, it will be set to True.
ブール変数を設定する別の方法は、その値を切り替えることです。したがって、変数の値がTrueの場合、Falseに設定されます。Falseの場合、Trueに設定されます。

Note: A Boolean variable that is, 'Not Set' (programmers call this, 'null') will not toggle. The value will remain, 'Not Set'. You can set the, 'Evaluate 'Not Set' as false' option to treat null (Not Set) variable values as false. This may save a few steps here and there.
注：ブール変数である「Not Set」（プログラマーはこれを「null」と呼びます）は切り替わりません。値は「未設定」のままです。あなたが設定することができ、「『偽として設定されていない評価』（設定されていません）はnullを治療するために、オプション」偽として変数の値を。これにより、あちこちでいくつかの手順を節約できます。

You can set your variable to the same value as another variable. Select, 'Another Boolean variable value' and type the name of the variable with the value that you want copied in the box provided.
変数を別の変数と同じ値に設定できます。[別のブール変数値]を選択し、コピーする値を持つ変数の名前を表示されたボックスに入力します。

To clear the value of your variable (make the value be, 'Not Set' (programmers will call this, 'null'), select the, 'Clear value' option.
変数の値をクリアするには（値を「設定しない」にします（プログラマーはこれを「null」と呼びます）、「値のクリア」オプションを選択します。

If you want your Boolean variable to be saved with the active profile, select the, 'Save value to profile' option (check box at the bottom). This will allow you to access the value between application sessions (VoiceAttack application is closed and then launched again).
ブール変数をアクティブなプロファイルと共に保存する場合は、「プロファイルに値を保存する」オプションを選択します（下部のチェックボックス）。これにより、アプリケーションセッション間の値にアクセスできるようになります（VoiceAttackアプリケーションが閉じられ、再度起動されます）。

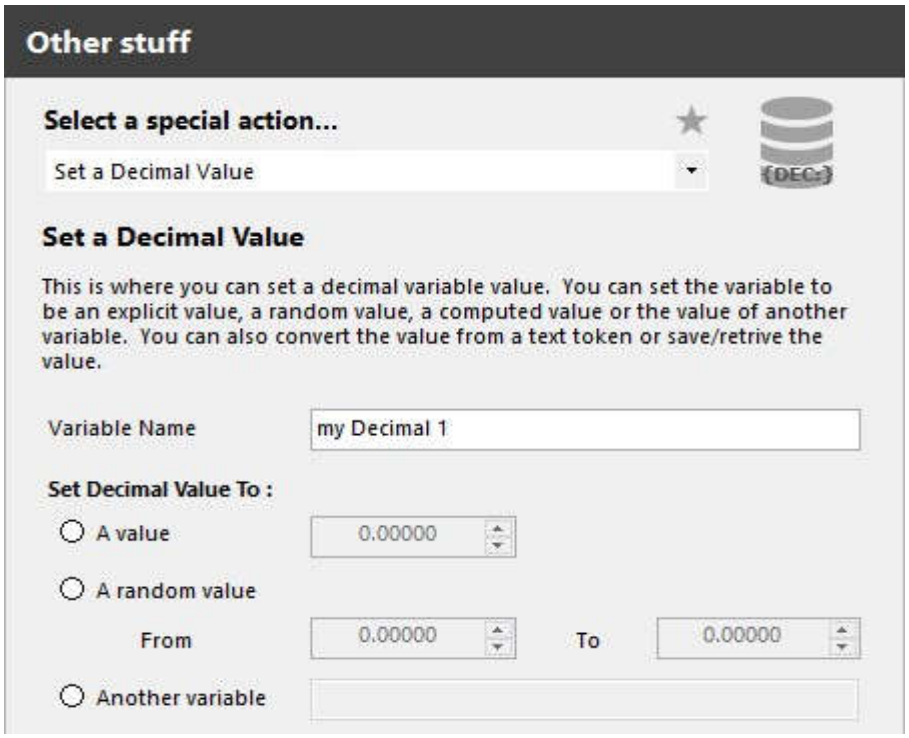
To retrieve the Boolean variable saved with the profile, select the, 'Retrieve saved value' option. If the value was previously saved (as indicated above), the value will be set. If no value is available, the variable value will be, 'Not Set'.
プロファイルとともに保存されたブール変数を取得するには、「保存された値を取得」オプションを選択します。値が以前に保存されている場合（上記のとおり）、値が設定されます。値が使用できない場合、変数値は「未設定」になります。

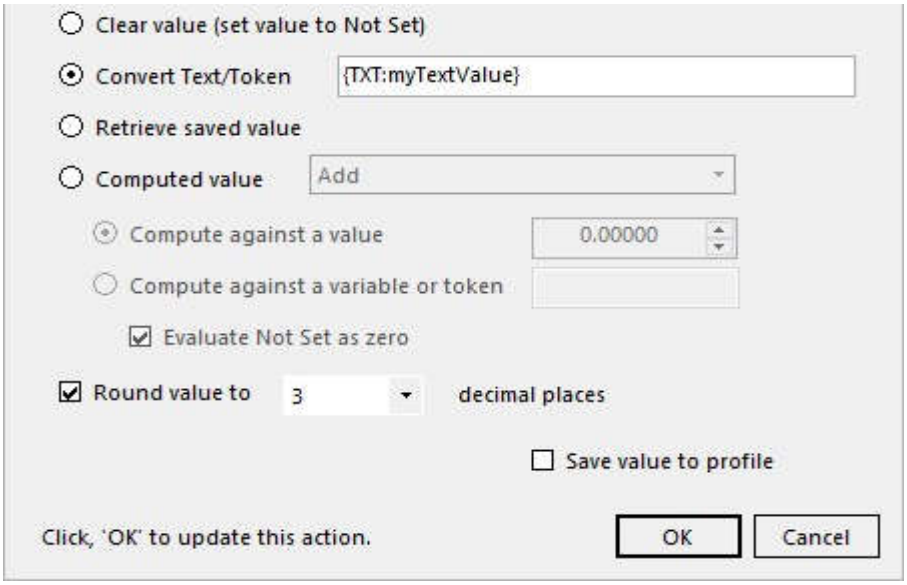
Note: To clear all previously-saved Boolean variables, see the, 'Clear Saved Values from Profile' action.
注：以前に保存されたすべてのブール変数をクリアするには、「プロファイルから保存された値をクリア」アクションを参照してください。

Note: You can define as many values as you want, however, the values that you define are not persisted. That is, they are not saved to disk. These values will be reset every time you restart VoiceAttack. If you want your values saved to disk for use between application sessions, select the 'Save value to profile' option.
注：必要な数の値を定義できますが、定義した値は保持されません。つまり、ディスクには保存されません。これらの値は、VoiceAttackを再起動するたびにリセットされます。アプリケーションセッション間で使用するために値をディスクに保存する場合は、[プロファイルに値を保存する]オプションを選択します。

Advanced: Variables can be scoped at the command-level, at the profile-level and globally. Most will use the globally-scoped variables (for good reason), however, for those that need a finer level of control, make sure to check out the, 'Advanced Variable Control (Scope)' section later in this document.
高度：変数は、コマンドレベル、プロファイルレベル、およびグローバルでスコープできます。大部分はグローバルなスコープの変数を使用します（正当な理由によります）が、より細かいレベルの制御が必要な場合は、このドキュメントで後述する「高度な変数制御（スコープ）」セクションを確認してください。

'Set a Decimal Value'
「10進値を設定する」





This command action will allow you to set the value of a decimal variable. These variables can be used in conjunction with Conditional blocks ('If' statements) to control the flow of your command actions, provide feedback through things like text-to-speech as well as do other stuff like provide information to plugins.
このコマンドアクションを使用すると、10進変数の値を設定できます。これらの変数を条件ブロック(「If」ステートメント)と組み合わせて使用して、コマンドアクションのフローを制御したり、音声合成などのフィードバックを提供したり、プラグインに情報を提供するなどの他のことを実行したりできます。

The Variable Name can be whatever name you want. The variable names are not case-sensitive and they must not contain semicolons or colons (variable names can only contain colons if contained within a token… this would be a good place to indicate that this input box also processes tokens).
変数名には任意の名前を指定できます。変数名は大文字と小文字を区別せず、セミコロンまたはコロンを含めることはできません(変数名には、トークンに含まれる場合にのみコロンを含めることができます。これは、この入力ボックスがトークンも処理することを示す適切な場所です)。

The purpose of this screen is to set the value of the variable, and you do that by selecting one of several different ways. The first way is to set an exact value (such as 100.1). Just select the option labeled, 'A value' and type the value in the box.
この画面の目的は変数の値を設定することであり、いくつかの異なる方法のいずれかを選択することでそれを行います。最初の方法は、正確な値(100.1など)を設定することです。「A value」というラベルの付いたオプションを選択し、ボックスに値を入力するだけです。

Another way to set a decimal variable value is to give it a random value. Just select, 'A random value' and provide a minimum and maximum value and the variable will have a random number chosen within that range.
10進数の変数値を設定する別の方法は、ランダムな値を与えることです。「ランダム値」を選択し、最小値と最大値を指定するだけで、変数はその範囲内で選択された乱数を持ちます。

You can set your variable to the same value as another variable. Select, 'Another variable' and type the name of the variable with the value that you want copied in the box provided.
変数を別の変数と同じ値に設定できます。[別の変数]を選択し、コピーする値を持つ変数の名前を表示されたボックスに入力します。

To clear the value of your variable (make the value be, 'Not Set' (programmers will call this, 'null'), select the, 'Clear value' option.
変数の値をクリアするには（値を「設定しない」にします（プログラマーはこれを「null」と呼びます）、「値のクリア」オプションを選択します。

If you have a value in text or in a token, you can attempt to convert the value to a decimal by selecting, 'Convert Text/Token' and type the text and/or tokens into the box provided. If the value cannot be converted, the variable value will be, 'Not Set'. This is kind of advanced, and you may never even use this option.
テキストまたはトークンに値がある場合は、「テキスト/トークンの変換」を選択してテキストまたはトークンを入力ボックスに入力することにより、値を10進数に変換することができます。値を変換できない場合、変数値は「未設定」になります。これは一種の高度な方法であり、このオプションを使用することはできません。

If you want your decimal variable to be saved with the active profile, select the, 'Save value to profile' option (check box at the bottom). This will allow you to access the value between application sessions (VoiceAttack application is closed and then launched again).
10進変数をアクティブなプロファイルとともに保存する場合は、[プロファイルに値を保存する]オプションを選択します（下部のチェックボックス）。これにより、アプリケーションセッション間の値にアクセスできるようになります（VoiceAttackアプリケーションが閉じられ、再度起動されます）。

To retrieve the decimal variable saved with the profile, select the, 'Retrieve saved value' option. If the value was previously saved (as indicated above), the value will be set. If no value is available, the variable value will be, 'Not Set'.
プロファイルで保存された10進変数を取得するには、「保存された値を取得」オプションを選択します。値が以前に保存されている場合（上記のとおり）、値が設定されます。値が使用できない場合、変数値は「未設定」になります。

Note: To clear all previously-saved decimal variables, see the, 'Clear Saved Values from Profile' action.
注：以前に保存された10進変数をすべてクリアするには、「プロファイルから保存された値をクリア」アクションを参照してください。

If you want your decimal variable's value to be computed for you, there are some simple math functions available to you. First, select the, 'Computed value' option. Next, select the appropriate function. You can add, subtract, multiply and divide (plus several more functions). The next thing you will want to do is indicate what you would like to compute against… that can be an explicit value (such as 2.6) or another variable or even a converted token. To select an explicit value, select, 'Compute against a value' and provide a value. To select another variable, choose, 'compute against a variable or token' and provide the variable name. To compute against a token, select this same option and indicate the token in the box provided. If the token cannot be converted, or the computed value falls outside of the acceptable range of values for a decimal (-79228162514264337593543950335 to 79228162514264337593543950335 (lol)) , the value of computation will be, 'Not Set'.
10進変数の値を計算したい場合は、いくつかの簡単な数学関数を使用できます。最初に、「計算値」オプションを選択します。次に、適切な機能を選択します。加算、減算、乗算、除算（さらにいくつかの関数）ができます。次に行うことは、計算対象を指定することです。これは、明示的な値（2.6など）、別の変数、または変換されたトークンでもかまいません。明示的な値を選択するには、「値に対して計算」を選択して値を指定します。別の変数を選択するには、「変数またはトークンに対して計算」を選択し、変数名を指定します。トークンに対して計算するには、この同じオプションを選択し、提供されたボックスにトークンを示します。トークンを変換できない場合、

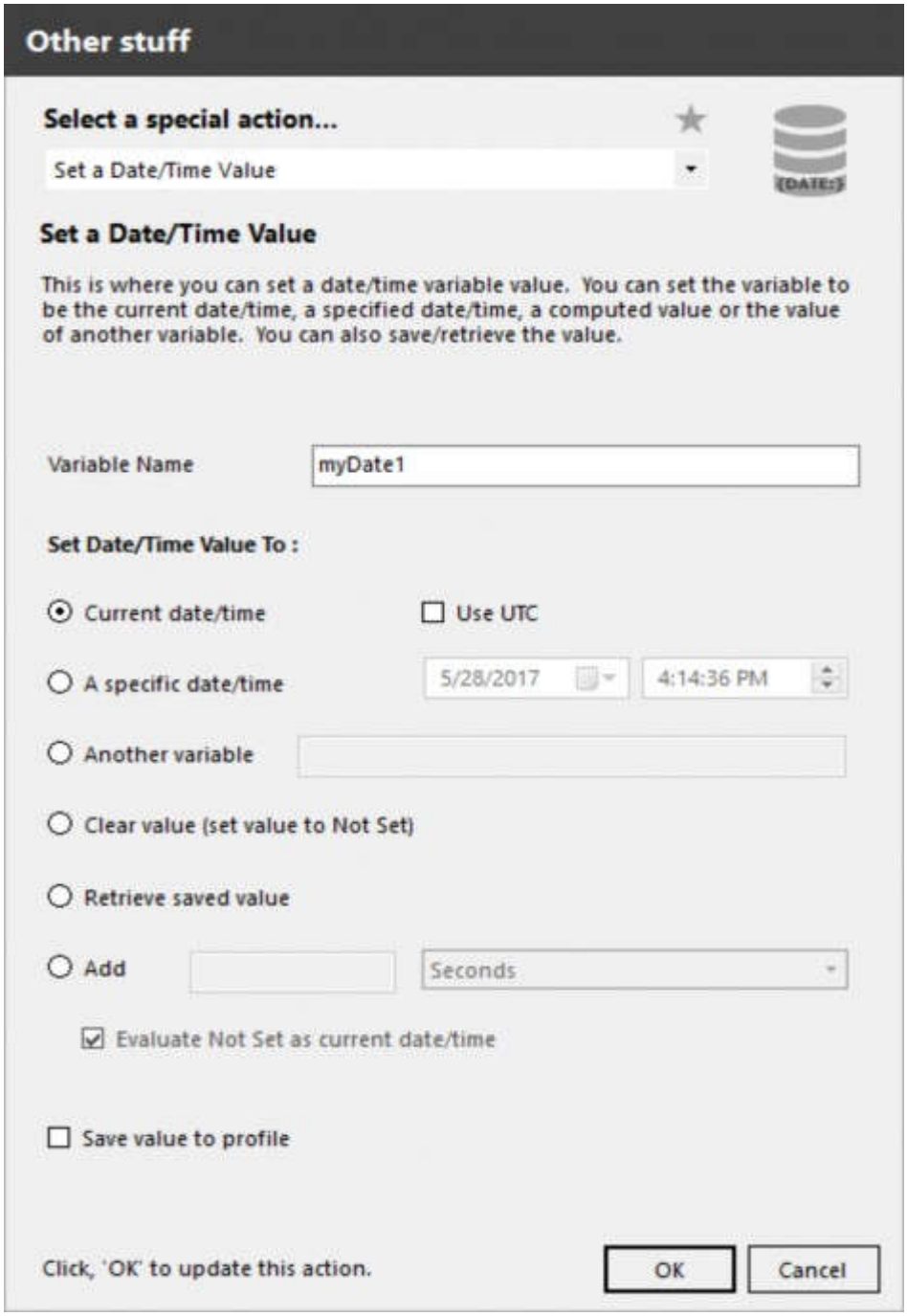
As a convenience feature, there is a check box labeled, 'Evaluate Not Set as zero'. This will allow you to initialize your variables as zero if they are not set when computing values. This is merely to save an initialization step (yes, another advanced bit you may never use).
便利な機能として、「ゼロに設定しないで評価する」というラベルの付いたチェックボックスがあります。これにより、値を計算するときに変数が設定されていない場合、変数をゼロとして初期化できます。これは、単に初期化ステップを保存するためのものです（はい、使用できないもう1つの高度なビット）。

If you would like your assigned variable to be rounded to a certain number of decimal places (from 0 up to 10), select the 'Round value' option and choose the appropriate value.
割り当てられた変数を特定の小数点以下の桁数（0から10まで）に丸めたい場合は、「Round value」オプションを選択し、適切な値を選択します。

Note: You can define as many values as you want, however, the values that you define are not persisted. That is, they are not saved to disk. These values will be reset every time you restart VoiceAttack. If you want your values saved to disk for use between application sessions, select the 'Save value to profile' option.
注：必要な数の値を定義できますが、定義した値は保持されません。つまり、ディスクには保存されません。これらの値は、VoiceAttackを再起動するたびにリセットされます。アプリケーションセッション間で使用するために値をディスクに保存する場合は、[プロファイルに値を保存する]オプションを選択します。

Advanced: Variables can be scoped at the command-level, at the profile-level and globally. Most will use the globally-scoped variables (for ease of use), however, for those that need a finer level of control, make sure to check out the, 'Advanced Variable Control (Scope)' section later in this document.
高度：変数は、コマンドレベル、プロファイルレベル、およびグローバルでスコープできます。ほとんどはグローバルスコープの変数を使用します（使いやすさのため）が、より細かいレベルの制御が必要な場合は、このドキュメントで後述する「高度な変数制御（スコープ）」セクションを確認してください。

'Set a Date/Time Value'
「日付/時刻値の設定」



This command action will allow you to set the value of a date/time variable. These variables can be used in conjunction with Conditional blocks ('If' statements) to control the flow of your command actions, provide feedback through things like text-to-speech as well as do other stuff like provide information to plugins.
このコマンドアクションを使用すると、日付/時刻変数の値を設定できます。これらの変数を条件ブロック(「If」ステートメント)と組み合わせて使用して、コマンドアクションのフローを制御したり、音声合成などのフィードバックを提供したり、プラグインに情報を提供するなどの他のことを実行したりできます。

The Variable Name can be whatever name you want. This value will be stored at the application level (shared among all profiles), so care must be made in order to make
変数名には任意の名前を指定できます。この値はアプリケーションレベル（すべてのプロファイルで共有）に保存されるため、作成するには注意が必要です。

your name is unique enough so you don't overwrite your values accidentally. The variable names are not case-sensitive and they must not contain semicolons or
名前は十分に一意であるため、誤って値を上書きすることはありません。変数名は大文字と小文字が区別されず、セミコロンまたは

colons (variable names can only contain colons if contained within a token… this would be a good place to indicate that this input box also processes tokens).
コロン（変数名には、トークン内に含まれる場合にのみコロンを含めることができます。これは、この入力ボックスもトークンを処理することを示すのに適した場所です）。

The purpose of this screen is to set the value of the variable, and you do that by selecting one of several different ways. The first way is to set the date/time variable to the current date/time (the current date/time when the action is executed). Just select the option, 'Current date/time'. If you want to set this value to UTC, select the, 'Use UTC' option.
この画面の目的は変数の値を設定することであり、いくつかの異なる方法のいずれかを選択することでそれを行います。最初の方法は、日付/時刻変数を現在の日付/時刻（アクションが実行される現在の日付/時刻）に設定することです。[現在の日付/時刻]オプションを選択するだけです。この値をUTCに設定する場合は、「UTCを使用」オプションを選択します。

Another way is to set an exact date and time. You can do this by selecting the, 'Specific date/time' option and selecting the date and time in the boxes provided.
別の方法は、正確な日付と時刻を設定することです。これを行うには、[特定の日付/時刻]オプションを選択し、表示されたボックスで日付と時刻を選択します。

You can set your variable to the same value as another variable. Select, 'Another variable' and type the name of the variable with the value that you want copied in the box provided.
変数を別の変数と同じ値に設定できます。[別の変数]を選択し、コピーする値を持つ変数の名前を表示されたボックスに入力します。

To clear the value of your variable (make the value be, 'Not Set' (programmers will call this, 'null'), select the, 'Clear value' option.
変数の値をクリアするには（値を「設定しない」にします（プログラマーはこれを「null」と呼びます）、「値のクリア」オプションを選択します。

If you want your date/time variable to be saved with the active profile, select the, 'Save value to profile' option (check box at the bottom). This will allow you to access the value between application sessions (VoiceAttack application is closed and then launched again).
日付/時刻変数をアクティブなプロファイルとともに保存する場合は、[プロファイルに値を保存する]オプションを選択します（下部のチェックボックス）。これにより、アプリケーションセッション間の値にアクセスできるようになります（VoiceAttackアプリケーションが閉じられ、再度起動されます）。

To retrieve the date/time variable saved with the profile, select the, 'Retrieve saved value' option. If the value was previously saved (as indicated above), the value will be set. If no value is available, the variable value will be, 'Not Set'.
プロファイルで保存された日付/時刻変数を取得するには、「保存された値を取得」オプションを選択します。値が以前に保存されている場合（上記のとおり）、値が設定されます。値が使用できない場合、変数値は「未設定」になります。

Note: To clear all previously-saved date/time variables, see the, 'Clear Saved Values from Profile' action.
注：以前に保存されたすべての日付/時刻変数をクリアするには、「プロファイルから保存された値をクリア」アクションを参照してください。

If you want add or subtract time from your date/time variable, there are some simple options available. Select the, 'Add' option and then indicate the number of seconds, milliseconds, minutes, hours, days, months or years to add to the variable. To subtract time from your date/time variable, simply use negative values in the input box. Note that this input box can contain a large integer variable, as well as any combination of tokens that may resolve into an integer. If your date/time variable is, 'Not Set', adding or subtracting time from it will still result in, 'Not Set'. As a convenience, the, 'Evaluate Not Set as current date/time' option is available. If this option is selected and the variable is new or cleared (Not Set), the variable will initialize as the current date/time (might save a step).
日付/時刻変数から時間を追加または削除する場合は、いくつかの簡単なオプションを使用できます。「追加」オプションを選択し、変数に追加する秒数、ミリ秒数、分数、時間数、日数、月数または年数を指定します。日付/時間変数から時間を減算するには、入力ボックスで負の値を使用します。この入力ボックスには、整数に解決できるトークンの任意の組み合わせだけでなく、大きな整数変数を含めることができることに注意してください。日付/時刻変数が「未設定」の場合、そこに時間を加算または減算しても「未設定」になります。便宜上、「現在の日付/時刻として設定しないを評価する」オプションを使用できます。このオプションが選択され、変数が新規またはクリア（未設定）の場合、変数は現在の日付/時刻として初期化されます（ステップを保存する場合があります）。

Note: You can define as many values as you want, however, the values that you define are not persisted. That is, they are not saved to disk. These values will be reset every time you restart VoiceAttack. If you want your values saved to disk for use between application sessions, select the 'Save value to profile' option.
注：必要な数の値を定義できますが、定義した値は保持されません。つまり、ディスクには保存されません。これらの値は、VoiceAttackを再起動するたびにリセットされます。アプリケーションセッション間で使用するために値をディスクに保存する場合は、[プロファイルに値を保存する]オプションを選択します。

Advanced: Variables can be scoped at the command-level, at the profile-level and
高度：変数は、コマンドレベル、プロファイルレベル、

globally. Most will use the globally-scoped variables (for good reason), however, for those that need a finer level of control, make sure to check out the, ‘Advanced Variable Control (Scope)’ section later in this document.
グローバルに。大部分はグローバルなスコープの変数を使用します（正当な理由によります）が、より細かいレベルの制御が必要な場合は、このドキュメントで後述する「高度な変数制御（スコープ）」セクションを確認してください。

'Convert a Value'
「値を変換する」

This action will allow you to convert the value from one variable type to another. For instance, if you have a text variable with a value of “1234” and you want that text to be converted to an integer value, you can select what integer variable in which to place the value of 1234 into. The available types to convert are the six standard types: text, small integers, decimals, integers, true/false (Boolean) and date/time.
このアクションにより、ある変数タイプから別の変数タイプに値を変換できます。たとえば、値が「1234」のテキスト変数があり、そのテキストを整数値に変換する場合、1234の値を配置する整数変数を選択できます。変換できるタイプは、テキスト、短整数、小数、整数、真/偽（ブール）、日付/時刻の6つの標準タイプです。

Simply put the source and target variable names into the source variable and target variable boxes and select their types. Note that the source and target variable boxes can process tokens to resolve variable names.
ソース変数とターゲット変数の名前をソース変数とターゲット変数のボックスに入れて、タイプを選択するだけです。ソース変数ボックスとターゲット変数ボックスは、トークンを処理して変数名を解決できることに注意してください。

Note – Any failed attempt to convert a value will result in the target variable having its value removed (not set).
注–値の変換に失敗すると、ターゲット変数の値が削除されます（設定されません）。

· Converting text to any other type will attempt to parse the text into the target type, EXCEPT Boolean, which will also accept “0” as false and “1” as true in addition to “true” and “false”.
・ テキストを他のタイプに変換すると、テキストを解析してターゲットタイプEXCEPT Booleanに変換しようとします。これは、「true」と「false」に加えて「0」をfalse、「1」をtrueとしても受け入れます。

· Converting a small integer, decimal or small integer into a Boolean will normally fail EXCEPT if the value is 0 or 1. If the value is 0, the converted Boolean value will be false. If the value is 1, the Boolean value will be true.
・ 小整数、小数、または小整数をブール値に変換すると、値が0または1の場合を除き、通常は失敗します。値が0の場合、変換されたブール値はfalseになります。値が1の場合、ブール値はtrueになります。

· The only type that can be converted to a date/time is text (or another date/time).
・ 日付/時刻に変換できる唯一のタイプはテキスト（または別の日付/時刻）です。

· Converting from a Boolean to text will yield the text value for the Boolean (and not “0” or “1”).
・ ブール値からテキストに変換すると、ブール値のテキスト値が生成されます（「0」または「1」ではありません）。

‘Get User Input’
「ユーザー入力を取得」

An inline function or plugin makes sense when it comes to presenting a dialog to the user to get input. Also, a number of specific commands may be required to get a proper spoken response from a user. Sometimes you just want something basic to interact with the user, and that's what the, 'Get User Input' set of features will do. The, 'Get User Input' screens consist of spoken response, text, integer, decimal and pick list (choice).
インライン関数またはプラグインは、入力を取得するためにユーザーにダイアログを表示する場合に意味があります。また、ユーザーから適切な音声応答を得るには、いくつかの特定のコマンドが必要になる場合があります。ユーザーとやり取りするための基本的なものが必要な場合があります。それが、「ユーザー入力の取得」機能セットの機能です。「ユーザー入力の取得」画面は、音声応答、テキスト、整数、10進数、および選択リスト(選択)で構成されています。

'Get User Input - Wait for Spoken Response' - This action will make the executing command wait until the user has spoken a response from a specific set of responses. The response is then placed in a text variable that can be examined with a conditional statement, so that the command can be instructed to do certain things. This action also provides a timeout that you can specify, so that if the response takes too long, the command can continue. There are two items that are required for this action to work.
「ユーザー入力の取得-音声応答を待つ」-このアクションは、ユーザーが特定の応答セットから応答を音声するまで、実行中のコマンドを待機させます。その後、応答は条件付きステートメントで調べることができるテキスト変数に配置されるため、コマンドに特定のことを行うように指示できます。このアクションでは、指定可能なタイムアウトも提供されるため、応答に時間がかかりすぎる場合にコマンドを続行できます。このアクションが機能するために必要な2つの項目があります。

First, you must provide some type of response(s) in the, 'Responses' input box. This can be as simple as one word, ('fire'), or, it can be a set of several words or phrases separated by semicolons ('fire;fire weapons;fire at will'). Also, this input box will process dynamic phrases ('fire[the;all;][weapons;lasers;squirrels]') (see, 'Dynamic command sections' earlier in this document for help on that). This box will also render any combination of tokens to help broaden your response possibilities. Note that the maximum number of expected responses is limited to 250 items, and if more than 250 items are rendered via token, only the first 250 will be used (as you can tell, this is an advanced feature).
最初に、「応答」入力ボックスに何らかのタイプの応答を指定する必要があります。これは、1つの単語(「発火」)のように単純な場合もあれば、セミコロンで区切られた複数の単語またはフレーズのセット(「発火;武器を発射;意のままに発火」)の場合もあります。また、この入力ボックスは、動的フレーズ('fire [the; all;] [weapons; lasers; squirrels]')を処理します(これに関するヘルプについては、このドキュメントの「動的コマンドセクション」を参照してください)。また、このボックスは、トークンの組み合わせをレンダリングして、応答の可能性を広げます。予想される応答の最大数は250アイテムに制限され、250を超えるアイテムがトークン経由でレンダリングされる場合、最初の250のみが使用されることに注意してください(これは高度な機能です)。

Second, you must indicate a text variable name in the, 'Text Variable' input box to hold the user's spoken response. Note that this box also will resolve tokens into variable names. Optionally, you can specify a value in the, 'Timeout' input box. The timeout value indicates how long this action is to wait (in seconds) for a proper response before giving up and continuing. A value of zero indicates no timeout, which means that the action will wait for a response indefinitely. If the timeout period expires, the action will continue and the value placed in the indicated text variable will be, Not Set. If you would like the command to continue no matter what the user says, check the, 'Continue on any Speech' checkbox. If the user says something that is not in your provided response list, control will move from the action, but the value placed the text variable will be, '@@invalid' (without the quotes).
次に、ユーザーの音声応答を保持するために、「テキスト変数」入力ボックスにテキスト変数名を指定する必要があります。このボックスは、トークンを変数名に解決することにも注意してください。オプションで、「タイムアウト」入力ボックスに値を指定できます。タイムアウト値は、このアクションがproperめて続行する前に適切な応答を待機する時間(秒単位)を示します。値ゼロはタイムアウトがないことを示します。つまり、アクションは無期限に応答を待機します。タイムアウト期間が終了すると、アクションが続行され、指定されたテキスト変数に配置される値は「未設定」になります。ユーザーの発言に関係なくコマンドを続行したい場合は、「音声で続行」をチェックしてください。チェックボックス。ユーザーが指定された応答リストにないものを言うと、コントロールはアクションから移動しますが、テキスト変数に配置された値は「@@invalid」(引用符なし)になります。

Note: Again, if the user replies with a proper response, the text value of their response will be placed in the indicated text variable. This response will already be lower case for easy comparison in your command.
注：繰り返しますが、ユーザーが適切な応答で応答すると、応答のテキスト値は指定されたテキスト変数に配置されます。コマンドで簡単に比較できるように、この応答は既に小文字になっています。

Note: The number of responses indicated in the, ‘Responses’ box must resolve to a maximum of 250 possible responses. This is an arbitrary limitation set to prevent the overloading of the speech engine.
注：[応答]ボックスに表示される応答の数は、最大250の可能な応答に解決される必要があります。これは、音声エンジンの過負荷を防ぐために設定された任意の制限です。

‘Get User Input – Text’ – When this action is executed, the user will be presented with a simple dialog box that allows the user to type in whatever they would like. The user can click, ‘OK’ or, ‘Cancel’ to submit or not submit their information. This action will require you to indicate a text variable name in the, ‘Storage Variable’ field. This is the text variable that will be populated with the result of whatever the user enters. If the user clicks, ‘OK’, the value in the variable will be what the user types in. If the user clicks, ‘Cancel’, the value in the variable will be, ‘Not Set’.
「ユーザー入力の取得–テキスト」–このアクションが実行されると、ユーザーは簡単なダイアログボックスが表示され、ユーザーが好きなように入力できます。ユーザーは、[OK]または[キャンセル]をクリックして、情報を送信するかどうかを選択できます。このアクションでは、「ストレージ変数」フィールドにテキスト変数名を指定する必要があります。これは、ユーザーが入力した結果を入力するテキスト変数です。ユーザーが「OK」をクリックすると、変数の値はユーザーが入力したものになります。ユーザーが「キャンセル」をクリックすると、変数の値は「未設定」になります。

The other values for this action are optional but are important for user presentation.
このアクションの他の値はオプションですが、ユーザーのプレゼンテーションにとって重要です。

The, ‘Window Title’ option allows you to specify what will appear in the top bar of the displayed dialog. This can be a text variable name that contains the text to display. This can also be a literal value like, ‘Enter Some Text’, or can contain any combination of literal text and tokens. NOTE: Since this option can contain either a variable or literal text, if the variable’s value results in, ‘Not Set’, the action assumes that the value entered is literal text. So, if you have a variable named, ‘myVariable’ and its value is not set, the window title will display, ‘myVariable’.
[ウィンドウタイトル]オプションを使用すると、表示されたダイアログのトップバーに表示される内容を指定できます。これは、表示するテキストを含むテキスト変数名にすることができます。これは、「Enter Text」などのリテラル値にすることも、リテラルテキストとトークンの任意の組み合わせにすることもできます。注： このオプションには変数またはリテラルテキストのいずれかを含めることができるため、変数の値が「未設定」になる場合、アクションは入力された値がリテラルテキストであると想定します。したがって、「myVariable」という名前の変数があり、その値が設定されていない場合、ウィンドウのタイトルは「myVariable」と表示されます。

The, ‘Prompt Text’ option lets you specify some instructions just above where the user
「プロンプトテキスト」オプションを使用すると、ユーザーが

will type their input. This can be a text variable or literal text/tokens just like, ‘Window Title’ above. Note: Tokens such as, ‘{NEWLINE}’ can be handy here.
入力を入力します。これは、上記の「ウィンドウタイトル」のように、テキスト変数またはリテラルテキスト/トークンにすることができます。注：「{NEWLINE}」などのトークンはここで便利です。

The, ‘Initial Value’ option lets you indicate a prepopulated value in the input box as a matter of convenience to the user. So, let’s say you are requesting info from the user, and the answer they are going to provide is most likely going to be, ‘Bananas’, you can indicate that here. ‘Bananas’ will appear in the input box and the user can choose to keep that value if they want to (and be spared from having to type it in). The same variable/text/token convention in the previous options apply here as well.
[初期値]オプションを使用すると、ユーザーの利便性を考慮して、入力ボックスに事前入力された値を指定できます。したがって、ユーザーに情報を要求しており、ユーザーが提供しようとしている答えが「バナナ」である可能性が最も高いとしましょう。ここでそれを示すことができます。「Bananas」が入力ボックスに表示され、ユーザーは必要に応じてその値を保持することを選択できます（入力する必要はありません）。ここでも、前のオプションと同じ変数/テキスト/トークンの規則が適用されます。

The, ‘Maximum Length’ option allows you to specify the maximum number of characters the user is going to type in. So, let’s say you are asking for some information that will only ever be 3 characters in length, you can specify that here. If you do not care about a maximum length, simply put a zero in this box.
「最大長」オプションを使用すると、ユーザーが入力する文字の最大数を指定できます。したがって、3文字の長さになる情報を要求しているとしましょう。ここで指定できます。最大長を気にしない場合は、このボックスにゼロを入力してください。

The, ‘Require Input’ option lets you require that the user type something in. The, ‘OK’ button will not be enabled for the user as long as the input box is empty.
[入力が必要]オプションを使用すると、ユーザーは何かを入力する必要があります。入力ボックスが空である限り、[OK]ボタンはユーザーに対して有効になりません。

The, ‘Stay on Top’ option indicates that the input screen should be the top-most form when it is displayed.
「上部に表示」オプションは、入力画面が表示されたときに一番上に表示されることを示します。

‘Get User Input - Choice’ - When this action is executed, the user will be presented with a simple dialog box that allows the user to pick an item from a drop-down list of items. The user can click, ‘OK’ or, ‘Cancel’ to submit or not submit their selection. This action will require you to indicate a text variable name in the, ‘Storage Variable’ field.
「ユーザー入力の取得-選択」-このアクションが実行されると、ユーザーはアイテムのドロップダウンリストからアイテムを選択できるシンプルなダイアログボックスが表示されます。ユーザーは、[OK]または[キャンセル]をクリックして、選択内容を送信するかどうかを選択できます。このアクションでは、「ストレージ変数」フィールドにテキスト変数名を指定する必要があります。

This is the text variable that will be populated with the result of whatever the user chooses. If the user clicks, ‘OK’, the value in the variable will be set to be the text of the item that the user selects. If the user clicks, ‘Cancel’, the value in the variable will be, ‘Not Set’.
これは、ユーザーが選択した結果が入力されるテキスト変数です。ユーザーが「OK」をクリックすると、変数の値はユーザーが選択したアイテムのテキストに設定されます。ユーザーが「キャンセル」をクリックすると、変数の値は「未設定」になります。

The other values for this action are optional but are important for user presentation.
このアクションの他の値はオプションですが、ユーザーのプレゼンテーションにとって重要です。

The, ‘Window Title’ option allows you to specify what will appear in the top bar of the displayed dialog. This can be a text variable name that contains the text to display. This can also be a literal value like, ‘Choose an Item’, or can contain any combination of literal text and tokens. Note: Since this option can contain either a variable or literal text, if the variable’s value results in, ‘Not Set’, the action assumes that the value entered is literal text. So, if you have a variable named, ‘myVariable’ and its value is not set, the window title will display, ‘myVariable’.
[ウィンドウタイトル]オプションを使用すると、表示されたダイアログのトップバーに表示される内容を指定できます。これは、表示するテキストを含むテキスト変数名にすることができます。これは、「アイテムの選択」のようなリテラル値にすることも、リテラルテキストとトークンの任意の組み合わせを含めることもできます。注：このオプションには変数またはリテラルテキストのいずれかを含めることができるため、変数の値が「未設定」になる場合、アクションは入力された値がリテラルテキストであると想定します。したがって、「myVariable」という名前の変数があり、その値が設定されていない場合、ウィンドウのタイトルは「myVariable」と表示されます。

The, ‘Prompt Text’ option lets you specify some instructions just above where the user will make their selection. This can be a text variable or literal text/tokens just like, ‘Window Title’ above. Note: Tokens such as, ‘{NEWLINE}’ can be handy here.
[プロンプトテキスト]オプションを使用すると、ユーザーが選択する場所のすぐ上にいくつかの指示を指定できます。これは、上記の「ウィンドウタイトル」のように、テキスト変数またはリテラルテキスト/トークンにすることができます。注：「{NEWLINE}」などのトークンはここで便利です。

The, ‘Values’ option, although technically optional, must be provided if you actually want your user to have a selection to choose from. In order to indicate multiple items for your user to select from, you must separate your items by semicolons (;). For example, if you want to provide a pick list consisting of Apples, Oranges, Bananas and
「値」オプションは、技術的にはオプションですが、実際にユーザーが選択できるようにする場合に提供する必要があります。ユーザーが選択する複数のアイテムを指定するには、アイテムをセミコロン(;)で区切る必要があります。たとえば、リンゴ、オレンジ、バナナ、

Pluto, the value entered into this box must look like this: Apples;Oranges;Bananas;Pluto. The user will be presented with a pick list of those four items to choose from. The value in this box can also be a text variable or literal text/tokens just like the previous options.
uto王星、このボックスに入力する値は、Apples; Oranges; Bananas; Plutoのようになります。ユーザーには、これらの4つの項目から選択する選択リストが表示されます。このボックスの値は、前述のオプションと同様に、テキスト変数またはリテラルテキスト/トークンにすることもできます。

The, ‘Selected Value’ option lets you indicate the selected item in the pick list when it is presented to the user. So, let’s say that you want the selected item to be, ‘Pluto’.
[選択された値]オプションを使用すると、ユーザーに表示される選択リストで選択されたアイテムを指定できます。したがって、選択したアイテムを「Pluto」にしたいとします。

Simply put, ‘Pluto’ in the box. When the pick list is displayed, ‘Pluto’ will be the item that is selected. Note: If the pick list does NOT contain the selected value, the selected value will be added to the pick list (and selected).
ボックスに「Pluto」と入力するだけです。選択リストが表示されると、「Pluto」が選択されたアイテムになります。注：選択リストに選択値が含まれていない場合、選択値は選択リストに追加されます（選択されます）。

The, ‘Stay on Top’ option indicates that the input screen should be the top-most form when it is displayed.
「上部に表示」オプションは、入力画面が表示されたときに一番上に表示されることを示します。

‘Get User Input – Integer’ – When this action is executed, the user will be presented with a simple dialog box that allows the user to type in an integer value. The user can click, ‘OK’ or, ‘Cancel’ to submit or not submit their information. This action will require you to indicate an integer variable name in the, ‘Storage Variable’ field. This is the integer variable that will be populated with the result of whatever the user enters. If the user clicks, ‘OK’, the value in the variable will be what the user types in. If the user clicks, ‘Cancel’, the value in the variable will be, ‘Not Set’.
「ユーザー入力の取得-整数」-このアクションを実行すると、整数値を入力できるシンプルなダイアログボックスが表示されます。ユーザーは、[OK]または[キャンセル]をクリックして、情報を送信するかどうかを選択できます。このアクションでは、「ストレージ変数」フィールドに整数変数名を指定する必要があります。これは、ユーザーが入力した結果を入力する整数変数です。ユーザーが「OK」をクリックすると、変数の値はユーザーが入力したものになります。ユーザーが「キャンセル」をクリックすると、変数の値は「未設定」になります。

The other values for this action are optional but are important for user presentation.
このアクションの他の値はオプションですが、ユーザーのプレゼンテーションにとって重要です。

The, ‘Window Title’ option allows you to specify what will appear in the top bar of the displayed dialog. This can be a text variable name that contains the text to display. This can also be a literal value like, ‘Enter a Number’, or can contain any combination of literal text and tokens. NOTE: Since this option can contain either a variable or literal text, if the variable’s value results in, ‘Not Set’, the action assumes that the value entered is literal text. So, if you have a variable named, ‘myVariable’ and its value is not set, the window title will display, ‘myVariable’.
[ウィンドウタイトル]オプションを使用すると、表示されたダイアログのトップバーに表示される内容を指定できます。これは、表示するテキストを含むテキスト変数名にすることができます。これは、「数値を入力」などのリテラル値にすることも、リテラルテキストとトークンの任意の組み合わせを含めることもできます。注： このオプションには変数またはリテラルテキストのいずれかを含めることができるため、変数の値が「未設定」になる場合、アクションは入力された値がリテラルテキストであると想定します。したがって、「myVariable」という名前の変数があり、その値が設定されていない場合、ウィンドウのタイトルは「myVariable」と表示されます。

The, ‘Prompt Text’ option lets you specify some instructions just above where the user will type their input. This can be a text variable or literal text/tokens just like, ‘Window Title’ above. Note: Tokens such as, ‘{NEWLINE}’ can be handy here.
「プロンプトテキスト」オプションを使用すると、ユーザーが入力を入力する場所のすぐ上にいくつかの指示を指定できます。これは、上記の「ウィンドウタイトル」のように、テキスト変数またはリテラルテキスト/トークンにすることができます。注：「{NEWLINE}」などのトークンはここで便利です。

The, ‘Initial Value’ option lets you indicate a prepopulated value in the input box as a matter of convenience to the user. So, let’s say you are requesting info from the user, and the answer they are going to provide is most likely going to be, ‘55’, you can indicate that here. ‘55’ will appear in the input box and the user can choose to keep that value if they want to (and be spared from having to type it in). The value of this box can be a literal value (such as 55), or it can be a variable name that resolves to an integer or any combination of tokens that resolves to an integer value. If an integer value cannot be resolved, 0 will be used (or, whatever the minimum value is set to (see below)).
[初期値]オプションを使用すると、ユーザーの利便性を考慮して、入力ボックスに事前入力された値を指定できます。したがって、ユーザーに情報を要求しているとしましょう。ユーザーが提供しようとしている答えは「55」である可能性が最も高く、ここでそれを示すことができます。入力ボックスに「55」が表示され、ユーザーは必要に応じてその値を保持することを選択できます（入力する必要はありません）。このボックスの値は、リテラル値（55など）にすることも、整数に解決される変数名、または整数値に解決されるトークンの任意の組み合わせにすることもできます。整数値を解決できない場合、0が使用されます（または、最小値が設定されているものは何でも（下記を参照））。

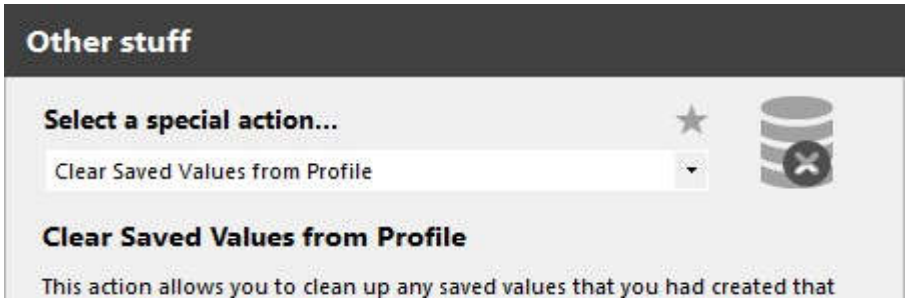
The, ‘Minimum’ and ‘Maximum’ options allow you to specify a range to constrain user input. So, if you know your user’s input is going to be between 1 and 10, you can put 1
「最小」および「最大」オプションを使用すると、ユーザー入力を制限する範囲を指定できます。そのため、ユーザーの入力が1～10の間であることがわかっている場合、1を入力できます。

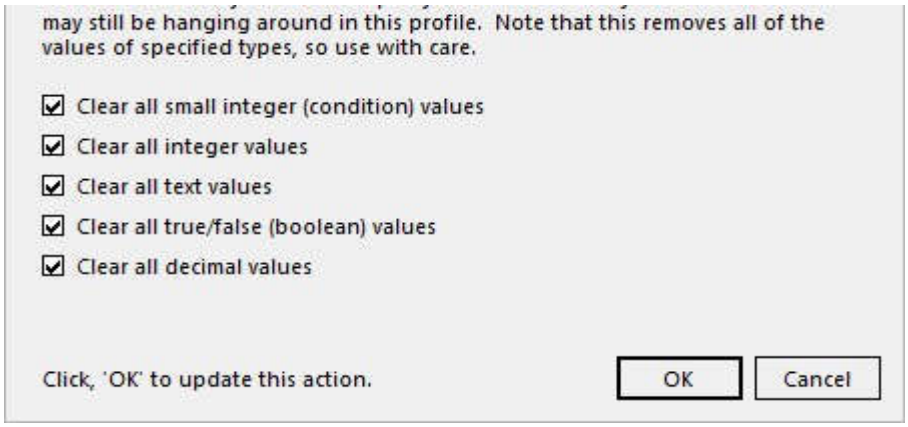
in the, ‘Minimum’ box and, 10 in the ‘Maximum’ box. The OK button will not be enabled unless the user inputs a value in the range you specify. If you do not want to specify a range for either, ‘Minimum’ or ‘Maximum’ just leave either box blank. Note that the same variable/token conditions apply here as they do for the, ‘Initial Value’ option above.
「最小」ボックス、および「最大」ボックスに10。ユーザーが指定した範囲の値を入力しない限り、[OK]ボタンは有効になりません。いずれかの範囲を指定しない場合は、「最小」または「最大」のいずれかのボックスを空白のままにします。上記の「初期値」オプションの場合と同じ変数/トークン条件がここで適用されることに注意してください。

The, ‘Stay on Top’ option indicates that the input screen should be the top-most form when it is displayed.
「上部に表示」オプションは、入力画面が表示されたときに一番上に表示されることを示します。

‘Get User Input - Decimal’ - This action behaves exactly the same as the ‘Get User Input - Integer’ action above, except that all the input is in decimal form and not integer.
「ユーザー入力の取得-10進数」- このアクションは、すべての入力が整数ではなく10進数形式であることを除いて、上記の「ユーザー入力の取得-整数」アクションとまったく同じ動作をします。

'Clear Saved Values from Profile'
「プロファイルから保存値を消去」

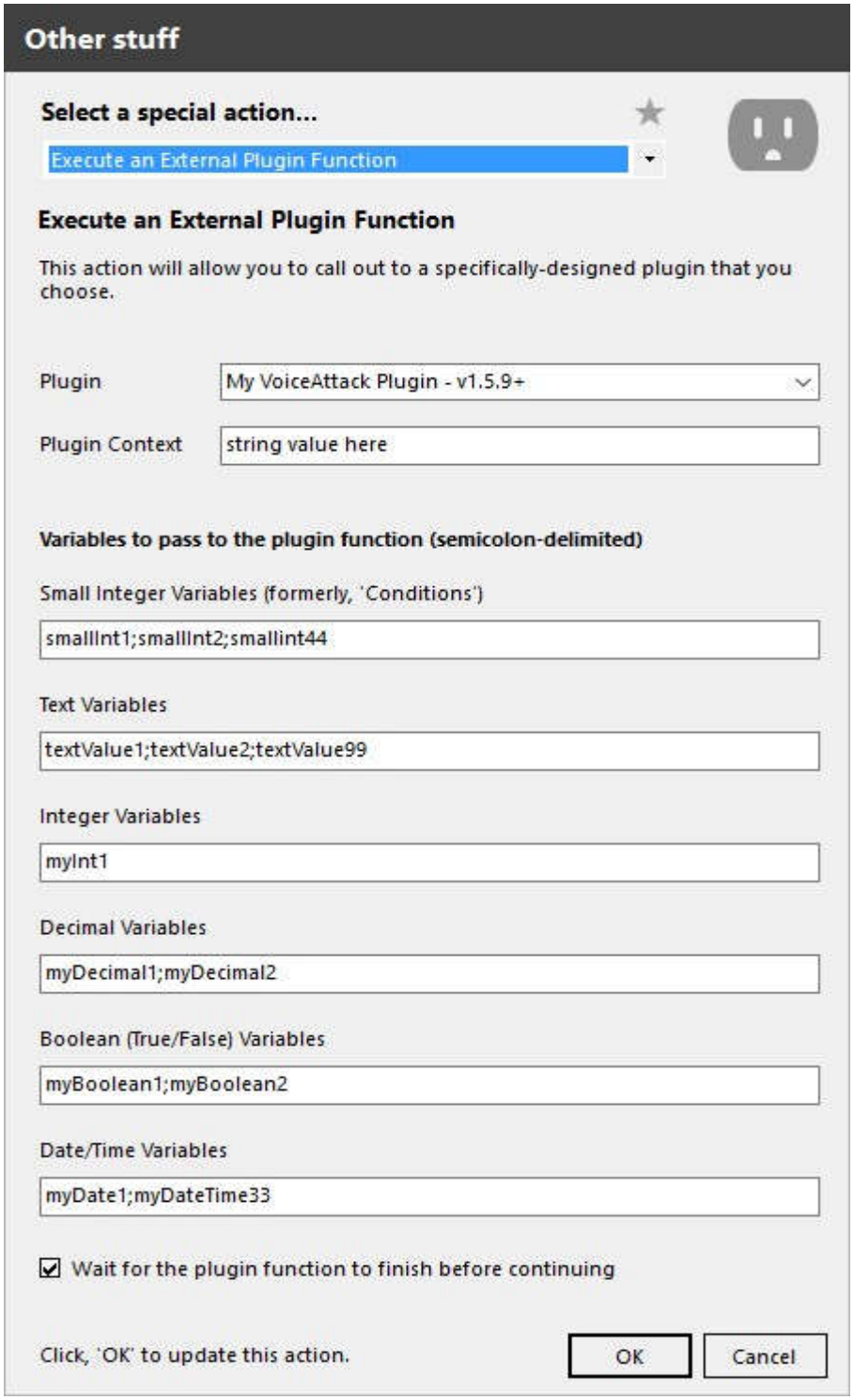




This action will allow you to clear all saved variables of a given type from a profile. This is an easy way to initialize or clean up what has been persisted to disk. Simply select the data type(s) to clear using the check boxes provided.
このアクションにより、プロファイルから特定のタイプの保存されたすべての変数をクリアできます。これは、ディスクに保存されているものを初期化またはクリーンアップする簡単な方法です。提供されているチェックボックスを使用して、消去するデータタイプを選択するだけです。

Note: This will clear the variable values that are saved to the profile, but will NOT clear the resident variables from memory.
注：これにより、プロファイルに保存されている変数値は消去されますが、常駐変数はメモリから消去されません。

'Execute an External Plugin Function'
「外部プラグイン関数の実行」



This command action is what you will use to invoke an external plugin function (see plugin section near the end of this document). To invoke a plugin, you must have enabled VoiceAttack plugin support (see, 'Options' page). If you have plugins installed, you can pick one to invoke from the dropdown list. To know how to interact with a
このコマンドアクションは、外部プラグイン関数を呼び出すために使用します（このドキュメントの最後にあるプラグインセクションを参照してください）。プラグインを呼び出すには、VoiceAttackプラグインのサポートを有効にしておく必要があります（「オプション」ページを参照）。プラグインをインストールしている場合は、ドロップダウンリストから呼び出すプラグインを選択できます。と対話する方法を知るために

plugin, you must see the documentation provided by the developer of your plugin. The plugin developer will explain what values need to go in which of the several boxes.
プラグインの場合、プラグインの開発者が提供するドキュメントを参照する必要があります。プラグイン開発者は、いくつかのボックスのどれに値を入れる必要があるかを説明します。

Below is a short reference on what each control does.
以下は、各コントロールの機能に関する簡単なリファレンスです。

Plugin Context – This is a string value that can be used to pass a simple value to the plugin. This can be anything you want, including any combination of replacement tokens.
プラグインコンテキスト–これは、単純な値をプラグインに渡すために使用できる文字列値です。これは、置換トークンの任意の組み合わせなど、必要なものであれば何でもかまいません。

Small Integer Variables (formerly referred to as, 'Conditions') – This is a semicolon– delimited list of small integer (condition) variable names that you want to pass to the plugin. They can be small integers that already exist (see, 'Set a Small Integer (Condition) Variable' command action above), or new values that you want the plugin to fill (optional). The values can be modified in the plugin and returned to VoiceAttack for further processing.
小整数変数（以前は「条件」と呼ばれていました）–これは、プラグインに渡す小整数（条件）変数名のセミコロン区切りリストです。それらは、既に存在する小さな整数（上記の「小さな整数（条件）変数を設定する」コマンドアクションを参照）、またはプラグインに入力する新しい値（オプション）です。プラグインで値を変更し、さらに処理するためにVoiceAttackに返すことができます。

Text Variables – This works exactly the same as small integers, except you are passing the text variables instead.
テキスト変数–これは、代わりにテキスト変数を渡すことを除いて、小さな整数とまったく同じように機能します。

Integer Variables – This works exactly the same as small integers, except you are passing integer variables instead.
整数変数–これは、整数変数を代わりに渡すことを除いて、小さな整数とまったく同じように機能します。

Decimal Variables – This works exactly the same as small integers, except you are passing decimal variables instead.
小数変数–これは小整数とまったく同じように機能しますが、代わりに小数変数を渡します。

Boolean Variables – This works exactly the same as small integers, except you are passing Boolean variables instead.
ブール変数–これは、ブール変数を代わりに渡すことを除いて、小さな整数とまったく同じように機能します。

Date/Time Variables – This works exactly the same as small integers, except you are passing date/time variables instead.
日付/時間変数–これは、代わりに日付/時間変数を渡すことを除いて、小さな整数とまったく同じように機能します。

'Wait for the plugin function to finish before continuing' – This option allows you to have VoiceAttack wait until the plugin is finished so that you may have a chance to react to changes that have been made in the plugin. For instance, maybe the plugin goes out to the internet and retrieves data. The plugin packages up the data and returns it to VoiceAttack. If this option is checked, VoiceAttack's next action in sequence could do something with that data (like read it with text-to-speech, or invoke some other command based on a condition).
「続行する前にプラグイン機能が終了するのを待ちます」–このオプションを使用すると、プラグインで行われた変更に反応できるように、プラグインが終了するまでVoiceAttackを待機させることができます。たとえば、プラグインがインターネットに出てデータを取得する場合があります。プラグインはデータをパッケージ化し、VoiceAttackに返します。このオプションがチェックされている場合、VoiceAttackの次のアクションは、そのデータに対して何かを行うことができます（テキスト読み上げで読み取る、条件に基づいて他のコマンドを呼び出すなど）。

'Execute an Inline Function: C# or VB.net Code'
「インライン関数の実行：C＃またはVB.netコード」

This will allow you to write C# or VB.net code that will be compiled and executed as a VoiceAttack action. Within this code, you have access to the VoiceAttack proxy object, 'VA', that is the same object used within the plugin framework so you can execute commands, get/set variables, parse tokens, etc. (see, 'Plugins for the Truly Mad'/'Plugin Parameter Notes' section later in this document for information on using this object). Since this is an advanced, specialized feature that requires a fair amount of detail, the discussion about this feature will be on the VoiceAttack user forums. For now, a light description will be provided here.
これにより、VoiceAttackアクションとしてコンパイルおよび実行されるC＃またはVB.netコードを作成できます。このコード内では、VoiceAttackプロキシオブジェクト「VA」にアクセスできます。これは、プラグインフレームワーク内で使用されるのと同じオブジェクトであるため、コマンドの実行、変数の取得/設定、トークンの解析などを実行できます（「このオブジェクトの使用に関する情報については、このドキュメントで後述する「Truly Mad」/「Plugin Parameter Notes」セクションを参照してください。これはかなりの詳細を必要とする高度な特殊機能であるため、この機能に関する議論はVoiceAttackユーザーフォーラムで行われます。ここでは、簡単な説明をここで行います。

The first thing you will probably notice is the big code window right in the middle. This is where you will write your function(s). In there, you'll see various, optional using/Imports statements that you would find in a typical new C# or VB.net forms project. You'll notice that there is a required function, 'main' that needs to be present, as well as the class that encloses it (with required name of, 'VAInline'). From, 'main', you can call your functions or instantiate your classes (or just put it all right in, 'main'). As stated above, you will have access to a dynamic-typed VoiceAttack proxy object called, 'VA'. When you create a new, 'Inline Function', you'll see some commented-out examples showing how to use, 'VA'. Note this is a very basic editor that will probably evolve as time permits ;)
おそらく最初に気付くのは、真ん中にある大きなコードウィンドウです。ここで関数を作成します。そこには、典型的な新しいC＃またはVB.netフォームプロジェクトにあるさまざまなオプションのusing / Importsステートメントがあります。存在する 必要のある「main」という必須の関数と、それを囲むクラス（required「VAInline」の名前）。'main'から、関数を呼び出すか、クラスをインスタンス化できます（または、単に 'main'に入れます）。上記のように、「VA」と呼ばれる動的に型指定されたVoiceAttackプロキシオブジェクトにアクセスできます。新しい「インライン関数」を作成すると、使用方法「VA」を示すコメントアウトされた例が表示されます。これは非常に基本的なエディターであり、おそらく時間の許す限り進化することに注意してください;)

Above the code editor is the, ‘Referenced Assemblies’ box. Within this box, you’ll see an initial set of common referenced assemblies that are separated by semicolons. You can add or remove assemblies from this list as you need them for compiling your inline function (see notes on references below). Note that full paths to assemblies can be used (e.g., “C:¥MyAssemblies¥SomeAssembly.dll”), as well as a relative path to the VoiceAttack executable (e.g., “¥Shared¥Assemblies¥SomeAssembly.dll”). This box will also accept tokens.
コードエディタの上には、「参照アセンブリ」ボックスがあります。このボックス内には、セミコロンで区切られた共通の参照アセンブリの初期セットが表示されます。インライン関数のコンパイルに必要な場合は、このリストからアセンブリを追加または削除できます（以下の参照に関する注意事項を参照）。アセンブリへのフルパス（例：「C：¥ MyAssemblies ¥ SomeAssembly.dll」）、およびVoiceAttack実行可能ファイルへの相対パス（例、「¥ Shared ¥ Assemblies ¥ SomeAssembly.dll」）を使用できることに注意してください。このボックスはトークンも受け入れます。

Some notes on references:
参照に関する注意事項：

The assembly references that you indicate when you are compiling your inline function must be available to VoiceAttack when that code is actually executed. That is, all assemblies that are used by your code must be able to be found by VoiceAttack in order for your code to run. So, even though your inline function will compile (because your referenced assemblies are visible by the compiler using your explicit paths), you may receive an error when trying to run your code (either by running by clicking the, ‘Test Run’ button or when your function is called from within a command) as VoiceAttack is relying on standard search rules to locate your referenced assemblies (outlined below).
インライン関数をコンパイルするときに指定するアセンブリ参照は、そのコードが実際に実行されるときにVoiceAttackで利用できる必要があります。つまり、コードを実行するには、コードで使用されるすべてのアセンブリがVoiceAttackで検出できる必要があります。そのため、インライン関数はコンパイルされますが（参照アセンブリは明示的なパスを使用してコンパイラによって表示されるため）、コードを実行しようとするとエラーが表示される場合があります（「テスト実行」ボタンをクリックするか、関数がコマンド内から呼び出された場合）VoiceAttackは標準の検索ルールに依存して、参照されているアセンブリを見つけます（以下に概要を示します）。

For inline functions that are only compiled on-the-fly (that is, NOT precompiled but compiled just before it is run), your referenced assemblies must reside in one of three places: The Global Assembly Cache (or, ’GAC’ … which is where all the .Net framework assemblies reside. All the default assemblies in the reference box will typically be in the GAC), in the VoiceAttack installation root directory (where VoiceAttack.exe resides… kinda messy, tho) or in the Shared¥Assemblies folder (this is inside the VoiceAttack installation folder and is created when VoiceAttack is installed). If your inline function is precompiled, your referenced assemblies can exist in any of the locations listed or can reside in the same directory as the inline function’s compiled assembly. Again, your code may be able to compile successfully but you will receive an error if the code is run and VoiceAttack cannot locate your referenced assembly in any of the places listed above.
オンザフライでのみコンパイルされる（つまり、プリコンパイルではなく実行直前にコンパイルされる）インライン関数の場合、参照アセンブリは次の3つの場所のいずれかに存在する必要があります。グローバルアセンブリキャッシュ（または ’GAC’… 参照ボックス内のすべてのデフォルトアセンブリは、通常、GACにあります）、VoiceAttackのインストールルートディレクトリ（VoiceAttack.exeが存在する場所…ちょっと乱雑、tho）、またはShared ¥ Assembliesにあります。フォルダー（これはVoiceAttackインストールフォルダー内にあり、VoiceAttackのインストール時に作成されます）。インライン関数がプリコンパイルされている場合、参照されるアセンブリは、リストされている任意の場所に存在するか、インライン関数のコンパイル済みアセンブリと同じディレクトリに存在できます。再び、

Down below the code window is a simple box for a description or name that will show up in the action list (instead of just, ‘Inline C# function’).
コードウィンドウの下には、アクションリストに表示される説明または名前の単純なボックスがあります（「インラインC＃関数」の代わりに）。

The, ‘Wait for the inline function to finish before continuing’ option will cause the calling command to wait until the, ‘main’ function is finished before it resumes executing the next action. This is referred to as running the inline function
「続行する前にインライン関数が終了するのを待つ」オプションは、次のアクションの実行を再開する前に、「メイン」関数が終了するまで呼び出しコマンドを待機させます。これは、インライン関数の実行と呼ばれます

synchronously. If this option is not checked, the inline command will be run asynchronously, and the command will resume execution of its next action immediately. Note: Running your inline function asynchronously will make it so that the code you write will run on in the background with no way to stop it with a command stop (by pressing the, ‘stop all commands’ button or by issuing a, ‘stop commands’ action). In order to help with this type of situation, the VoiceAttack proxy object (‘VA’) can be queried to find out if a command stop has been issued. Simply put a check in your code using the, ‘Stopped’ property so that you can stop your function if it needs to be stopped with a command stop. Additionally, if you need to reset this flag for whatever reason, the, ‘ResetStopFlag()’ function can be used.
同期的に。このオプションがチェックされていない場合、インラインコマンドは非同期で 実行され、コマンドは次のアクションの実行をすぐに再開します。注：インライン関数を非同期で実行すると、作成したコードがバックグラウンドで実行され、コマンド停止（「すべてのコマンドを停止」ボタンを押すか、「stopコマンドのアクション）。このタイプの状況を支援するために、VoiceAttackプロキシオブジェクト（「VA」）を照会して、コマンド停止が発行されたかどうかを確認できます。「停止」を使用してコードにチェックを入れるだけで'コマンド停止で停止する必要がある場合に関数を停止できるようにするプロパティ。さらに、何らかの理由でこのフラグをリセットする必要がある場合は、 ' ResetStopFlag（）'関数を使用できます。

The, ‘Retain Instance’ option will allow you to keep the instance of the class resident for subsequent calls. When an inline function is executed, an instance of the ‘VAInline’ class is created and then the, ‘main’ function is called. When this option is not selected, the instance of ‘VAInline’ is destroyed when, ‘main’ completes. The next time the inline function is called, a fresh, new instance is created and the cycle starts over. If this option is selected, the instance is not destroyed when, ‘main’ completes and any properties that are set within the instance are maintained. This is so you can maintain your information between subsequent calls.
[インスタンスの保持]オプションを使用すると、後続の呼び出しのためにクラスのインスタンスを常駐させることができます。インライン関数が実行されると、「VAInline」クラスのインスタンスが作成され、「main」関数が呼び出されます。このオプションが選択されていない場合、「main」が完了すると「VAInline」のインスタンスが破棄されます。次回インライン関数が呼び出されると、新しい新しいインスタンスが作成され、サイクルが最初から始まります。このオプションを選択すると、「メイン」の完了時にインスタンスは破棄されず、インスタンス内で設定されたプロパティはすべて維持されます。これは、後続の呼び出し間で情報を維持できるようにするためです。

The, ‘Compile’ button will attempt to compile the code that is in the code window with the assembly references listed. The box below the code window will show the status of the compile (errors, warnings or confirmation of a successful compilation).
[コンパイル]ボタンは、リストされているアセンブリ参照を使用して、コードウィンドウにあるコードをコンパイルしようとします。コードウィンドウの下のボックスには、コンパイルのステータス（エラー、警告、またはコンパイルの成功の確認）が表示されます。

The, ‘Test Run’ button will allow you to test run the, ‘main’ function. When you click, ‘Test Run’, the code is first compiled (just as if you clicked the, ‘Compile’ button) and then run if successful (0 errors). If the code finishes, you will see, ‘Test Run Complete’ in the status box. If your function runs on a while, you’ll notice that the, ‘Test Run’ button has changed to, ‘Stop’. You can stop your function by pressing the button again.
「テスト実行」ボタンを使用すると、「メイン」機能をテスト実行できます。「テスト実行」をクリックすると、コードが最初にコンパイルされ（「コンパイル」ボタンをクリックした場合と同じように）、成功した場合（エラーなし）実行されます。コードが終了すると、ステータスボックスに「テスト実行完了」が表示されます。関数がしばらく実行されると、「テスト実行」ボタンが「停止」に変更されていることがわかります。もう一度ボタンを押すと、機能を停止できます。

Even more advanced, in case you haven’t had enough…
あなたが十分を持っていなかった場合のために、さらに高度な...

Normally, your compiled code in an Inline Function is not stored on disk. The function is completely kept in memory and only becomes compiled the first time it is run after VoiceAttack is launched. Sometimes, for whatever reason, you may not want to just have your code available to the end user, or, if your code requires a long compile time (probably not likely in this context, but you never know, right?) you may want to precompile your code into its very own file. You can do this by adding a file path to the, ‘Build Output’ box and then clicking on the, ‘Build’ button. The, ‘Build’ button works exactly like the, ‘Compile’ button, except that when the code is successfully compiled, the compiled, ‘function’ (also called an, ‘assembly’) will be written to the file location specified in the, ‘Build Output’ input box. The new file can then be referenced and executed by the, ‘Execute an Inline Function: Precompiled’ action outlined below. Note that the, ‘Build Output’ and, ‘Build’ buttons are specifically for this purpose and are not used for anything else (Frankly, I didn’t want to create an entirely separate set of screens just to do this one little bit that maybe two people out there might use… you’ll have to forgive me ;)).
通常、インライン関数でコンパイルされたコードはディスクに保存されません。関数は完全にメモリに保持され、VoiceAttackの起動後初めて実行されたときにのみコンパイルされます。場合によっては、何らかの理由で、エンドユーザーがコードを使用できるようにしたくない場合があります。または、コードに長いコンパイル時間が必要な場合（このコンテキストではおそらくそうではないかもしれませんが、知らないでしょう？）コードを独自のファイルにプリコンパイルします。あなたは、「にファイルパスを追加することによってこれを行うことができ、出力を作成し、″をクリックし、」ボックスとビルド'ボタン。「ビルド」ボタンは、「コンパイル」ボタンとまったく同じように機能しますが、コードが正常にコンパイルされると、コンパイルされた「関数」（「アセンブリ」とも呼ばれます）が、 、 'Build Output'入力ボックス。次に、新しいファイルは、以下に概説する「インライン関数の実行：プリコンパイル済み」アクションによって参照および実行できます。「Build Output」ボタンと「Build」ボタンはこの目的専用であり、他の用途には使用されないことに注意してください（率直に言って、これを少しだけ行うために完全に別個の画面セットを作成したくありませんでした多分そこに二人が使うかもしれない …あなたは私を許さなければならないでしょう;））。

Note: Since there is (currently) no debugger, you will need to use the VA.WriteToLog() function (detailed later in this document) to write values out to the VoiceAttack log.
注：（現在）デバッガーは存在しないため、VoiceAttackログに値を書き込むには、VA.WriteToLog（）関数（このドキュメントで後述）を使用する必要があります。

'Execute an Inline Function: Precompiled'
「インライン関数の実行：プリコンパイル済み」

This will allow you to execute a previously-compiled inline function that you or somebody else create. This action goes along directly with the, 'Execute an Inline Function: C#/VB.net Code' action above. An option of the, 'Execute an Inline Function: C#/VB.net Code' action is the ability to compile your code to a specified file. The resulting, 'function' (also called an, 'assembly') can then be referenced from this action and executed. To specify the file to use, just click on the '…' button to browse and select the file. You can then optionally provide a description for display purposes. The options, 'Wait for the inline function to finish before continuing' and 'Retain Instance' work exactly like they do in the, 'Execute an Inline Function: C#/VB.net Code' action listed above.
これにより、以前にコンパイルされたインライン関数を実行できます。インライン関数は、自分または他の誰かが作成します。このアクションは、上記の「インライン関数の実行：C＃/ VB.netコード」アクションと直接連動します。「インライン関数の実行：C＃/ VB.netコード」アクションのオプションは、指定したファイルにコードをコンパイルする機能です。結果の「関数」(「アセンブリ」とも呼ばれる)は、このアクションから参照して実行できます。使用するファイルを指定するには、[…]ボタンをクリックしてファイルを参照および選択します。次に、オプションで表示目的の説明を指定できます。オプション「続行する前にインライン関数が終了するのを待つ」および「インスタンスを保持する」は、上記の「インライン関数の実行：C＃/ VB.netコード」アクションとまったく同じように機能します。

Note: When using a compiled function provided by somebody else, make sure that you trust the party from which you received the file. A malicious function can cause serious harm or cause security to be compromised in your system and/or network.
注：他の人が提供するコンパイル済み関数を使用する場合は、ファイルの受信元を信頼してください。悪意のある機能は、システムやネットワークで重大な損害を引き起こしたり、セキュリティを侵害したりする可能性があります。

'Write Text to a File'
「テキストをファイルに書き込む」

This action will allow you to write text to a file. The value to be written can contain literal text or tokens. Just indicate the text you want to write in the, 'Text' input box. You will then need to select a file to output your value to. Just click on the, '…' button to browse for a file, or just type in the full path in the, 'Output File' input box.
このアクションにより、テキストをファイルに書き込むことができます。書き込まれる値には、リテラルテキストまたはトークンを含めることができます。書きたいテキストを[テキスト]入力ボックスに指定するだけです。次に、値を出力するファイルを選択する必要があります。「…」ボタンをクリックしてファイルを参照するか、「出力ファイル」入力ボックスにフルパスを入力します。

There are a couple of options that you can select. The first is whether or not you want to append the value to the target text file. If you choose the, 'Append text to file' option, the value will be written to the end of the target file. If the file does not exist, it will be created first. The next option is whether or not to overwrite an existing file. If the
選択できるオプションがいくつかあります。1つ目は、ターゲットテキストファイルに値を追加するかどうかです。「テキストをファイルに追加」オプションを選択すると、値はターゲットファイルの最後に書き込まれます。ファイルが存在しない場合は、最初に作成されます。次のオプションは、既存のファイルを上書きするかどうかです。もし

target file already exists and this option is selected, the target file will be completely overwritten. Make sure you use these options with absolute care, as you can erase or corrupt important data.
ターゲットファイルが既に存在し、このオプションが選択されている場合、ターゲットファイルは完全に上書きされます。重要なデータを消去または破損する可能性があるため、これらのオプションは必ず慎重に使用してください。

'Write a Value to the Event Log'
「イベントログに値を書き込む」

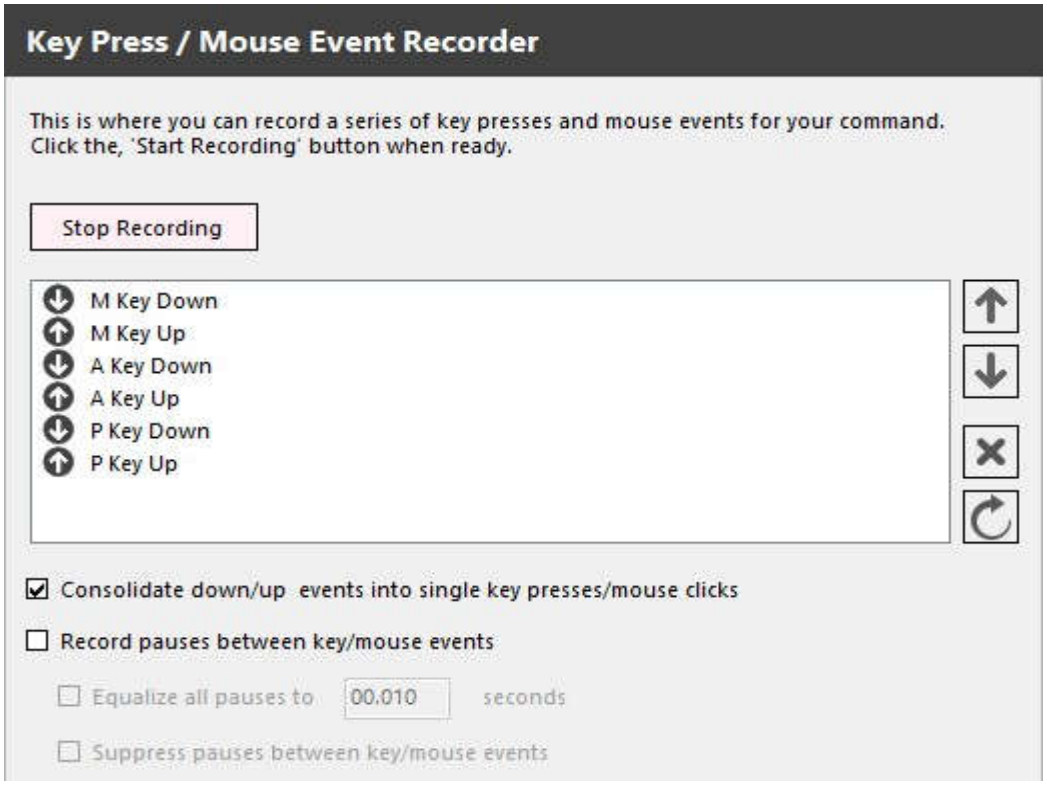
This action is very simple... just print some text to the VoiceAttack event log. Although not technically an, 'advanced' feature, this command action was created to assist in the development of commands that contain conditions and invoke plugins.
このアクションは非常に簡単です... VoiceAttackイベントログにテキストを印刷するだけです。技術的には「高度な」機能ではありませんが、このコマンドアクションは、条件を含みプラグインを呼び出すコマンドの開発を支援するために作成されました。

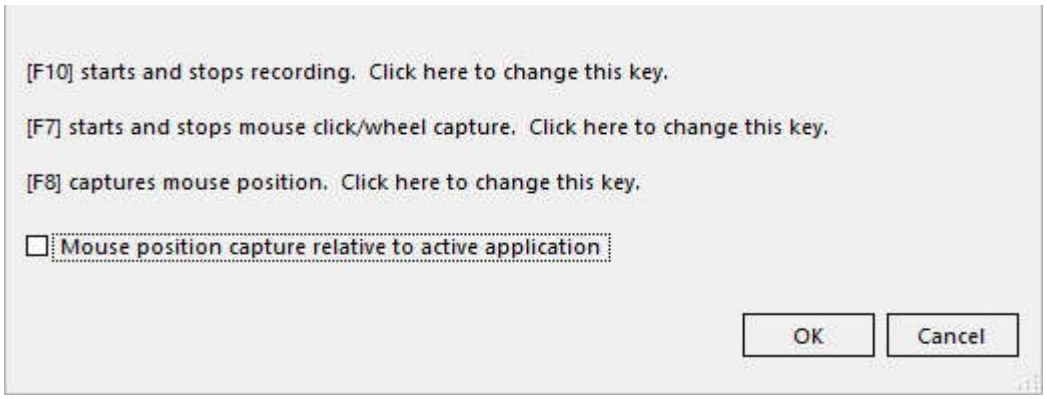
The value that you output to the log can contain as many tokens as you need. You can also specify a color-coded dot if you want:)
ログに出力する値には、必要な数のトークンを含めることができます。必要に応じて、色分けされたドットを指定することもできます。

'Add a Comment to the Action List'
「アクションリストにコメントを追加する」

This action is simply a comment that you can add to the action list for your own notation. It has no effect when executing commands. Note that this can be blank to add some spaces between actions (just to pretty things up a bit).
このアクションは、コメントとしてアクションリストに追加できる単なるコメントです。コマンドの実行時には効果がありません。これは、アクション間にスペースを追加するために空白にすることができることに注意してください（少しだけきれいにするため）。

Key Press / Mouse Event Recorder Screen
キープレス/マウスイベントレコーダー画面





This screen allows you to capture key presses and mouse events as you perform them. It will help you to create more lengthy macros quickly, as well as provide a better way to mimic human keyboard and mouse interaction (as required by a lot of games).
この画面では、キーの押下とマウスイベントを実行しながらキャプチャできます。より長いマクロをすばやく作成し、人間のキーボードとマウスの相互作用を模倣するためのより良い方法を提供するのに役立ちます（多くのゲームで必要とされます）。

To start recording key press and mouse events, click on the, 'Start Recording' button. Click this button again to stop event recording. Note that you can press an assigned keyboard hotkey to start and stop recording, and that this hotkey can be changed by clicking on the appropriate link. In the illustration, 'F10' is assigned as the hotkey, and 'F10' can be pressed to toggle event recording on and off.
キー押下とマウスイベントの記録を開始するには、[記録の開始]ボタンをクリックします。イベントの記録を停止するには、このボタンをもう一度クリックします。割り当てられたキーボードホットキーを押して記録を開始および停止でき、このホットキーは適切なリンクをクリックして変更できることに注意してください。図では、「F10」がホットキーとして割り当てられており、「F10」を押すとイベントの記録のオンとオフを切り替えることができます。

Once recording, to capture keyboard key presses, simply start typing. You'll notice that each
録音後、キーボードのキーの押下をキャプチャするには、入力を開始します。それぞれに気付くでしょう

key down and key up is recorded in the event list, along with any pauses between each event. In this example, we are not recording the pauses that occur between key events. To record pauses, check the, 'Record pauses between key/mouse events' box. If you want all of the pauses to be the same time value, check the, 'Equalize all pauses' box, then change value to what you would like the pauses to be. If your recording is very verbose (key X down, pause, key X up or mouse button X down, pause, mouse button X up), select the, 'Consolidate down/up events' box to consolidate multiple key or mouse button press events into a single command action (in the above example, M Key Down and M Key Up would become, 'Press and Release the M Key'). If you would like to omit pauses between the different key presses, simply select the, 'Suppress pauses between key/mouse events' option.
キーダウンとキーアップは、各イベント間の一時停止とともにイベントリストに記録されます。この例では、主要なイベントの間に発生する一時停止を記録していません。一時停止を記録するには、[キー/マウスイベント間の一時停止を記録する]ボックスをオンにします。すべての一時停止を同じ時間値にしたい場合は、「すべての一時停止を均等にする」ボックスをオンにして、値を一時停止したい値に変更します。録音が非常に詳細な場合（キーXダウン、一時停止、キーXアップまたはマウスボタンXダウン、一時停止、マウスボタンXアップ）、「Consolidate down / up events」ボックスを選択して、複数のキーまたはマウスボタンを押すイベントを統合します単一のコマンドアクションに変換します（上記の例では、Mキーを押し下げてMキーを押し上げると、「Mキーを押して離す」になります）。異なるキーを押すたびに一時停止を省略したい場合は、単に、

To include mouse clicks/scroll/tilt wheel events in your recording, you must first press the assigned hotkey. This is so that your mouse click recordings will start precisely when they should (and not when clicking away from VoiceAttack or moving windows around :)). To stop recording mouse clicks, simply press the assigned hotkey again. Note that the hotkey can be reassigned by clicking on the appropriate label.
マウスクリック/スクロール/チルトホイールイベントを記録に含めるには、まず割り当てられたホットキーを押す必要があります。これは、マウスクリックの記録が必要なときに正確に開始されるようにするためです（VoiceAttackから離れてクリックしたり、ウィンドウを移動したりしない場合:)）。マウスクリックの記録を停止するには、割り当てられたホットキーをもう一度押します。ホットキーは、適切なラベルをクリックして再割り当てできることに注意してください。

You can capture the mouse position in your recording by pressing its own assigned hotkey as well. In the illustration, the ‘F8’ key is the assigned hotkey. Using the illustration as an example, pressing, ‘F8’ would capture the mouse position and display the position in the event list. Again, this hotkey can be reassigned by clicking on the appropriate label. Note that there is an option available when capturing the mouse position called, ‘Mouse position capture relative to active application’. When selected, this option instructs the recorder to capture the mouse location as it relates to the active, focused application. When this option is not selected, the recorder records the mouse location as it relates to the screen.
割り当てられたホットキーを押すことでも、記録内のマウスの位置をキャプチャできます。図では、「F8」キーが割り当てられたホットキーです。図を例として使用して、「F8」を押すと、マウスの位置がキャプチャされ、イベントリストに位置が表示されます。繰り返しますが、このホットキーは適切なラベルをクリックして再割り当てできます。「アクティブなアプリケーションに対するマウス位置のキャプチャ」と呼ばれるマウス位置をキャプチャするときに使用できるオプションがあることに注意してください。選択すると、このオプションは、アクティブでフォーカスされたアプリケーションに関連するマウスの位置をキャプチャするようにレコーダーに指示します。このオプションが選択されていない場合、レコーダーは画面に関連するマウスの位置を記録します。

When you are satisfied with your recording, click the, 'OK' button to insert your recording into your command.
記録に満足したら、[OK]ボタンをクリックして、記録をコマンドに挿入します。

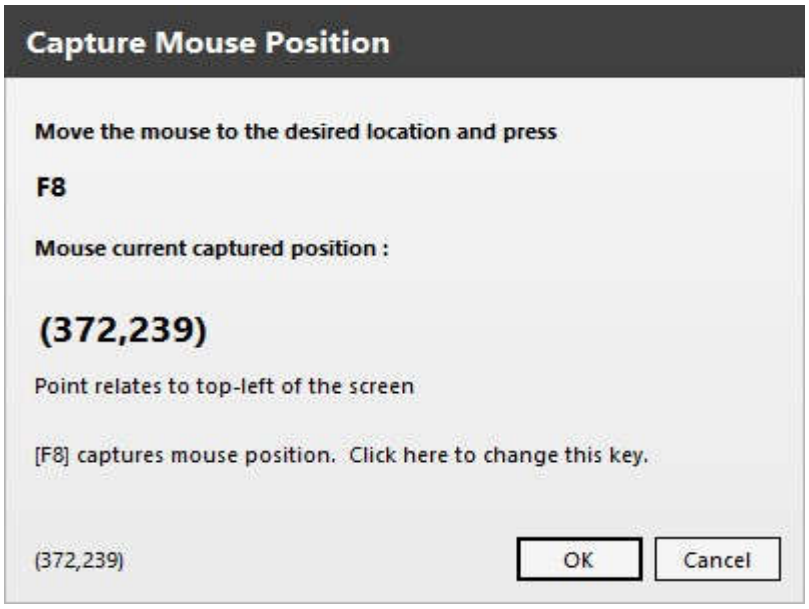
Mouse Action Screen
マウスアクション画面

The Mouse Action screen allows you to add mouse actions to your macros, such as moving the mouse, clicking a mouse button or using the scroll wheel. The Mouse Action screen is now divided up into two tabs: Move and Click.
[マウスアクション]画面では、マウスの移動、マウスボタンのクリック、スクロールホイールの使用など、マウスアクションをマクロに追加できます。[マウスアクション]画面は、[移動]と[クリック]の2つのタブに分かれています。

Mouse Move Tab
マウス移動タブ

Move to coordinates (X, Y)
座標に移動（X、Y）

Selecting this will allow you to move the mouse to a particular point on the screen. There are two ways of inputting the coordinates. First, you can manually enter the X and Y values in the input boxes, or you can click on the, 'Set Position' button to open up the 'Capture Mouse Position' helper screen:
これを選択すると、画面上の特定のポイントにマウスを移動できます。座標を入力するには2つの方法があります。まず、入力ボックスにX値とY値を手動で入力するか、「位置の設定」ボタンをクリックして「マウス位置のキャプチャ」ヘルパー画面を開きます。



To capture the mouse position, simply open your game or application and move your mouse to the location that you want and then press the designated hotkey / hotkey combination (in this example, the hotkey combination is Alt + F3). The coordinates of the mouse will then be displayed. Click, 'OK' to keep the coordinates.
マウスの位置をキャプチャするには、ゲームまたはアプリケーションを開いて、マウスを目的の場所に移動し、指定されたホットキー/ホットキーの組み合わせを押します（この例では、ホットキーの組み合わせはAlt + F3です）。マウスの座標が表示されます。[OK]をクリックして、座標を保持します。

Note: The hotkey can be reconfigured by clicking on the link provided.
注：提供されたリンクをクリックして、ホットキーを再構成できます。

There are several options that go along with the setting the mouse position. The first two are 'Screen Coordinates' and 'Application Coordinates'. Choosing, 'Screen Coordinates' indicates that the mouse position is in relation to the entire screen (this includes multiple-monitor setups). So, if your coordinates are (100, 100), you'll notice your mouse moves close to the top-left edge of your monitor.
マウスの位置の設定に伴ういくつかのオプションがあります。最初の2つは「スクリーン座標」と「アプリケーション座標」です。「画面座標」を選択すると、マウスの位置が画面全体に関連することを示します（これには、複数モニターのセットアップが含まれます）。したがって、座標が（100、100）の場合、マウスがモニターの左上近くに移動することに気付くでしょう。

'Screen Coordinates' is good for a lot of cases (especially games) but if you are using windows to do tasks, this could be almost useless (since you move you move your windows around all the time). To make the mouse position relative to the current, active window, choose 'Application Coordinates'. If the coordinates are (100, 100), you'll notice that the cursor is placed near the top left corner of whatever window you
「スクリーン座標」は多くの場合（特にゲーム）に適していますが、タスクを実行するためにウィンドウを使用している場合、これはほとんど役に立たない可能性があります（常に移動するため、ウィンドウを移動するため）。マウスの位置を現在のアクティブなウィンドウに対して相対的にするには、「アプリケーション座標」を選択します。座標が（100、100）の場合、カーソルはウィンドウの左上隅近くに配置されていることがわかります。

are looking at. Note that the, 'Capture Mouse Location' helper respects this setting.
見ています。「Capture Mouse Location」ヘルパーはこの設定を尊重することに注意してください。

The other four options for setting the mouse position deal with how the position relates to a particular corner (either the corner of the screen or the corner of a window). The default is Top-left of screen/window (since this is the option you will probably be using almost exclusively). The second-most used option will be the Bottom-right of screen/window (useful when windows are resizable… you know… so you can click that button that moves when your screen is resized). Your mouse coordinates will be in relation to the bottom-right of the screen or active window. Notice that the effective coordinates will be negative numbers, since the origin is the bottom-right corner. I hope this makes sense (lol). The 'Capture Mouse Location' helper respects this
マウスの位置を設定するための他の4つのオプションは、位置が特定のコーナー（画面のコーナーまたはウィンドウのコーナー）にどのように関係するかを扱います。デフォルトは画面/ウィンドウの左上です（これはおそらくほとんど排他的に使用するオプションであるため）。2番目に使用されるオプションは、画面/ウィンドウの右下です（ウィンドウのサイズを変更できる場合に便利です...ご存知のとおり、画面のサイズが変更されたときに移動するボタンをクリックできます）。マウス座標は、画面またはアクティブウィンドウの右下を基準にしています。原点は右下隅であるため、有効な座標は負の数値になることに注意してください。これが理にかなっていることを願っています（笑）。「Capture Mouse Location」ヘルパーはこれを尊重します

setting also.
設定も。

Move to text / token-based coordinates
テキスト/トークンベースの座標に移動する

This option allows you to specify your X and Y coordinates as tokens so you can, 'programmatically' set your mouse positions. If your X and Y tokens resolve to values that can be interpreted as integers, the position will be used. Otherwise, no movement will occur (info will be displayed in the log). If the evaluated coordinate is outside the boundaries of the area, the boundary will still be used. See the section on Tokens later in this document to find out what's available to you. Note: all variable types can be expressed using tokens.
このオプションを使用すると、XおよびY座標をトークンとして指定できるため、マウスの位置を「プログラムで」設定できます。XおよびYトークンが整数として解釈できる値に解決される場合、位置が使用されます。それ以外の場合、移動は発生しません（ログに情報が表示されます）。評価された座標が領域の境界外にある場合、境界が引き続き使用されます。このドキュメントの後半のトークンに関するセクションを参照して、利用可能なものを確認してください。注：すべての変数タイプは、トークンを使用して表現できます。

Move to specific location
特定の場所に移動する

This feature will allow you to move the mouse to specified locations. The locations are the top-left, top-right, bottom-left, bottom-right, center, top, bottom, left and right of certain screens. Choosing top-left, top-right, bottom-left or bottom-right will move the cursor to the corners of the target. Choosing center moves the mouse cursor to the middle of the target. Choosing left, right, top or bottom will move the mouse cursor
この機能により、指定した場所にマウスを移動できます。場所は、特定の画面の左上、右上、左下、右下、中央、上、下、左、右です。左上、右上、左下、または右下を選択すると、カーソルがターゲットの角に移動します。中心を選択すると、マウスカーソルがターゲットの中央に移動します。左、右、上、または下を選択すると、マウスカーソルが移動します

to that area of the target, related to the current location of the cursor. The targets are: 'Application' - the executing command's active target.
カーソルの現在の位置に関連するターゲットのその領域に。ターゲットは次のとおりです。「アプリケーション」-実行中のコマンドのアクティブなターゲット。

'Primary Screen' - the primary screen as indicated by Windows. 'Cursor Screen' - the screen where the cursor is located.
「プライマリ画面」-Windowsによって示されるプライマリ画面。「カーソル画面」-カーソルが置かれている画面。

'Active Window Screen' - the screen where the active window is located (as indicated by the top-left corner of the screen).
「アクティブウィンドウ画面」-アクティブウィンドウが配置されている画面（画面の左上隅で示される）。

'All Screens' - all active screens which constitute a, 'virtual screen'.
「すべてのスクリーン」-「仮想スクリーン」を構成するすべてのアクティブなスクリーン。

Save Current Position
現在位置を保存

Choosing this option will make VoiceAttack remember the current mouse cursor position in your macro. This, 'remembered' position can be recalled with the 'Recall mouse cursor to saved location' option.
このオプションを選択すると、VoiceAttackはマクロ内の現在のマウスカーソル位置を記憶します。この「記憶された」位置は、「保存された場所へのマウスカーソルの呼び出し」オプションで呼び出すことができます。

Recall mouse cursor to saved position
保存した位置にマウスカーソルを呼び出す

Adding this action to your command will move the mouse cursor to the last remembered position (after 'Save current position' is issued). Note: If no position is saved, the mouse position will not change. Additionally, when the 'Recall mouse cursor to saved location' action is issued, the remembered value is cleared.
このアクションをコマンドに追加すると、マウスカーソルが最後に記憶された位置に移動します（「現在の位置を保存」が発行された後）。注：位置が保存されていない場合、マウスの位置は変更されません。さらに、「保存場所へのマウスカーソルの呼び出し」アクションが発行されると、記憶されていた値はクリアされます。

Move Mouse Left / Right / Up / Down (Adjust the Mouse Cursor Location)
マウスを左/右/上/下に移動（マウスカーソルの位置を調整）

These actions will allow you to move your mouse a specified number of pixels. You
これらのアクションにより、指定したピクセル数でマウスを動かすことができます。君は

can choose to move your mouse left/right and up/down. Simply select the radio button to select the direction and type the number of units to move in the box provided. The input boxes for each direction will allow you to input numbers (like 100), tokens (like, '{INT:myVariable}' (see the VoiceAttack token reference later in this document for information on tokens)) and long integer variables (like 'myVariable'). VoiceAttack will attempt to resolve your numbers, tokens or variables into a value and move the mouse accordingly. If the value provided cannot be resolved into an integer, the value will be resolved as zero and the mouse will not be moved in that particular direction.
マウスを左右/上下に移動することを選択できます。ラジオボタンを選択して方向を選択し、提供されるボックスに移動するユニットの数を入力するだけです。各方向の入力ボックスを使用すると、数字（100など）、トークン（ '{INT：myVariable}'（トークンに関する情報についてはこのドキュメントのVoiceAttackトークンリファレンスを参照）およびロング整数変数（たとえば「myVariable」）。VoiceAttackは、数値、トークン、または変数を値に解決し、それに応じてマウスを移動しようとします。指定された値を整数に解決できない場合、値はゼロとして解決され、マウスはその特定の方向に移動しません。

The option, 'Move from cursor position' will make your mouse movement relative to the current location of your mouse cursor. The option, 'Move from last-moved position' will make the mouse move from the last place that VoiceAttack had set it (note that if this position has not been set, the current mouse cursor location will be used).
オプション ' カーソル位置から移動 'は、マウスカーソルの現在位置を基準にしてマウスを移動します。オプション「最後に移動した位置から移動」は、VoiceAttackが最後に設定した場所からマウスを移動します（この位置が設定されていない場合、現在のマウスカーソル位置が使用されることに注意してください）。

Some 3D games require a certain type of input to work properly. The option labeled, 'Move using relative data' is provided to assist with moving the mouse in, '3D space'. Try checking this box if your ship or your FPS character doesn't respond to mouse movements.
一部の3Dゲームでは、適切に機能するために特定の種類の入力が必要です。「相対データを使用して移動」というラベルの付いたオプションは、「3Dスペース」でのマウスの移動を支援するために提供されています。船またはFPSキャラクターがマウスの動きに反応しない場合は、このボックスをチェックしてみてください。

A fun feature added to the movement of the mouse is the ability to do some simple mouse animation. This was added to help with keeping track of the mouse, versus having it just move to another part of the screen instantly. To animate your mouse cursor movement, simply check the option labeled, ‘Animate movement’. There are some options that go along with animating the mouse that are required. First, you need to choose whether or not your movement will ease in to its destination or just move consistently. To make your mouse ease into its destination, select, ‘Ease movement’, otherwise, uncheck the box. The next two parts go hand-in-hand: ‘Timing’ and, ‘Steps’. Steps is the maximum number of steps to take when animating your mouse, and Timing is a base time for the movement. Increasing the timing makes the mouse cursor take longer to reach its final destination. Increasing the steps increases how smooth the mouse cursor seems to move. There are catches, however. Timing is based on seconds, but the end result will usually not be exact (especially if you add a lot of steps). The more steps you add, the more the base time is going to expand. On the flipside of this is if you use the, ‘Ease movement’ option, where the mouse cursor may reach its destination faster. You will want to play around the settings to get it just right for you. Easing the movement with timing of 1.00 and 60 steps works great here, but it may not with your system.
マウスの動きに追加された楽しい機能は、簡単なマウスアニメーションを実行できることです。これは、マウスをただ画面の別の部分に移動させるのではなく、マウスの追跡を支援するために追加されました。マウスカーソルの動きをアニメーション化するには、単に、「ラベルオプションをチェック動きをアニメーション化します」。マウスのアニメーション化に必要なオプションがいくつかあります。最初に、あなたの動きが目的地にやさしくするか、単に一貫して動くかを選択する必要があります。マウスを目的地に簡単に移動するには、「動きを簡単にする」を選択します'、それ以外の場合は、チェックボックスをオフにします。次の2つの部分は、「タイミング」と「ステップ」の2つです。Stepsは、マウスをアニメーション化するときに実行するステップの最大数であり、Timingは移動の基本時間です。タイミングを長くすると、マウスカーソルが最終目的地に到達するまでの時間が長くなります。ステップを増やすと、マウスカーソルがスムーズに動くように見えます。ただし、キャッチがあります。タイミングは秒に基づいていますが、最終結果は通常正確ではありません（特に多くのステップを追加する場合）。追加するステップが多いほど、ベースタイムは長くなります。これの反対側にあるのは、マウスカーソルがより速く目的地に到達する可能性のある「移動を容易にする」オプションを使用する場合です。設定をいろいろ試して、自分に合ったものにしたいと思うでしょう。1.00と60ステップのタイミングで動きを緩和することは、ここでうまく機能します。

Mouse Click / Scroll Tab
マウスクリック/スクロールタブ

Click Left / Right / Middle / Back / Forward button
左/右/中央/戻る/進むボタンをクリックします

These actions will make the mouse click at the current point on the screen using the selected mouse button.
これらのアクションにより、選択されたマウスボタンを使用して、画面上の現在の位置でマウスがクリックされます。

Double-click Left / Right / Middle / Back / Forward button
左/右/中央/戻る/進むボタンをダブルクリックします

These actions will make the mouse double-click at the current point on the screen using the selected mouse button.
これらのアクションは、選択されたマウスボタンを使用して、画面上の現在のポイントでマウスをダブルクリックします。

Mouse Click Duration
マウスクリック時間

Available only for click and double-click actions, this value is the amount of time in seconds between when the mouse button is pressed and when it is released. For most Windows applications, this value can be zero. For games, however, a value of zero is usually too fast to be detected reliably. The default value is 0.1 second, which may need to be adjusted for your application.
クリックアクションとダブルクリックアクションでのみ使用できます。この値は、マウスボタンが押されてから離されるまでの時間（秒単位）です。ほとんどのWindowsアプリケーションでは、この値はゼロになる場合があります。ただし、ゲームの場合、値0は通常、高速で信頼性の高い検出ができません。デフォルト値は0.1秒で、アプリケーションに合わせて調整する必要がある場合があります。

Left / Right / Middle / Back / Forward button down
左/右/中央/戻る/進むボタンを押す

These actions will make the mouse press down only at the current point on the screen using the selected button. This effectively simulates, 'press-and-hold', and, you will need to add a subsequent release.
これらのアクションは、選択されたボタンを使用して、画面上の現在のポイントでのみマウスを押し下げます。これは効果的に「長押し」をシミュレートします。その後のリリースを追加する必要があります。

Release Left / Right / Middle / Back / Forward button
左/右/中央/戻る/進むボタンを離します

These actions will make the mouse release the selected button. Use this as a follow- up to a previous mouse button down action.
これらのアクションにより、マウスは選択されたボタンを離します。これを前のマウスボタンダウンアクションへのフォローアップとして使用します。

Toggle Left / Right / Middle / Back / Forward button
左/右/中央/戻る/進むボタンの切り替え

These actions will make the mouse press down if it is up, or release if it is down (see above for more info).
これらのアクションにより、マウスが上にある場合はマウスを押し下げ、下にある場合は離します（詳細については上記を参照）。

Scroll wheel forward / backward
スクロールホイールを前後に動かす

These actions will allow you to scroll the mouse wheel forward or backward by however many number of 'wheel clicks' that you choose.
これらのアクションにより、選択した「ホイールクリック」の回数に関係なく、マウスホイールを前後にスクロールできます。

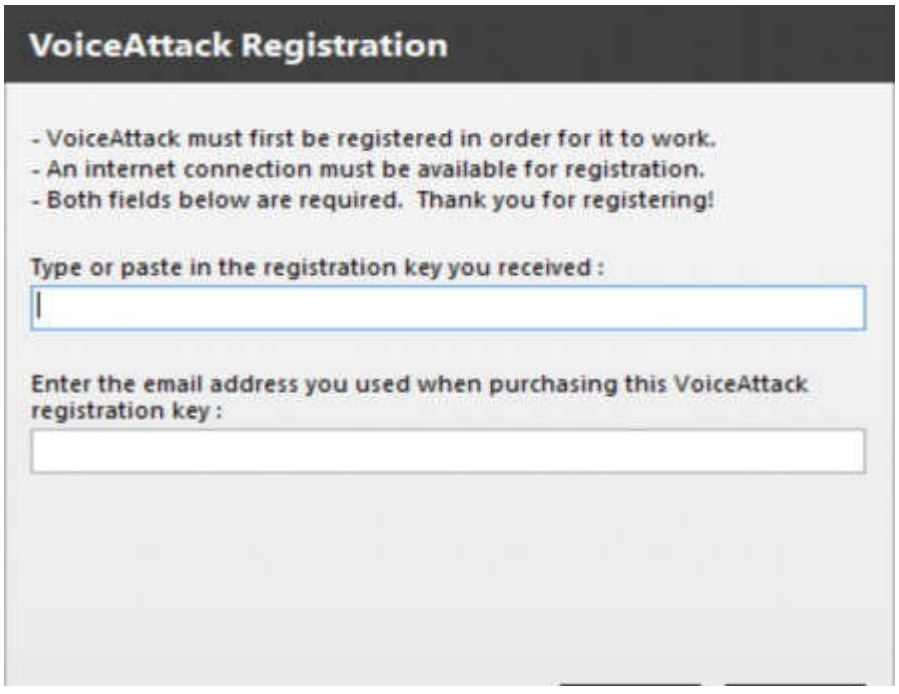
Note: To simulate what an actual mouse does when clicking, any mouse down action will now set the command's process target to be the application of the window that is located under the mouse. This will be for the remainder of the executing command. In older versions of VoiceAttack, the active window or specified process target always received the command's mouse messages for input. This would cause problems unless you knew a lot about VoiceAttack. Not good for something that should be so simple, right?
注：実際のマウスがクリックしたときの動作をシミュレートするために、任意のマウスダウンアクションで、コマンドのプロセスターゲットがマウスの下にあるウィンドウのアプリケーションに設定されるようになりました。これは、実行中のコマンドの残りの部分に使用されます。VoiceAttackの古いバージョンでは、アクティブウィンドウまたは指定されたプロセスターゲットは、入力用のコマンドのマウスメッセージを常に受信していました。VoiceAttackについてよく知らない限り、これは問題を引き起こします。とてもシンプルなものには向いていませんよね？

In order to turn this feature off, select the option, 'Bypass Mouse Targeting' on the System tab on the Options screen.
この機能をオフにするには、[オプション]画面の[システム]タブで[マウスターゲティングのバイパス]オプションを選択します。

Registration Screen
登録画面

If you are using the limited trial version of VoiceAttack and are still within the trial period, you will see the splash screen below each time you run VoiceAttack. In the bottom-left corner, there will be an indication of how much time is left to use the trial.
VoiceAttackの限定試用版を使用していて、まだ試用期間内である場合、VoiceAttackを実行するたびに下にスプラッシュ画面が表示されます。左下隅には、トライアルを使用するために残っている時間の表示があります。

If you have run out of time, or, you have clicked the 'Options' button on the main screen (See 'Options Screen') and then clicked the 'Registration' button, you will be presented with the registration screen for VoiceAttack:
時間切れになった場合、またはメイン画面の「オプション」ボタンをクリックして（「オプション画面」を参照）、「登録」ボタンをクリックした場合、VoiceAttackの登録画面が表示されます。



If you have purchased a registration key from VoiceAttack.com, this is the place to put it in. Simply type or paste in your registration key into the box provided. Additionally, you will need to put in the very same email address that you used when you purchased the VoiceAttack registration key. Click the, 'OK' button to start the validation process. This should only take seconds depending on your Internet connection and firewall settings.
VoiceAttack.comから登録キーを購入した場合は、これを配置します。登録キーを入力するか、提供されているボックスに貼り付けます。さらに、VoiceAttack登録キーを購入したときに使用したものとまったく同じメールアドレスを入力する必要があります。「OK」ボタンをクリックして、検証プロセスを開始します。これには、インターネット接続とファイアウォールの設定に応じて数秒しかかかりません。

Upon successful validation, the registration screen changes to display only what you have just entered, just for your records. If validation is not successful (for various reasons), you can hit cancel to continue using the VoiceAttack trial until time runs out.
検証が成功すると、登録画面が変わり、入力したものだけが記録用に表示されます。検証が成功しない場合（さまざまな理由）、キャンセルを押して、タイムアウトになるまでVoiceAttackトライアルの使用を続けることができます。

Options Screen
オプション画面

The, Options screen allows you to configure your VoiceAttack application. The Options screen and it's supporting dialogs are outlined below. Note that the Options screen now has three tabs with various settings on each tab.
[オプション]画面では、VoiceAttackアプリケーションを構成できます。[オプション]画面とそのサポートダイアログの概要を以下に示します。[オプション]画面には3つのタブがあり、各タブにはさまざまな設定があります。

General tab
一般タブ

Below are the items you'll find on the, 'General' tab of the Options screen. Registration Button
以下は、オプション画面の「一般」タブにある項目です。登録ボタン

Click this to bring up the registration screen (See 'Registration Screen').
これをクリックして、登録画面を表示します（「登録画面」を参照）。

Check for Updates Button
更新ボタンを確認

Click this button to have VoiceAttack check if an update is available for download (internet connection is required, of course). An additional option for this feature is ‘Include Beta Versions’, which will allow you to also check for VoiceAttack pre-release versions (if you like to keep up with the latest VoiceAttack builds).
このボタンをクリックして、VoiceAttackに更新がダウンロード可能かどうかを確認させます（もちろん、インターネット接続が必要です）。この機能の追加オプションは「Include Beta Versions」です。これにより、VoiceAttackのプレリリースバージョンも確認できます（最新のVoiceAttackビルドに対応したい場合）。

Reset Defaults Button
デフォルトのリセットボタン

Click this button if you want VoiceAttack to return all of the settings back to what
VoiceAttackですべての設定を元に戻すには、このボタンをクリックします

they were when it was first installed. The screen presented also contains an option to remove all registration data. If the option is selected, the registration data will also be reset. Note: Profile data is not reset.
それらは最初にインストールされたときでした。表示される画面には、すべての登録データを削除するオプションも含まれています。オプションが選択されている場合、登録データもリセットされます。注：プロファイルデータはリセットされません。

System Info Button
システム情報ボタン

Clicking this button will allow you to display information about VoiceAttack, your system and DirectX.
このボタンをクリックすると、VoiceAttack、システム、およびDirectXに関する情報を表示できます。

Launch with Windows Start
Windowsスタートで起動

Checking this option will make VoiceAttack launch when Windows is started. This is a per-user setting. Note: If VoiceAttack is running as an administrator, Windows may prevent it from launching at startup.
このオプションをオンにすると、Windowsの起動時にVoiceAttackが起動します。これはユーザーごとの設定です。注：VoiceAttackが管理者として実行されている場合、Windowsは起動時に起動できない場合があります。

Minimize to System Tray
システムトレイへの最小化

When selected, a VoiceAttack icon is placed in the System Tray. When unselected, VoiceAttack minimizes to the task bar.
選択すると、VoiceAttackアイコンがシステムトレイに配置されます。選択しない場合、VoiceAttackはタスクバーに最小化します。

Check for Updates on Startup
起動時に更新を確認する

Selecting this option will make VoiceAttack check for updates each time it is started. You will only get a message if there is actually an update available.
このオプションを選択すると、VoiceAttackが起動されるたびに更新をチェックします。実際に利用可能な更新がある場合にのみ、メッセージが表示されます。

Start Minimized
最小化を開始

Select this option to make VoiceAttack start up in a minimized window state.
VoiceAttackを最小化されたウィンドウ状態で起動するには、このオプションを選択します。

Show Tips at Startup
起動時にヒントを表示する

When this is selected, the VoiceAttack Tips screen will be displayed each time VoiceAttack is launched.
これを選択すると、VoiceAttackが起動するたびにVoiceAttackのヒント画面が表示されます。

Close Button Minimize Only
閉じるボタン最小化のみ

When this option is checked, clicking the, 'close' button on the main screen will just minimize VoiceAttack instead of closing it. Closing VoiceAttack can then be done from the task bar or the system tray (depending on if, 'Minimize to System Tray' is enabled or not).
このオプションがチェックされている場合、メイン画面の「閉じる」ボタンをクリックすると、VoiceAttackを閉じるのではなく最小化します。VoiceAttackの終了は、タスクバーまたはシステムトレイから実行できます([システムトレイに最小化]が有効かどうかによって異なります)。

Show Control Tips
コントロールのヒントを表示

Tool tips pop up to give descriptions of most items. Deselect this box to turn them off. Default value is 'selected'.
ツールヒントがポップアップ表示され、ほとんどのアイテムの説明が表示されます。オフにするには、このボックスの選択を解除します。デフォルト値は「選択」です。

Enable Auto Profile Switching
自動プロファイル切り替えを有効にする

This turns the VoiceAttack automatic profile switching on and off for all profiles
これにより、すべてのプロファイルに対してVoiceAttackの自動プロファイル切り替えがオンおよびオフになります。

that are enabled. See the Profile Options page for more details on automatic profile switching.
有効になっています。自動プロファイル切り替えの詳細については、「プロファイルオプション」ページを参照してください。

Enable Plugin Support
プラグインサポートを有効にする

This will turn on plugin support in VoiceAttack. Plugins are external pieces of code that can be custom-built to work with VoiceAttack. See the section entitled, 'Plugins' near the end of this document for more information. Note that this is an advanced feature of VoiceAttack, and you will receive a warning box indicating the dangers of enabling this feature.
これにより、VoiceAttackのプラグインサポートが有効になります。プラグインは、VoiceAttackで動作するようにカスタム構築できる外部コードです。詳細については、このドキュメントの最後にある「プラグイン」というタイトルのセクションを参照してください。これはVoiceAttackの高度な機能であり、この機能を有効にすることの危険性を示す警告ボックスが表示されます。

Plugin Manager Button
プラグインマネージャーボタン

When plugins are enabled, you can access the, 'Plugin Manager' screen. This basic screen will allow you to enable or disable specific plugins. Note that enabling or disabling plugins will require a restart of VoiceAttack to become effective.
プラグインを有効にすると、「プラグインマネージャー」画面にアクセスできます。この基本画面では、特定のプラグインを有効または無効にすることができます。プラグインを有効または無効にするには、VoiceAttackを再起動して有効にする必要があります。

Load Profile on Startup
起動時にプロファイルをロード

This option will allow you to select a profile that gets loaded when VoiceAttack is launched. Setting this option to, 'None' will revert back to the default behavior, which is loading the last-used profile on startup.
このオプションを使用すると、VoiceAttackの起動時にロードされるプロファイルを選択できます。このオプションを「なし」に設定すると、デフォルトの動作に戻り、起動時に最後に使用したプロファイルがロードされます。

Global Profiles
グローバルプロファイル

This option will allow you to reference, or, ‘include’ the commands from any or all of your other profiles with any current, active profile. This way, you can create common profiles filled with commands that can be shared among all profiles. The profiles that you include can be arranged in priority, from highest to lowest. When duplicate-named commands are encountered, the command in the profile with the higher priority will be retained. For instance, let’s say you have two profiles that you want to include: Profile A and Profile B. Profile A has been given a higher priority than Profile B (it’s higher up on the list). Both profiles have a command called, ‘Fire Weapons’. When you issue the command, ‘Fire Weapons’, the command from Profile A will be used, since Profile A has a higher priority.
このオプションを使用すると、現在アクティブなプロファイルを使用して、他のプロファイルの一部またはすべてのコマンドを参照したり、「含める」ことができます。この方法で、すべてのプロファイル間で共有できるコマンドで満たされた共通プロファイルを作成できます。含めるプロファイルは、優先度の高いものから順に並べることができます。重複した名前のコマンドが検出されると、プロファイル内の優先度の高いコマンドが保持されます。たとえば、含めるプロファイルとしてプロファイルAとプロファイルBの2つのプロファイルがあるとします。プロファイルAにはプロファイルBよりも高い優先順位が与えられています（リストの上位にあります）。両方のプロファイルには、「Fire Weapons」というコマンドがあります。コマンド「Fire Weapons」を発行すると、プロファイルAの優先度が高いため、プロファイルAのコマンドが使用されます。

Note: The profiles indicated in this set are lower priority than the, ‘Include commands from other profiles’ option on the Profile Options screen (See, ‘Include commands from other profiles’ on the Profile Options screen). Also, the current, active profile will always have the highest priority.
注：このセットで示されるプロファイルは、[プロファイルオプション]画面の[他のプロファイルからのコマンドを含める]オプションよりも優先度が低くなります（[プロファイルオプション]画面の[他のプロファイルからのコマンドを含める]を参照）。また、現在のアクティブなプロファイルは常に最高の優先度を持ちます。

To edit the list of included profiles, click on the, ‘…’ button to the right. This will bring
含まれるプロファイルのリストを編集するには、右側の「…」ボタンをクリックします。これはもたらすでしょう

up the, ‘Include Profile Commands’ screen. Use the controls on the right side of the screen to add, arrange and delete included profiles. Click, ‘OK’ when you are finished.
「プロファイルコマンドを含める」画面を開きます。画面の右側にあるコントロールを使用して、含まれるプロファイルを追加、配置、削除します。完了したら、「OK」をクリックします。

Keyboard Display Layout
キーボード表示レイアウト

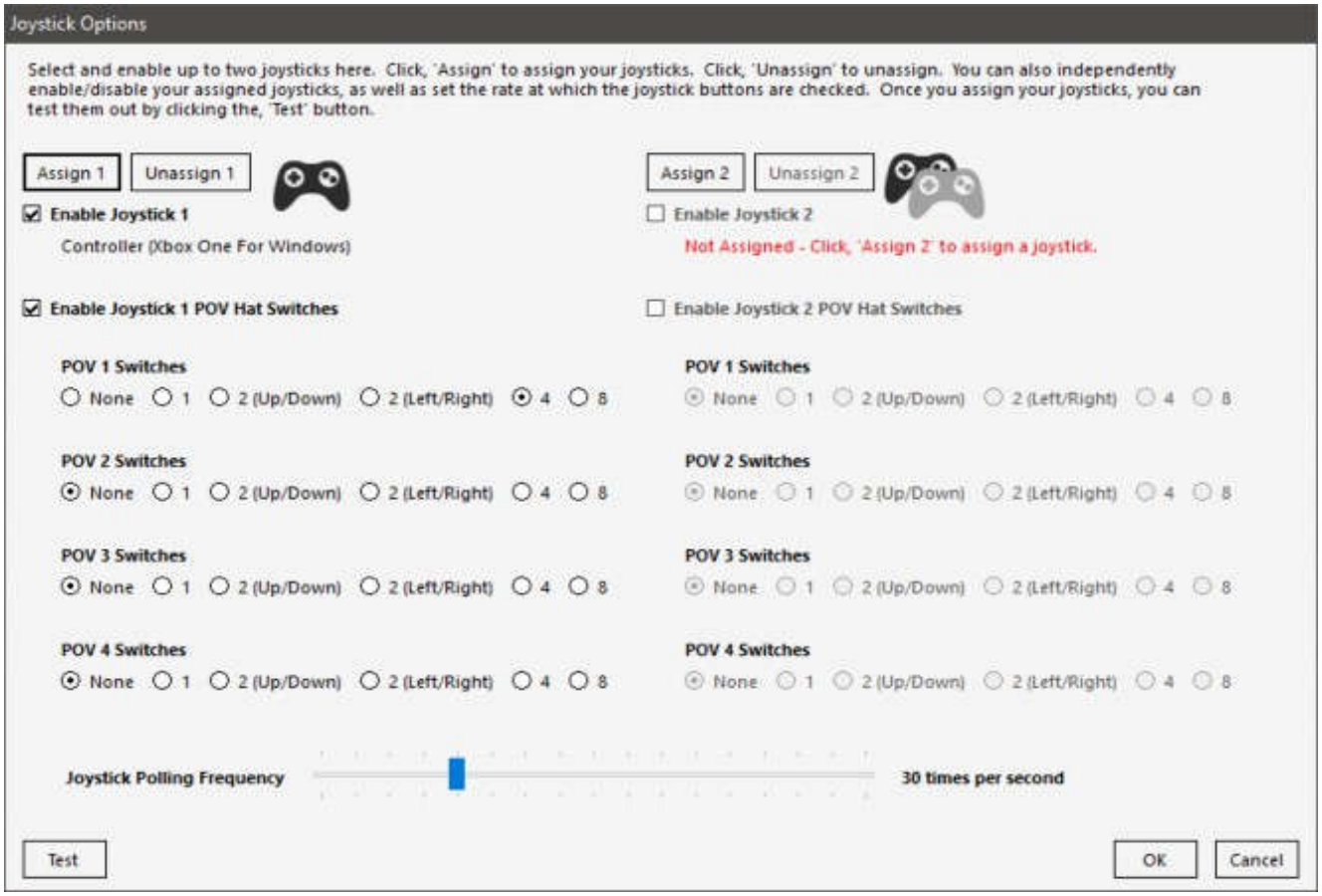
When using the Key Press screen, this option allows you to change what is displayed for selected keys. For instance, key code 51 maps to the, ’3’ key. If you are in the US, ’3 #’ is displayed. If you are in the UK, ’3 £’ is displayed. If your layout is not
キープレス画面を使用する場合、このオプションを使用すると、選択したキーの表示内容を変更できます。たとえば、キーコード51は「3」キーにマッピングされます。米国にいる場合、「3＃」が表示されます。英国にいる場合、「3£」が表示されます。レイアウトがそうでない場合

yet supported, VoiceAttack will use the US layout (which is what VoiceAttack had been using up to v1.5.3). For most users, this value will not need to be changed. If you would like to force the display to another layout, just drop down the list to select from the ones available (more to come).
まだサポートされていますが、VoiceAttackはUSレイアウトを使用します（これはVoiceAttackがv1.5.3まで使用していたものです）。ほとんどのユーザーの場合、この値を変更する必要はありません。強制的に別のレイアウトに表示したい場合は、リストをドロップダウンして利用可能なものから選択します（詳細は後ほど）。

Joystick Options
ジョイスティックのオプション

This is where you will assign and enable joysticks for use within VoiceAttack.
ここで、VoiceAttack内で使用するジョイスティックを割り当てて有効にします。

Joysticks can be used to carry out various tasks, such as executing commands as well as turning VoiceAttack's listening on and off. Clicking on this button will take you to the Joystick Options screen, shown below:
ジョイスティックを使用すると、コマンドの実行やVoiceAttackのリスニングのオン/オフなど、さまざまなタスクを実行できます。このボタンをクリックすると、以下に示すジョイスティックオプション画面が表示されます。



From this screen, you will be able to assign up to two joysticks for use in VoiceAttack. You will notice when you first open this screen, you will have no assigned joysticks. To assign a joystick to use within VoiceAttack, click the, 'Assign' button next to either joystick slot 1 or joystick slot 2. You will then be asked to select a joystick to assign
この画面から、VoiceAttackで使用するジョイスティックを2つまで割り当てることができます。この画面を最初に開くと、ジョイスティックが割り当てられていません。VoiceAttack内で使用するジョイスティックを割り当てるには、ジョイスティックスロット1またはジョイスティックスロット2の横にある[割り当て]ボタンをクリックします。次に、割り当てるジョイスティックを選択するよう求められます。

from any currently-enabled devices reporting to be joysticks on your PC (Note: Controllers identifying as gamepads will have an extra, selectable option labeled as, 'Gamepad Controller 1-4'. This extra selection is in place to assist those that are having compatibility difficulties using certain wireless controllers with VoiceAttack (weird, right?). If you are not having trouble with your gamepad controller, make sure to select the specifically-named option of your controller for maximum compatibility). Select the joystick that you want to assign and then you will be returned to the Joystick Options screen. Now that you have a joystick assigned, you can enable it or disable it by using the checkbox labeled, 'Enable Joystick (1 or 2)'. To clear a joystick assignment, just click the, 'Unassign' button next to the appropriate assignment. You'll notice that you can also change the rate at which VoiceAttack checks the state of your joysticks. This value is 30 times per second by default, but you can raise or lower that rate as you see fit.
PCのジョイスティックであると報告されている現在有効なデバイスから(注:ゲームパッドとして識別するコントローラーには、「Gamepad Controller 1-4」というラベルの付いた追加の選択可能なオプションがあります。この追加の選択は、VoiceAttackで特定のワイヤレスコントローラーを使用する際に互換性の問題がある人を支援するために用意されています(奇妙なことですか?)。ゲームパッドコントローラーで問題が発生していない場合は、互換性を最大限に高めるため、コントローラーの具体的に指定されたオプションを選択してください。割り当てるジョイスティックを選択すると、ジョイスティックオプション画面に戻ります。ジョイスティックが割り当てられたので、「ジョイスティックを有効にする(1または2)」というラベルの付いたチェックボックスを使用して、ジョイスティックを有効または無効にできます。ジョイスティックの割り当てをクリアするには、適切な割り当ての横にある[割り当て解除]ボタンをクリックするだけです。君は' VoiceAttackがジョイスティックの状態をチェックするレートを変更することもできます。この値はデフォルトで1秒間に30回ですが、適切と思われる場合はそのレートを増減できます。

In the big middle of everything is where you can set up your POV (hat) controllers
すべての大きな真ん中に、POV(帽子)コントローラーをセットアップできる場所があります

to act like simple switches. Your POV can become a switch that acts like 1, 2, 4 or 8 buttons. For instance, if you select 4, pushing forward, right, back or left on your POV will make VoiceAttack treat each position as POV button 1, 2, 3 and 4 respectively. If you select option1, VoiceAttack treats any direction as POV button 1. Selecting 2 (Up/Down), if you push up, that represents POV button 1. Pulling back represents POV button 2, and so on. VoiceAttack can support up to four POV controllers on each joystick if you've got 'em.
シンプルなスイッチのように動作します。POVは、1、2、4、または8ボタンのように機能するスイッチになることができます。たとえば、4を選択すると、POVを前後左右に押すと、VoiceAttackは各位置をPOVボタン1、2、3、4としてそれぞれ処理します。option1を選択すると、VoiceAttackは任意の方向をPOVボタン1として扱います。2(上/下)を選択すると、上に押すとPOVボタン1を表します。引き戻すとPOVボタン2を表します。VoiceAttackは、各ジョイスティックで最大4つのPOVコントローラーをサポートできます。

In the bottom-left portion of the screen, you will see the, 'Test' button. If your joysticks are assigned and enabled, you can click on this button to try them out.
画面の左下部分に、「テスト」ボタンが表示されます。ジョイスティックが割り当てられて有効になっている場合、このボタンをクリックして試してみることができます。

Sounds Folder
サウンドフォルダー

Part of the advanced features of VoiceAttack, the VoiceAttack Sounds folder makes it easy to have a central place to keep and maintain your VoiceAttack sound effects files. This folder can be located anywhere on your system and can be accessed via the
VoiceAttackの高度な機能の一部であるVoiceAttack Soundsフォルダーを使用すると、VoiceAttackサウンドエフェクトファイルを簡単に保持および管理できます。このフォルダーは、システム上のどこにでも配置でき、

{VA_SOUNDS} token (see tokens near the end of this document). Where this comes in handy is that it allows you to have kind of a virtual directory that you can export with your profile and share. Let's say you have a collection of sounds in C:¥VoiceAttack¥Sounds¥VeryCoolSoundPack. Inside this sound pack folder, you have a sound file called, 'detonate.wav'. You can set the Sounds folder to C:¥VoiceAttack¥Sounds, and access the sound file in the sound pack directory like this
{VA_SOUNDS}トークン（このドキュメントの終わり近くにあるトークンを参照）。これが役立つのは、プロファイルと共有でエクスポートできる一種の仮想ディレクトリを作成できることです。C：¥ VoiceAttack ¥ Sounds ¥ VeryCoolSoundPackにサウンドのコレクションがあるとします。このサウンドパックフォルダー内には、「detonate.wav」というサウンドファイルがあります。SoundsフォルダーをC：¥ VoiceAttack ¥ Soundsに設定し、次のようにサウンドパックディレクトリのサウンドファイルにアクセスできます。

{VA_SOUNDS}¥VeryCoolSoundPack¥detonate.wav. If you export this in a command in your profile, the token goes with it, so your friends do not have to have the same file structure as you do. They only need to have the Sounds folder set up with the, 'VeryCoolSoundPack' folder located appropriately.
{VA_SOUNDS} ¥ VeryCoolSoundPack ¥ detonate.wav。これをプロファイルのコマンドでエクスポートすると、トークンも一緒に送信されるため、友人はあなたと同じファイル構造を持つ必要はありません。適切に配置された'VeryCoolSoundPack'フォルダーを使用してSoundsフォルダーを設定するだけです。

Apps Folder
Appsフォルダー

Just like the Sounds folder above, the Apps folder makes it easy to have a central place to keep your apps (.exe files) and plugins (.dll files). This is also a special folder for VoiceAttack in regards to plugins. VoiceAttack will only look in the subfolders of this folder to look for plugins (see section on Plugins near the end of this document).
上記のSoundsフォルダーと同様に、Appsフォルダーを使用すると、アプリ（.exeファイル）とプラグイン（.dllファイル）を簡単に保管できます。これは、プラグインに関するVoiceAttackの特別なフォルダーでもあります。VoiceAttackは、このフォルダーのサブフォルダーのみを検索してプラグインを検索します（このドキュメントの最後にあるプラグインのセクションを参照してください）。

Recognition Tab
認識タブ

Speech Engine
音声エンジン

This will allow you to choose a SAPI-compliant engine. You may never need to change this. It was fun for us to mess with, but, 99.99999 percent of users will just need to leave this set to 'System Default'. More on this at a later time.
これにより、SAPI準拠のエンジンを選択できます。これを変更する必要はないかもしれません。面倒なことは面白かったのですが、ユーザーの99.99999％はこの設定を「システムのデフォルト」のままにしておくだけで済みます。これについては後で詳しく説明します。

Recognized Speech Delay
認識された音声遅延

This is the amount of time that VoiceAttack's speech engine waits to perform the actions of a command after it understands a phrase and detects a silence. Play around with this number. If you have phrases that are similar and lengthy, a higher value might be needed. If your phrases are short and different, go with a lower value. The value range is 0 - 10000 (10 seconds). The default value is 0 (it is recommended that you start at zero and work your way up if are having trouble).
これは、フレーズを理解して無音を検出した後、VoiceAttackの音声エンジンがコマンドのアクションの実行を待機する時間です。この番号で遊んでください。類似した長いフレーズがある場合は、より高い値が必要になる場合があります。フレーズが短くて異なる場合は、値を小さくしてください。値の範囲は0～10000（10秒）です。デフォルト値は0です（問題が発生した場合は、ゼロから開始して作業を進めることをお勧めします）。

A simple example of a reason to increase the value of 'Recognized Speech Delay' would be if you had two commands. The first one is, 'pet attack' and the second one is, 'pet attack plus ten'. Depending on *your* speaking ability, VoiceAttack may execute, 'pet attack' if the value is set too low. Increasing the value causes VoiceAttack to wait before execution, so you can finish articulating your command.
「認識された音声遅延」の値を増やす理由の簡単な例は、2つのコマンドがある場合です。1つ目は「ペット攻撃」で、2つ目は「ペット攻撃プラス10」です。*あなたの*話す能力に応じて、VoiceAttackは、値が低すぎると「ペット攻撃」を実行する可能性があります。値を大きくすると、VoiceAttackが実行前に待機するため、コマンドを明確に表現できます。

Unrecognized Speech Delay
認識されない音声遅延

This is the amount of time that VoiceAttack's speech engine waits to reject a speech stream that it does not understand and then detects a silence. The value range is from 0 to 10000 (10 seconds). The higher the value, the longer VoiceAttack will take to reject the speech and move on.
これは、VoiceAttackの音声エンジンが、理解できない音声ストリームを拒否してから無音を検出するまで待機する時間です。値の範囲は0～10000（10秒）です。値が高いほど、VoiceAttackが音声を拒否して先に進むのにかかる時間が長くなります。

An example of a reason to increase the, 'Unrecognized Speech Delay' value would be in a situation where you have a two-word command like, 'pet attack' (also, let's consider that you do not have a command called, 'pet'). Depending on *your* speaking ability, you may only be able to get out, 'pet' (which is unrecognized). Increasing the value causes VoiceAttack to not immediately reject the command, thereby allowing
「認識されない音声遅延」の値を増やす理由の例は、「pet attack」のような2ワードのコマンドがある場合です（また、「pet」と呼ばれるコマンドがないと考えてみましょう'）。*あなたの*話す能力によっては、「ペット」（認識されない）しか出られない場合があります。値を大きくすると、VoiceAttackはすぐにコマンドを拒否せず、

you to finish speaking your command.
あなたの命令を話すことを終えること。

Note: This setting is useful for those that leave VoiceAttack's listening on all of the time. It adjusts the delay after one finishes talking over their headset before being able to issue a command.
注：この設定は、VoiceAttackが常にリッスンしている場合に役立ちます。ヘッドセットでの会話が終了してからコマンドを発行できるようになるまでの遅延を調整します。

Command Weight
コマンド重量

This value is the relative weight of the commands in your profile versus everything else you say. The higher the number, the more likely VoiceAttack will make a, 'best guess' at what you say. For example, when this value is at maximum (100), your commands have full weight. That means that basically everything you say will be interpreted into a command. If you say, 'flag', and, you have no command called, 'flag', but, the closest match is, 'bag', VoiceAttack will execute the command named, 'bag'. Note that a high value in this option will probably not be desirable when VoiceAttack's listening is on all of the time (especially if you carry on conversations). However, a high value may be beneficial when using the push-to-talk feature. Play around with this number to get the right balance for your speaking style.
この値は、プロファイル内のコマンドと他のすべてのコマンドの相対的な重みです。数値が大きいほど、VoiceAttackはあなたの言うことを「最良の推測」する可能性が高くなります。たとえば、この値が最大（100）の場合、コマンドには完全な重みがあります。つまり、基本的にあなたが言うことはすべてコマンドに解釈されます。「flag」と言って、「flag」というコマンドがない場合、最も近い一致は「bag」である場合、VoiceAttackは「bag」という名前のコマンドを実行します。VoiceAttackのリスニングが常にオンになっている場合（特に会話を続ける場合）、このオプションの値を高くすることはおそらく望ましくないことに注意してください。ただし、プッシュツートーク機能を使用する場合は、高い値が有益な場合があります。

Minimum Confidence Level
最小信頼レベル

When the Windows speech engine recognizes a phrase, it provides a confidence rating of just how accurate it thinks it was at doing its job. VoiceAttack will allow you to filter out anything that the speech engine recognizes but does not meet a minimum rating. You can set this value here (from 0 to 100). The higher the number, the more selective VoiceAttack will be about executing commands. Note that this value can be overridden at both the profile level (on the Profile Options screen) as well as for each individual command (on the Command Add/Edit screen). Any phrase that is recognized but rejected because it falls below the minimum value will show up in the log.
Windowsスピーチエンジンはフレーズを認識すると、その仕事をしているときにどれだけ正確であると思うかについての信頼性評価を提供します。VoiceAttackを使用すると、音声エンジンが認識しているものの、最低評価を満たしていないものをすべて除外できます。ここでこの値を設定できます（0～100）。数値が大きいほど、VoiceAttackはコマンドの実行についてより選択的になります。この値は、プロファイルレベル（[プロファイルオプション]画面）と各コマンド（[コマンドの追加/編集]画面）の両方で上書きできることに注意してください。認識されたが、最小値を下回ったために拒否されたフレーズは、ログに表示されます。

Select the, 'Show confidence level' option to show the speech engine's confidence level in the log for each recognized phrase.
「信頼レベルを表示」オプションを選択して、認識された各フレーズのログに音声エンジンの信頼レベルを表示します。

Min Unrecognized Confidence Level
最小未認識信頼レベル

Setting this value allows you to filter the, 'Unrecognized' log items a bit. The higher the value, the more likely your, 'unrecognized' log items will be filtered out. This will help cut down on log chattiness in certain noisy situations.
この値を設定すると、「認識されない」ログ項目を少しフィルタリングできます。値が高いほど、「認識されない」ログアイテムが除外される可能性が高くなります。これは、特定のノイズの多い状況でログのチャットを削減するのに役立ちます。

Disable Adaptive Recognition
適応認識を無効にする

The speech engine used by VoiceAttack is constantly learning from what it, 'hears'. When environments are noisy, the speech engine may become somewhat unresponsive. Although selecting this option is not recommended, it may be very helpful if you are using VoiceAttack in a noisy place.
VoiceAttackが使用する音声エンジンは、「聞く」ことから常に学習しています。環境が騒々しい場合、音声エンジンはやや反応しなくなることがあります。このオプションを選択することはお勧めしませんが、VoiceAttackを騒がしい場所で使用している場合は非常に便利です。

Disable Acoustic Echo Cancellation
音響エコーキャンセルを無効にする

This will disable the acoustic echo cancellation on your pc in an attempt to increase recognition reliability. Note that this is a system-wide change and can affect other applications that may depend on this.
これにより、認識の信頼性を高めるために、PCの音響エコーキャンセレーションが無効になります。これはシステム全体の変更であり、これに依存する他のアプリケーションに影響する可能性があることに注意してください。

Reject Pending Speech
保留中の発言を拒否

When you turn off, 'listening' in VoiceAttack, you can choose how to handle what happens if you are speaking at the same time. When this option is not checked and you turn off, 'listening', VoiceAttack will let you finish the phrase you already started before it stops processing your commands. Checking this box will stop command processing immediately, including the phrase you are speaking.
VoiceAttackで「聞く」ことをオフにすると、同時に話している場合の処理を選択できます。このオプションがチェックされていない場合、 'listening'をオフにすると、VoiceAttackはコマンドの処理を停止する前に、既に開始したフレーズを終了させます。このボックスをオンにすると、発言中のフレーズを含むコマンド処理がすぐに停止します。

Repeat Command Phrases
コマンドフレーズを繰り返す

When, ‘When I say’ phrases get long they are long, it’s kind of a chore to repeat them over and over again. It would be easier to be able to just say, ‘repeat’ or, ‘repeat that’ and have the last spoken command be executed again. This option allows you to specify, ‘repeat’ phrases that you can speak to execute (repeat) the last spoken command. If you want to simply say, ‘repeat’, just put the word, ‘repeat’ in the input box. Note that this is a semicolon-delimited list, so, if you also wanted to say, ‘repeat that’, just enter, ‘repeat;repeat that’ in the box (without quotes). Note also that this only repeats the last spoken command, and not commands executed by other means.
「私が言うとき」のフレーズが長くなると、それらは長くなりますが、繰り返し繰り返すのは一種の雑用です。「繰り返し」または「繰り返し」と言って、最後に話されたコマンドを再度実行する方が簡単です。このオプションを使用すると、最後に話されたコマンドを実行（繰り返し）するために話すことができる「繰り返し」フレーズを指定できます。単に「繰り返し」と言いたい場合は、入力ボックスに「繰り返し」と入力してください。これはセミコロンで区切られたリストであるため、「繰り返し」と言いたい場合は、ボックスに「繰り返し;繰り返し」と入力してください（引用符なし）。また、これは最後に話されたコマンドのみを繰り返し、他の手段で実行されたコマンドは繰り返さないことに注意してください。

Recognition prefix exclusions
認識プレフィックスの除外

Sometimes when we speak commands there are ambient noises. You might breathe or make a, 'pop' noise. These noises are sometimes interpreted as words by the speech engine. For instance, if you have a command called, 'power to shields', you
コマンドを話すときに、周囲のノイズが聞こえることがあります。息をするか、「ポップ」な音を立てるかもしれません。これらのノイズは、スピーチエンジンによって単語として解釈されることがあります。たとえば、「シールドへの力」というコマンドがある場合、

might see it come up as, 'Unrecognized: if power to shields'. This is because the speech engine interpreted the, 'if' from some kind of noise
「認識されない：シールドへの電力」と表示される場合があります。これは、音声エンジンが何らかのノイズからの「if」を解釈したためです。

it picked up (mostly breathing). Items that you add to this semicolon-delimited list will be filtered out from the beginning of unrecognized phrases and reprocess those phrases to see if they are actually recognized. Adding, 'if' to the list will filter out the 'if' at the beginning of, 'if power to shields' and recheck to see if, 'power to shields' is acceptable (which it will be in this case). Your command will then be recognized.
それは拾いました（大部分は呼吸）。このセミコロン区切りリストに追加したアイテムは、認識されないフレーズの先頭から除外され、それらのフレーズを再処理して、実際に認識されるかどうかが確認されます。リストに「の先頭に「が」除外されます「が」、加算ならシールドに電源を」と、「シールドの電源が」（この場合にされる）許容可能であるかどうかを確認するために再検査。その後、コマンドが認識されます。

Note that the default is, 'if;but;the' (without quotes). That means that all three of these words will be filtered from the beginning of all unrecognized commands. You will probably need to change these if you are not using the English speech engine ;)
デフォルトは「if; but; the」（引用符なし）であることに注意してください。つまり、これらの3つの単語はすべて、認識されないすべてのコマンドの先頭からフィルタリングされます。英語の音声エンジンを使用していない場合は、おそらくこれらを変更する必要があります;)

For a long time, VoiceAttack has filtered out some words ('if' and 'but'). This doesn't work well with non-English speakers and they are hard-coded (of course). This option lets you pick what words to use. This is a semicolon-delimited list of values (the default is, 'if;but;the').
長い間、VoiceAttackはいくつかの単語（「if」と「but」）を除外していました。これは英語以外のスピーカーではうまく機能せず、ハードコーディングされています（もちろん）。このオプションを使用すると、使用する単語を選択できます。これは、セミコロンで区切られた値のリストです（デフォルトは「if; but; the」です）。

Windows Speech Recording Device
Windows音声録音デバイス

This is the recording device (microphone) that is selected by Windows for speech recognition. When VoiceAttack initializes the speech engine that it is using, this will be the device that is used. If you swap out microphones a lot, you might want to set this value to a device that you always use for speech recognition. If you only have one device, you would probably just want to leave this as, 'Default'. Once you have selected the device that you would like to use, you can set its volume from the volume slider to the right. Note that changing these values will change Windows' very own settings, so all applications that depend on these settings will then use the selected device. Also note that the settings are not saved unless you click, 'OK'.
これは、音声認識のためにWindowsによって選択される録音デバイス（マイク）です。VoiceAttackが使用している音声エンジンを初期化すると、これが使用されるデバイスになります。マイクを頻繁に交換する場合は、音声認識に常に使用するデバイスにこの値を設定することをお勧めします。デバイスが1つしかない場合は、おそらく「デフォルト」のままにしておきます。使用したいデバイスを選択したら、右側の音量スライダーから音量を設定できます。これらの値を変更すると、Windows自体の設定も変更されるため、これらの設定に依存するすべてのアプリケーションは、選択したデバイスを使用することに注意してください。また、「OK」をクリックしない限り、設定は保存されないことに注意してください。

Disable Speech Recognition
音声認識を無効にする

This option will allow you to turn off VoiceAttack's speech engine facilities. No check for a speech engine will be made at start up so you can execute VoiceAttack's commands using keyboard shortcuts, mouse buttons or joystick buttons. When this option is changed, VoiceAttack will need to be restarted before the change goes into effect (see also the, '-nospeech' command line option).
このオプションを使用すると、VoiceAttackの音声エンジン機能をオフにできます。起動時に音声エンジンのチェックは行われないため、キーボードショートカット、マウスボタン、またはジョイスティックボタンを使用してVoiceAttackのコマンドを実行できます。このオプションを変更した場合、変更を有効にする前にVoiceAttackを再起動する必要があります（「-nospeech」コマンドラインオプションも参照）。

Utilities Button
ユーティリティボタン

This button contains shortcuts to common Windows applications regarding the speech engine, such as, 'Speech Control Panel', 'Speech Engine Training', 'Add/Remove Dictionary Words', 'Microphone Setup' and, if you are using later builds of Windows 10, access to the new, 'Sound Settings' app.
このボタンには、「音声コントロールパネル」、「音声エンジントレーニング」、「辞書の単語の追加と削除」、「マイクのセットアップ」などの音声エンジンに関する一般的なWindowsアプリケーションへのショートカットが含まれます。 10、新しい「サウンド設定」アプリへのアクセス。

Audio Tab
オーディオタブ

Audio Output Type
音声出力タイプ

This option has three selections, each with their own characteristics. You will want to choose the option that best suits your needs and/or your pc's needs. Each is explained below.
このオプションには3つの選択肢があり、それぞれ独自の特性があります。あなたのニーズやPCのニーズに最適なオプションを選択する必要があります。それぞれについて以下に説明します。

Legacy Audio – This is the oldest selection, going all the way back to the beginning of VoiceAttack. This option is the most limited, but also great if your system doesn't support the other options and you really need to hear sounds. Playing audio with this option limits you to .wav files, and also does not allow you to do other things like set the volume, start position, or wait for the audio to complete.
レガシーオーディオ–これは最も古い選択であり、VoiceAttackの最初までさかのぼります。このオプションは最も制限されていますが、システムが他のオプションをサポートしておらず、本当に音を聞く必要がある場合にも最適です。このオプションを使用してオーディオを再生すると、.wavファイルに制限されます。また、音量の設定、開始位置、オーディオの完了を待つなどの操作もできません。

Windows Media Components – This is the second-oldest selection in VoiceAttack. It will allow you to set the volume of your sound and other options. The largest benefit of this method of playback is that very robust, in that it will seemingly play anything you throw at it sample-wise as long as you have a codec it can use. The drawback is that it requires Windows Media Player 10 or later to be installed on your system, since VoiceAttack will share the components that come along with it. You also cannot choose the audio output channel or pan the volume from left to right (not a big deal for most).
Windows Mediaコンポーネント–これはVoiceAttackで2番目に古い選択です。サウンドの音量やその他のオプションを設定できます。この再生方法の最大の利点は、使用できるコーデックがある限り、サンプル単位で再生するように見えるため、非常に堅牢であるということです。欠点は、VoiceAttackが付属のコンポーネントを共有するため、Windows Media Player 10以降をシステムにインストールする必要があることです。また、オーディオ出力チャンネルを選択したり、ボリュームを左から右にパンしたりすることはできません（ほとんどの場合、大したことではありません）。

Integrated Components – This is the newest option in VoiceAttack (as well as the default selection on new installations). This selection affords you the most options when playing audio, such as volume, panning, output channel selection, etc. Another benefit is that Windows Media Player is not required to be installed. The only drawback is that this option will have a hard time playing back some audio files that have variable bitrates.
統合コンポーネント–これは、VoiceAttackの最新のオプションです（新規インストールのデフォルト選択）。この選択により、音量、パン、出力チャンネルの選択など、オーディオを再生する際に最も多くのオプションが提供されます。別の利点は、Windows Media Playerをインストールする必要がないことです。唯一の欠点は、このオプションでは、可変ビットレートを持つオーディオファイルを再生するのに苦労することです。

Notification Sounds
通知音

This turns on/off notification alert sounds within VoiceAttack (such as the sounds you hear when VoiceAttack starts and stops listening). Check the box to enable notification sounds, uncheck to disable. This option is enabled by default.
これにより、VoiceAttack内の通知アラート音（VoiceAttackがリスニングを開始および停止するときに聞こえる音など）をオンまたはオフにします。チェックボックスをオンにして通知音を有効にし、オフにして無効にします。このオプションはデフォルトで有効になっています。

Fade Stopped Audio
フェードストップドオーディオ

When the playback of audio files takes place in VoiceAttack, often that audio is interrupted by a user action or command action. Stopping audio is usually abrupt and not very appealing. This option will attempt to fade out audio that is stopped and hopefully make your ears happy. Check the box to enable this feature, uncheck to disable. This option is enabled by default. Note that this feature does nothing if, 'Legacy Audio' is selected as your output type.
VoiceAttackでオーディオファイルの再生が行われると、多くの場合、そのオーディオはユーザーアクションまたはコマンドアクションによって中断されます。通常、オーディオの停止は突然であり、あまり魅力的ではありません。このオプションは、停止しているオーディオをフェードアウトしようとし、できればあなたの耳を幸せにします。この機能を有効にする場合はチェックボックスをオンにし、無効にする場合はオフにします。このオプションはデフォルトで有効になっています。出力タイプとして「レガシーオーディオ」が選択されている場合、この機能は何も実行しないことに注意してください。

Override Default Playback Device
デフォルトの再生デバイスを上書き

If you have, 'Integrated Components' selected as your audio output type (above), you'll be able to select the output channel that your audio files play through (like desktop speakers, headset, Bluetooth device, etc.). When, 'Default' is selected as the channel, VoiceAttack simply plays the audio through the default playback channel as currently indicated by Windows. Choosing a device from this option will allow you to override the Windows default device with the one selected.
オーディオ出力タイプ(上記)として「統合コンポーネント」が選択されている場合、オーディオファイルが再生される出力チャンネル(デスクトップスピーカー、ヘッドセット、Bluetoothデバイスなど)を選択できます。チャネルとして「デフォルト」が選択されている場合、VoiceAttackは、Windowsによって現在示されているデフォルトの再生チャネルを介してオーディオを再生するだけです。このオプションからデバイスを選択すると、Windowsのデフォルトデバイスを選択したデバイスで上書きできます。

Override Default Text-to-Speech Device
デフォルトの音声合成デバイスを上書きする

This works exactly like, 'Override Default Playback Device' above, except this option controls the default channel that VoiceAttack's text-to-speech audio is played through.
これは、上記の「デフォルトの再生デバイスをオーバーライドする」とまったく同じように機能しますが、このオプションは、VoiceAttackのテキスト読み上げオーディオが再生されるデフォルトのチャネルを制御します。

Stop Commands, Feature On and Feature Off Sound
停止コマンド、機能オンおよび機能オフのサウンド

For a little bit of added fun (and to possibly aid in immersion), you can set the sounds
ちょっとした追加の楽しみのために(そしておそらく没入を助けるために)、音を設定することができます

of the three base VoiceAttack sounds from a predefined set. Setting the, ‘Stop Commands Sound’ will change the sound when you stop a command. Setting the, ‘Feature On Sound’ and ‘Feature Off Sound’ options will change the on/off sounds in VoiceAttack (listening on/off, hotkeys on/off, etc.). Use, ‘Default’ for any of these to keep it like it’s always been ;) See the, ‘For fun… Maybe’ section at the end of this document for information on how to use your own sounds.
事前定義されたセットからの3つのベースVoiceAttackサウンドの。「コマンドの停止音」を設定すると、コマンドを停止したときに音が変わります。「Feature On Sound」および「Feature Off Sound」オプションを設定すると、VoiceAttackのオン/オフサウンドが変更されます（オン/オフの聞き取り、ホットキーのオン/オフなど）。これらのいずれかに「デフォルト」を使用して、常にそれを維持します;）独自のサウンドの使用方法については、このドキュメントの最後にある「楽しみのために...」セクションを参照してください。

Sound File Volume Offset
サウンドファイルのボリュームオフセット

This is the offset value for the volume of all sound files that are played using the, 'Play a Sound' command action. Think of this as a main volume control for all of your sounds. Note: This feature only works if you are not using Legacy Audio.
これは、「サウンドの再生」コマンドアクションを使用して再生されるすべてのサウンドファイルのボリュームのオフセット値です。これをすべてのサウンドのメインボリュームコントロールと考えてください。注：この機能は、レガシーオーディオを使用していない場合にのみ機能します。

TTS Volume Offset
TTSボリュームオフセット

This is the offset value for the volume of all Text-To-Speech playback using the, 'Say Something' command action. Think of this as a main volume control for Text-To- Speech in VoiceAttack.
これは、「Say Something」コマンドアクションを使用したすべての音声合成再生の音量のオフセット値です。これは、VoiceAttackのText-To-Speechのメインボリュームコントロールと考えてください。

Set Windows Default Multimedia Audio Playback Device on Startup
起動時にWindowsのデフォルトのマルチメディアオーディオ再生デバイスを設定する

This is kind of a multi-function feature that allows you to set the Windows default audio playback device (like speakers or headphones) when VoiceAttack starts. Also, the, ‘Change Now’ button is provided as a convenience so you can change the audio device immediately.
これは一種の多機能機能で、VoiceAttackの起動時にWindowsのデフォルトのオーディオ再生デバイス（スピーカーやヘッドフォンなど）を設定できます。また、オーディオデバイスをすぐに変更できるように、「今すぐ変更」ボタンが便利なように提供されています。

VoiceAttack uses Windows Media components to do its thing, and since these components only work with the default playback device, this feature may save you a step (or several) while using VoiceAttack. To use this feature, simply select the device you would like to use (the list only shows your currently-available devices). If you want to switch to the selected device on startup of VoiceAttack, just check the option box. If you want to change the device immediately, just click the, 'Change Now' button. There are also some command line options for controlling the default playback device. See, '-output' in the command line options section later in this document.
VoiceAttackはWindows Mediaコンポーネントを使用して処理を行います。これらのコンポーネントはデフォルトの再生デバイスでのみ機能するため、この機能によりVoiceAttackの使用中に1つ(または複数)の作業を節約できます。この機能を使用するには、使用するデバイスを選択するだけです(リストには現在使用可能なデバイスのみが表示されます)。VoiceAttackの起動時に選択したデバイスに切り替える場合は、オプションボックスをオンにします。デバイスをすぐに変更する場合は、[今すぐ変更]ボタンをクリックするだけです。デフォルトの再生デバイスを制御するためのコマンドラインオプションもいくつかあります。このドキュメントで後述するコマンドラインオプションセクションの「-output」を参照してください。

Note: This feature is only available for Windows 7 and later.
注:この機能は、Windows 7以降でのみ使用できます。

Note: If the device's underlying identifier is changed (driver update, Windows update, crash, etc.), VoiceAttack will attempt to resolve the device by its last-known device name. If you see a log message containing, 'Resolved by device name', VoiceAttack has successfully made the change, but you may want to update this setting.
注:デバイスの基盤となる識別子が変更された場合(ドライバーの更新、Windowsの更新、クラッシュなど)、VoiceAttackは最後に認識されたデバイス名でデバイスの解決を試みます。「デバイス名で解決」を含むログメッセージが表示された場合、VoiceAttackは変更を正常に行っていますが、この設定を更新することができます。

WARNING: This is not a VoiceAttack setting, rather a Windows device setting and can (and probably will) cause other applications that depend on the changed devices to appear to malfunction. It's not THAT big of a deal, but it will definitely throw you off when your Skype or TeamSpeak is not working how you left them.
警告:これはVoiceAttackの設定ではなく、Windowsデバイスの設定であり、変更されたデバイスに依存する他のアプリケーションが誤動作するように見える可能性があります。それほど大したことではありませんが、SkypeまたはTeamSpeakがあなたがそれらを残したように機能していない場合、間違いなくあなたを失望させます。

Set Windows Default Communications Audio Recording Device on Startup
起動時にWindowsのデフォルト通信オーディオ録音デバイスを設定する

This works exactly like the option above but for the default communications device. Everything applies, except you'll be looking for, '-outputcomms' in the command line options section.
これは上記のオプションとまったく同じように機能しますが、デフォルトの通信デバイス用です。コマンドラインオプションセクションで「-outputcomms」を探している場合を除き、すべてが適用されます。

Set Windows Default Multimedia Audio Recording Device on Startup
起動時にWindowsのデフォルトのマルチメディアオーディオ録音デバイスを設定する

This works exactly like the options above but for the default recording device (table mic, headset mic, webcam mic, etc.). Everything applies, except you'll be looking for, '-input' in the command line options section.
これは上記のオプションとまったく同じですが、デフォルトの録音デバイス(テーブルマイク、ヘッドセットマイク、ウェブカメラマイクなど)に対して機能します。コマンドラインオプションセクションで「-input」を探していることを除いて、すべてが適用されます。

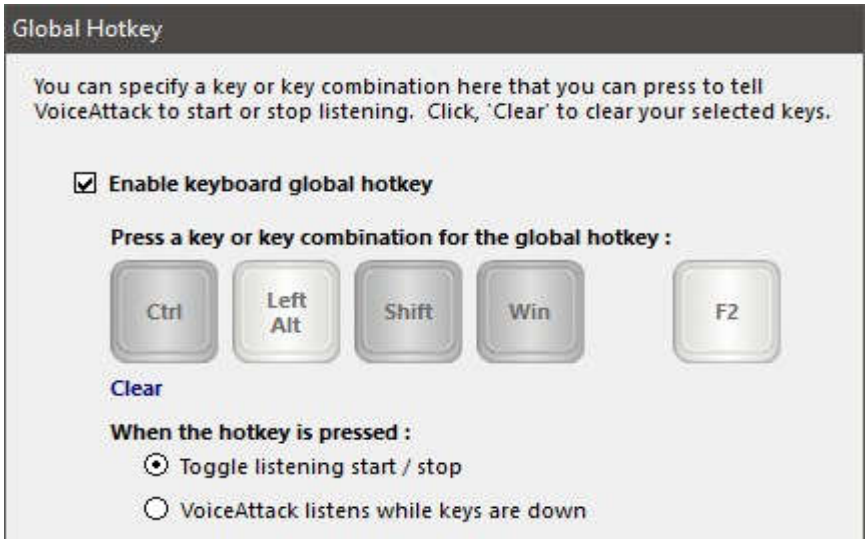
Set Windows Default Communications Audio Recording Device on Startup
起動時にWindowsのデフォルト通信オーディオ録音デバイスを設定する

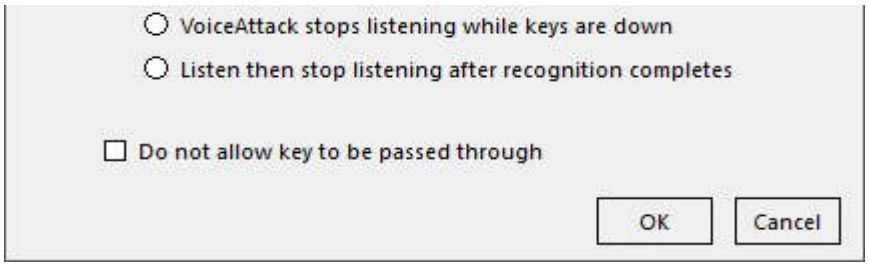
This works exactly like the option above but for the default communications device. Everything applies, except you'll be looking for, '-inputcomms' in the command line options section.
これは上記のオプションとまったく同じように機能しますが、デフォルトの通信デバイス用です。コマンドラインオプションセクションで「-inputcomms」を探している場合を除き、すべてが適用されます。

Hotkeys Tab
ホットキータブ

Recognition Global Hotkey
認識グローバルホットキー

This is the key/key combination that stops and starts VoiceAttack's, 'listening'. You can make VoiceAttack toggle listening on and off, or, you can hold down this key/key combination to make VoiceAttack listen and more. The default value for this is to toggle listening with Alt + F1 (works just like clicking on the Listening/Not Listening button on the Main screen. To change this value, click the, '…' button, and, you will be presented with the following screen:
これは、VoiceAttackの「リスニング」を停止および開始するキー/キーの組み合わせです。VoiceAttackでリッスンのオンとオフを切り替えることができます。または、このキー/キーの組み合わせを押し続けると、VoiceAttackでリッスンすることができます。これのデフォルト値は、Alt + F1でリスニングを切り替えることです(メイン画面の[リスニング/非リスニング]ボタンをクリックするのと同じように機能します。この値を変更するには、[…]ボタンをクリックします。次の画面:





From this screen, you can enable/disable this key combination. To choose a key combination, simply hit the keys that you want to use. Modifiers (shift/alt/ctrl/win) will highlight on the left, while the main key will highlight on the right. To clear all of the
この画面から、このキーの組み合わせを有効/無効にすることができます。キーの組み合わせを選択するには、使用するキーを押します。修飾キー（shift / alt / ctrl / win）は左側で強調表示され、メインキーは右側で強調表示されます。すべてをクリアするには

keys, just click the, 'clear' link.
キーは、「クリア」リンクをクリックするだけです。

The listening methods are as follows:
リスニング方法は次のとおりです。

Toggle listening start/stop – Choose this option if you want VoiceAttack to toggle listening on and off with each press of the key/key combination.
リスニングの開始/停止の切り替え–キー/キーの組み合わせを押すたびに、VoiceAttackがリスニングのオンとオフを切り替える場合は、このオプションを選択します。

VoiceAttack listens when keys are down – VoiceAttack's listening is only enabled when the key/key combination are held down.
VoiceAttackは、キーが押されたときにリッスンします– VoiceAttackのリッスンは、キー/キーの組み合わせが押されたときにのみ有効になります。

VoiceAttack stops listening while keys are down – VoiceAttack's listening is always enabled, except for when the key/key combination is held down.
VoiceAttackは、キーが押されている間、リスニングを停止します– VoiceAttackのリスニングは、キー/キーの組み合わせが押されている場合を除き、常に有効になっています。

Listen then stop listening after recognition completes – Also known as, 'Listen Once', this option enables VoiceAttack's listening until the next speech event completes (either recognized or unrecognized) and then listening is automatically disabled.
聞き取り、認識完了後に聞き取りを停止–「一度聞く」とも呼ばれるこのオプションは、次の音声イベントが完了する（認識または認識されない）までVoiceAttackの聞き取りを有効にし、自動的に無効になります。

The, 'Do not allow key to be passed through' option prevents the main key (non- modifier) from being passed through to the application. For example, if your hotkey is F1 and this option is selected, VoiceAttack will respond to the F1 key press and then prevent any other application from receiving this key press (for this example, if F1 is being handled by VoiceAttack, you will not be able to use the F1 key while other applications are running. If you rely on F1 to bring up, 'Help', then, you'll have to pick another key). Use care when selecting this option.
[キーのパススルーを許可しない]オプションは、メインキー（非修飾子）がアプリケーションにパススルーされるのを防ぎます。たとえば、ホットキーがF1でこのオプションが選択されている場合、VoiceAttackはF1キーの押下に応答し、他のアプリケーションがこのキーの押下を受け取らないようにします（この例では、F1がVoiceAttackによって処理されている場合、他のアプリケーションの実行中にF1キーを使用できます。F1を使用して「ヘルプ」を表示する場合は、別のキーを選択する必要があります。このオプションを選択するときは注意してください。

Note: If the key is part of a combination and this option is selected, hitting the main key by itself will still pass through to the application.
注：キーが組み合わせの一部であり、このオプションが選択されている場合、メインキーを単独で押すと、アプリケーションにパススルーされます。

Note: This option can be overridden at the profile level. See, 'Profile Options' screen.
注：このオプションは、プロファイルレベルでオーバーライドできます。「プロファイルオプション」画面を参照してください。

Mouse Click Recognition
マウスクリック認識

Works like the Global Hotkey, only with your mouse. You can toggle 'Listening/Not Listening', 'Listen until recognition', hold the mouse button to listen and hold the mouse button to stop listening. This feature is available on the five standard mouse
マウスでのみ、グローバルホットキーのように機能します。「リスニング/非リスニング」、「認識されるまでリッスン」を切り替え、マウスボタンを押したままにして、マウスボタンを押したままにして聞くことができます。この機能は、5つの標準マウスで使用できます

buttons (left, right, middle, forward & back).
ボタン（左、右、中央、前後）。

Note: This option can be overridden at the profile level. See, 'Profile Options' screen.
注：このオプションは、プロファイルレベルでオーバーライドできます。「プロファイルオプション」画面を参照してください。

Stop Command Hotkey
停止コマンドホットキー

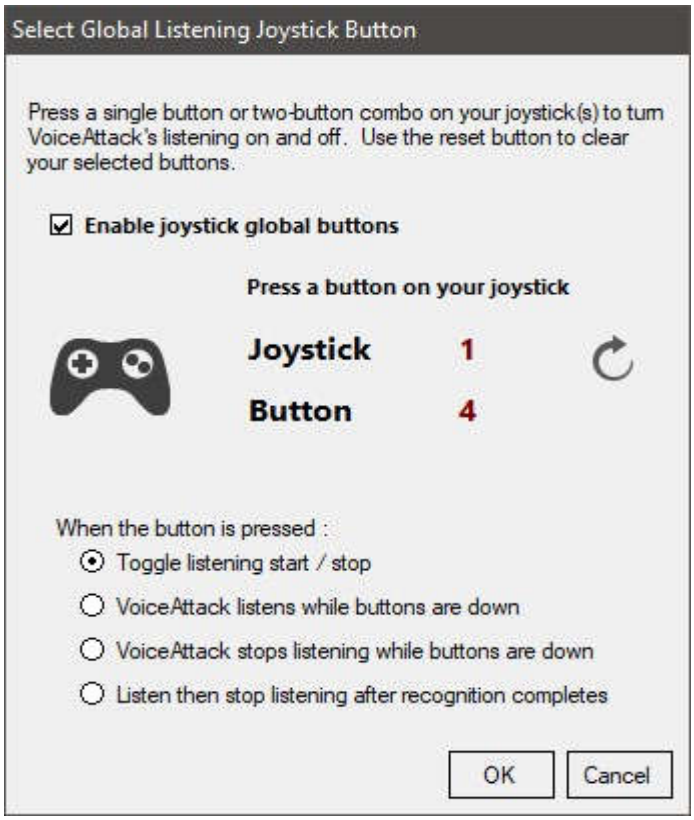
This is the hotkey you can select to act as a panic button to halt all running macros. This works the same as pressing the 'Stop Commands' button on the main screen (See, 'Main Screen'). Click the, '…' button to change this option (this works pretty much the same as the, 'Global Hotkey', so, for brevity, the screen shot and description is omitted).
これは、実行中のすべてのマクロを停止するパニックボタンとして機能するために選択できるホットキーです。これは、メイン画面の「停止コマンド」ボタンを押すのと同じ働きをします（「メイン画面」を参照）。[…]ボタンをクリックしてこのオプションを変更します（これは[グローバルホットキー]とほぼ同じように機能するため、簡潔にするためにスクリーンショットと説明は省略します）。

Note: This option can be overridden at the profile level. See, 'Profile Options' screen.
注：このオプションは、プロファイルレベルでオーバーライドできます。「プロファイルオプション」画面を参照してください。

Joystick Button Recognition
ジョイスティックボタン認識

This is the joystick button that stops and starts VoiceAttack's, 'listening'. You can make VoiceAttack toggle listening on and off, or, you can hold down this button to make
これは、VoiceAttackの「リスニング」を停止および開始するジョイスティックボタンです。VoiceAttackでリッスンのオンとオフを切り替えることができます。または、このボタンを押し続けると、

VoiceAttack listen and more. To change this value, click the, '…' button, and, you will be presented with the following screen:
VoiceAttackのリッスンなど。この値を変更するには、[…]ボタンをクリックすると、次の画面が表示されます。



From this screen, you can enable/disable this joystick button. To choose a button, simply press the button on the joystick that you want to use. To clear the button, just click the, 'reset' icon in the top-right.
この画面から、このジョイスティックボタンを有効/無効にすることができます。ボタンを選択するには、使用するジョイスティックのボタンを押すだけです。ボタンをクリアするには、右上の「リセット」アイコンをクリックするだけです。

The listening methods are as follows:
リスニング方法は次のとおりです。

Toggle listening start/stop – Choose this option if you want VoiceAttack to toggle listening on and off when the selected buttons are pressed.
リスニングの開始/停止の切り替え–選択したボタンが押されたときにVoiceAttackでリスニングのオンとオフを切り替える場合は、このオプションを選択します。

VoiceAttack listens while button is down – VoiceAttack's listening is only enabled when the selected button is held down.
VoiceAttackはボタンが押されている間、リッスンします–VoiceAttackのリッスンは、選択されたボタンが押されているときにのみ有効になります。

VoiceAttack stops listening while the button is down – VoiceAttack's listening is always enabled, except for when the selected button is held down.
VoiceAttackは、ボタンが押されている間、リスニングを停止します–VoiceAttackのリスニングは、選択されたボタンが押されている場合を除き、常に有効になっています。

Listen then stop listening after recognition completes – Also known as, 'Listen Once', this option enables VoiceAttack's listening until the next speech event completes (either recognized or unrecognized) and then listening is automatically disabled.
聞き取り、認識完了後に聞き取りを停止–「一度聞く」とも呼ばれるこのオプションは、次の音声イベントが完了する(認識または認識されない)までVoiceAttackの聞き取りを有効にし、自動的に無効になります。

Note: This option can be overridden at the profile level. See, 'Profile Options' screen.
注:このオプションは、プロファイルレベルでオーバーライドできます。「プロファイルオプション」画面を参照してください。

System/Advanced Tab
システム/詳細タブ

Bypass Mouse Targeting
マウスターゲティングのバイパス

This option disables setting the process target of the application located under the mouse on a mouse-down event. When this option is not checked, when a mouse- down event occurs (mouse down, click, double-click) the application located under the mouse is selected as the process target (overriding any set process target) for the duration of the command.
このオプションは、マウスダウンイベントでマウスの下にあるアプリケーションのプロセスターゲットの設定を無効にします。このオプションがチェックされていない場合、マウスダウンイベント（マウスダウン、クリック、ダブルクリック）が発生すると、コマンドの実行中、マウスの下にあるアプリケーションがプロセスターゲットとして選択されます（設定されたプロセスターゲットをオーバーライドします）。

Single TTS Instance
単一のTTSインスタンス

VoiceAttack will allow for essentially any number of text-to-speech (TTS) instances by default. Some TTS packages are particular about the number of instances of their voices that are running and will cause VoiceAttack to hang or crash. Check this option if you are experiencing trouble with a TTS voice package.
VoiceAttackは、デフォルトで基本的に任意の数のテキスト読み上げ（TTS）インスタンスを許可します。一部のTTSパッケージは、実行中のボイスのインスタンス数に特化しており、VoiceAttackがハングまたはクラッシュする原因になります。TTS音声パッケージで問題が発生している場合は、このオプションをオンにします。

Cancel Blocked Commands
ブロックされたコマンドをキャンセル

Checking this option will prevent commands from executing that are blocked by synchronous commands (commands that do not have the option, 'Allow other commands to be executed while this one is running' checked). Deselecting this option will cause any commands that are triggered during the block to be executed when
このオプションをオンにすると、同期コマンドによってブロックされているコマンドの実行が防止されます（「このコマンドの実行中に他のコマンドの実行を許可する」オプションがチェックされていないコマンド）。このオプションを選択解除すると、ブロック中にトリガーされたコマンドが実行されます。

the blocking command completes. Note that there is no guarantee of the order of the execution of the blocked commands and there is also no guarantee that the blocked commands will themselves be able to block. Hope that makes sense o_O.
ブロッキングコマンドが完了します。ブロックされたコマンドの実行順序の保証はなく、ブロックされたコマンド自体がブロックできるという保証もないことに注意してください。o_Oが理にかなっていることを願っています。

Show Third-Party App Warnings
サードパーティアプリの警告を表示する

Some third-party applications can cause conflicts with VoiceAttack. Selecting this option will show a warning in the log at the startup of VoiceAttack if any of the predefined applications are running. The list of applications will be updated when you click on the, 'Check for Updates' button. This value is on by default, so if you get sick of seeing warning messages when you start up, deselect this option.
一部のサードパーティアプリケーションは、VoiceAttackとの競合を引き起こす可能性があります。定義済みのアプリケーションが実行されている場合、VoiceAttackの起動時にこのオプションを選択すると、ログに警告が表示されます。「更新の確認」ボタンをクリックすると、アプリケーションのリストが更新されます。この値はデフォルトでオンになっているため、起動時に警告メッセージが表示されることにうんざりしている場合は、このオプションを選択解除してください。

Stop Running Commands When Editing
編集時にコマンドの実行を停止

When this option is selected, any commands that are running when the Profile Edit screen is opened will be stopped. This will also prevent any commands from running while the screen is open. Deselecting this option will allow commands to continue while the profile is being edited and will not stop commands from being executed. Use with caution when turning off this option.
このオプションを選択すると、プロファイル編集画面を開いたときに実行されていたコマンドはすべて停止します。これにより、画面が開いているときにコマンドが実行されなくなります。このオプションを選択解除すると、プロファイルの編集中にコマンドを続行でき、コマンドの実行が停止することはありません。このオプションをオフにするときは注意して使用してください。

Use Nested Tokens
ネストされたトークンを使用する

When this option is selected, tokens are processed from the inner-most token to the outer-most token, reprocessing from the inside out on each iteration. This allows for tokens to be nested, or, in other words, tokens can provide input values to other tokens. This may possibly cause issues with older commands. Unchecking this box puts VoiceAttack back to what it used to be and may help if some tokens are giving you problems. Note that this is a global setting, so you will want to fix up your old tokens if you want to be able to use this functionality.
このオプションを選択すると、トークンは最も内側のトークンから最も外側のトークンに処理され、各反復で内側から外側に再処理されます。これにより、トークンをネストできます。つまり、トークンは入力値を他のトークンに提供できます。これにより、古いコマンドで問題が発生する可能性があります。このボックスをオフにすると、VoiceAttackは以前の状態に戻り、一部のトークンが問題を引き起こしている場合に役立ちます。これはグローバル設定であるため、この機能を使用できるようにする場合は、古いトークンを修正する必要があります。

Allow Command Segment Info for Composite Commands
複合コマンドのコマンドセグメント情報を許可する

This option will allow you to capture command segment information when using composite (prefix/suffix) commands. Capturing command segment information requires a level of processing that may be noticeable when large numbers of composite commands are created that use dynamic command sections. Uncheck this box if you do not intend to use command segments with composite commands
このオプションを使用すると、複合（プレフィックス/サフィックス）コマンドを使用するときにコマンドセグメント情報をキャプチャできます。コマンドセグメント情報のキャプチャには、動的コマンドセクションを使用する多数の複合コマンドが作成されたときに目立つレベルの処理が必要です。複合コマンドでコマンドセグメントを使用しない場合は、このボックスをオフにします

(see the, '{CMDSEGMENT]' token for more information).
（詳細については、「{CMDSEGMENT]」トークンを参照してください）。

Upon import, profiles will have, 'Block potentially harmful profile actions' selected
インポート時に、プロファイルには「有害な可能性のあるプロファイルアクションをブロックする」が選択されます

If this option is selected, each profile that is imported will have its ‘Block potentially harmful profile actions’ option turned on. This will help keep a variety of items from being run from within a profile prior to inspection. See the, ‘Profile Options’ screen for more details.
このオプションを選択すると、インポートされる各プロファイルの[有害な可能性のあるプロファイルアクションをブロックする]オプションがオンになります。これにより、さまざまなアイテムが検査前にプロファイル内から実行されるのを防ぐことができます。詳細については、「プロファイルオプション」画面を参照してください。

Enable Log Quiet Mode
Log Quietモードを有効にする

This option controls, in some part, what information is sent to the VoiceAttack log. When this option is checked, nothing is displayed in the VoiceAttack log unless there is an entry generated by VoiceAttack that is the result of an error (indicated by a red dot). Selecting the, ‘Display Warning Log Items’ option will allow warnings to also be displayed (indicated by a yellow dot). The, ‘Display ‘Write To Log’ Actions’ option will allow any action that writes to the log to be displayed.
このオプションは、VoiceAttackログに送信される情報の一部を制御します。このオプションをオンにすると、VoiceAttackによって生成されたエラーの結果であるエントリ（赤い点で示されている）がない限り、VoiceAttackログに何も表示されません。[警告ログアイテムの表示]オプションを選択すると、警告も表示できます（黄色のドットで示されます）。「「ログに書き込む」アクションを表示」オプションを使用すると、ログに書き込むアクションを表示できます。

Run VoiceAttack as an Administrator
VoiceAttackを管理者として実行する

This option will make VoiceAttack attempt to run as an administrator. Note: If you find yourself in a situation where you cannot get into VoiceAttack to uncheck this setting, you’ll want to use the, ‘-clearasadmin’ command line parameter (which clears the setting).
このオプションにより、VoiceAttackは管理者として実行されます。注：VoiceAttackにアクセスしてこの設定のチェックを外すことができない場合は、 '-clearasadmin'コマンドラインパラメーター（設定をクリアする）を使用することをお勧めします。

Reset Screens Button
画面のリセットボタン

Pressing this button will allow you to reset the position and size of all VoiceAttack screens.
このボタンを押すと、すべてのVoiceAttack画面の位置とサイズをリセットできます。

Use Built-In SAPI Speech Engines / Use Installed Speech Platform Speech Engines
組み込みのSAPIスピーチエンジンを使用する/インストール済みのスピーチプラットフォームスピーチエンジンを使用する

If these options are enabled and available for you, that means that you have both the built-in SAPI speech engine (the one that is installed with Windows), and at least one Microsoft Speech Platform speech engine installed on your computer. This option lets you toggle between the two speech platforms. Note that if you change this option, VoiceAttack must restart before the speech platform will switch. You can learn more about installing the Microsoft Speech Platform 11 speech engines by clicking the link below the options.
これらのオプションが有効で利用可能な場合は、組み込みのSAPIスピーチエンジン（Windowsにインストールされているもの）と、少なくとも1つのMicrosoft Speech Platformスピーチエンジンがコンピューターにインストールされていることを意味します。このオプションを使用すると、2つの音声プラットフォームを切り替えることができます。このオプションを変更した場合、音声プラットフォームが切り替わる前にVoiceAttackを再起動する必要があることに注意してください。Microsoft Speech Platform 11スピーチエンジンのインストールの詳細については、オプションの下のリンクをクリックしてください。

Use Built-In SAPI Text-To-Speech / Use Installed Speech Platform TTS Synthesizers
組み込みのSAPI Text-To-Speechの使用/インストール済みのSpeech Platform TTSシンセサイザーの使用

If these options are enabled and available for you, that means that you have both the built-in SAPI speech engine (the one that is installed with Windows), and at least one Microsoft Speech Platform text-to-speech synthesizer installed. This option lets you toggle between the two platforms. Note that if you change this option, VoiceAttack must restart before the speech platform will switch. You can learn more about
これらのオプションが有効で使用可能な場合、組み込みのSAPI音声エンジン（Windowsにインストールされているエンジン）と、少なくとも1つのMicrosoft Speech Platformテキスト読み上げシンセサイザーがインストールされていることを意味します。このオプションを使用すると、2つのプラットフォームを切り替えることができます。このオプションを変更した場合、音声プラットフォームが切り替わる前にVoiceAttackを再起動する必要があることに注意してください。についての詳細を学ぶことができます

installing the Microsoft Speech Platform 11 speech engines by clicking the link below the options.
オプションの下のリンクをクリックして、Microsoft Speech Platform 11音声エンジンをインストールします。

Enable Sleep Mode
スリープモードを有効にする

When the Windows speech engine is active, Windows is prevented from going to sleep on its own. When this option is enabled, VoiceAttack will attempt to put the speech engine to sleep (‘Sleep Mode’) if no audio is detected after the number of seconds you specify. When the speech engine is off, Windows will then be able to sleep if it is able. To wake up the speech engine out of, ‘Sleep Mode’ just move the mouse, click a mouse button or press a keyboard key.
Windowsスピーチエンジンがアクティブな場合、Windowsが単独でスリープ状態になることはできません。このオプションを有効にすると、指定した秒数が経過しても音声が検出されない場合、VoiceAttackは音声エンジンをスリープ（「スリープモード」）にしようとします。音声エンジンがオフの場合、Windowsは可能な場合はスリープ状態になります。音声エンジンをウェイクアップするには、「スリープモード」でマウスを移動するか、マウスボタンをクリックするか、キーボードのキーを押します。

Prevent Speech Engine From Changing Microphone Volume
音声エンジンがマイクの音量を変更しないようにする

This option attempts to keep the speech engine from changing the microphone volume when the speech engine is started up. The reason that the microphone volume is altered by the speech engine is due to the environment at the time that you had trained the speech engine. That is, the speech engine is attempting to compensate for an environment that was either too quiet (by raising the mic volume) or too loud (by lowering the mic volume). Generally, this is a good thing, and it is recommended that you do not mess with this setting. If the setting is too far out of comfortable range, it is recommended that you train your speech engine properly in an environment that is more speech engine-friendly (you, know… quiet and without kids bouncing around).
このオプションは、スピーチエンジンの起動時に、スピーチエンジンがマイクの音量を変更しないようにします。マイクの音量が音声エンジンによって変更される理由は、音声エンジンをトレーニングしたときの環境によるものです。つまり、音声エンジンは、静かすぎる（マイクの音量を上げる）か、大きすぎる（マイクの音量を下げる）環境を補正しようとしています。一般的に、これは良いことであり、この設定を混乱させないことをお勧めします。設定が快適な範囲から離れすぎている場合は、音声エンジンに優しい環境で音声エンジンを適切にトレーニングすることをお勧めします（ご存知のように...静かで、子供が跳ね回らない）。

However, sometimes you need to be able to just go all draconian and bypass whatever is in place to see if it helps, so, this setting is for that situation. Note that this is a Windows-level setting, and any other instances of the Windows speech engine will be affected by this setting.
ただし、場合によっては、すべての過酷な作業を行って、配置されているものをすべてバイパスして、それが役立つかどうかを確認する必要がある場合があるため、この設定はその状況に適しています。これはWindowsレベルの設定であり、Windowsスピーチエンジンの他のインスタンスはこの設定の影響を受けることに注意してください。

Export Settings Button
エクスポート設定ボタン

This will allow you to save VoiceAttack's settings to a single file that you can maintain as a backup. Note that registration details and profile information are not stored in this file.
これにより、VoiceAttackの設定を単一のファイルに保存して、バックアップとして維持できます。登録の詳細とプロファイル情報は、このファイルには保存されないことに注意してください。

Import Settings Button
インポート設定ボタン

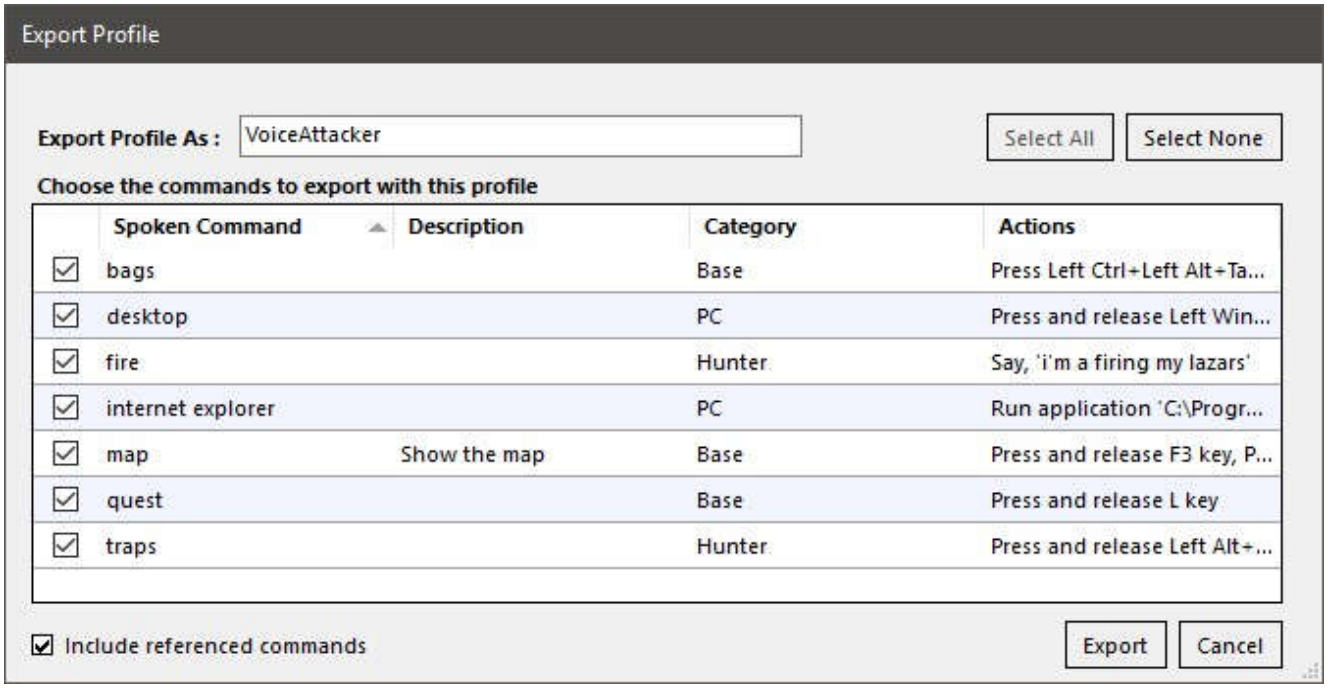
Use this feature to import your previously-saved settings file to restore your settings. Note that this will not affect registration details or profile data and VoiceAttack will close and need to be restarted after your settings are imported.
この機能を使用して、以前に保存した設定ファイルをインポートして、設定を復元します。これは登録の詳細やプロファイルデータに影響を与えず、VoiceAttackは閉じられ、設定のインポート後に再起動する必要があることに注意してください。

Browse VoiceAttack's Data Folder Link
VoiceAttackのデータフォルダリンクを参照

Click this link to browse the folder that holds VoiceAttack's central data file and backup folder. See the section labeled, 'Voice Attack's Data Storage' later in this document.
このリンクをクリックして、VoiceAttackの中央データファイルとバックアップフォルダーを保持するフォルダーを参照します。このドキュメントで後述する「音声攻撃のデータストレージ」というラベルの付いたセクションを参照してください。

Exporting Profiles
プロファイルのエクスポート

Exporting a VoiceAttack profile is very simple. Just click the 'Export' button on the main screen (See 'VoiceAttack's Main Screen') and you will be presented with the Export Profile screen below:
VoiceAttackプロファイルのエクスポートは非常に簡単です。メイン画面の「エクスポート」ボタンをクリックするだけで（「VoiceAttackのメイン画面」を参照）、以下の「プロファイルのエクスポート」画面が表示されます。



Next, you can give your exported profile a new name by typing it into the 'Export Profile
次に、エクスポートしたプロファイルに「エクスポートプロファイル」と入力して、新しい名前を付けることができます。

As' box. You will notice that all of the available commands are checked. If you do not want certain commands exported, simply deselect the boxes next to them. When you are ready to go, click on the 'Export' button, and, you will be presented with a 'Save Profile' dialog box, where you will have the choice of three different exported types. The first type, 'VoiceAttack Profile' is probably going to be your most-frequently used choice. This will export your profile to an XML-based file that you or anybody can later edit or examine using a text editor (like Notepad). The second item, and probably second-most frequently used will be, 'Quick Reference List as HTML'. This will export an HTML representation of your profile that you can view or print with a browser (like Internet Explorer or Chrome) (see the next section labeled, 'Creating Quick-Reference Lists' for more info on that). The third option, 'VoiceAttack Compressed Profile' will export your profile as a binary file which will be much smaller than the previously-mentioned XML option. The size will be a lot smaller, but you won't be able to edit the exported profile with a text editor (not a big deal for most, so even though this is the newest option, it may become the favorite due to the very small file size produced).
箱として。使用可能なすべてのコマンドがチェックされていることがわかります。特定のコマンドをエクスポートしたくない場合は、それらの隣のボックスを選択解除するだけです。準備ができたら、[エクスポート]ボタンをクリックすると、[プロファイルの保存]ダイアログボックスが表示され、3つの異なるエクスポートタイプを選択できます。最初のタイプの「VoiceAttack Profile」は、おそらく最も頻繁に使用される選択肢です。これにより、プロファイルがXMLベースのファイルにエクスポートされます。このファイルは、テキストエディター（メモ帳など）を使用して、後で編集または検証できます。2番目、おそらく2番目に頻繁に使用される項目は、「HTMLとしてのクイックリファレンスリスト」です。これにより、プロファイルのHTML表現がエクスポートされ、ブラウザ（Internet ExplorerやChromeなど）で表示または印刷できます（次のラベルのセクションを参照してください。詳細については、「クイックリファレンスリストの作成」を参照してください。3番目のオプション「VoiceAttack Compressed Profile」は、プロファイルをバイナリファイルとしてエクスポートします。これは、前述のXMLオプションよりもはるかに小さくなります。サイズははるかに小さくなりますが、エクスポートされたプロファイルをテキストエディターで編集することはできません（ほとんどの場合、大したことではないので、これは最新のオプションですが、生成される小さなファイルサイズ）。

The option, ‘Include referenced commands’ will allow you to include (by checking) or exclude (by unchecking) referenced commands with your export. A referenced command is a command that is either executed by an, ‘Execute Another Command’ action or is terminated by a ‘Stop Another Command’ action. Note: There are very few instances where you would want to exclude referenced commands. Note also that commands that
「参照コマンドを含める」オプションを使用すると、エクスポートに参照コマンドを含める（チェックする）か、除外する（チェックを外す）ことができます。参照されるコマンドは、「別のコマンドを実行」アクションで実行されるか、「別のコマンドを停止」アクションで終了するコマンドです。注：参照されているコマンドを除外したい場合はほとんどありません。また、そのコマンド

are included from other profiles will not be available to export from the current profile.
他のプロファイルから含まれているものは、現在のプロファイルからエクスポートすることはできません。

To export your profile to share or to make a backup, make sure that you have selected either 'VoiceAttack Profile' or ‘VoiceAttack Compressed Profile’ in the, 'Save as Type' box (see above for a description of each). Pick a good place to save and hit the, 'Save' button. All saved VoiceAttack profile files are given the '.vap' (VoiceAttack Profile) extension. To import this newly saved profile, see the 'Importing Profiles' section later in this document. To import individual commands from this exported profile, see, 'Importing Commands' (also later in this document).
プロファイルをエクスポートして共有またはバックアップを作成するには、[種類として保存]ボックスで[VoiceAttack Profile]または[VoiceAttack Compressed Profile]を選択したことを確認してください（それぞれの説明については上記を参照）。保存する適切な場所を選択し、「保存」ボタンを押します。保存されたすべてのVoiceAttackプロファイルファイルには、「。vap」（VoiceAttackプロファイル）拡張子が付けられます。この新しく保存されたプロファイルをインポートするには、このドキュメントで後述する「プロファイルのインポート」セクションを参照してください。このエクスポートされたプロファイルから個々のコマンドをインポートするには、「コマンドのインポート」を参照してください（このドキュメントでも後述）。

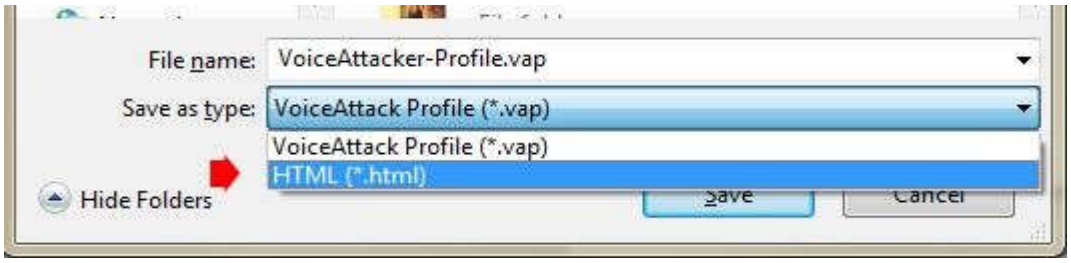
NOTE: The Export Profile functionality is only available for registered versions of VoiceAttack.
注：プロファイルのエクスポート機能は、VoiceAttackの登録済みバージョンでのみ使用できます。

Creating Quick-Reference Lists
クイックリファレンスリストの作成

Creating a Quick-Reference List like the one below uses the same steps as exporting a profile.
以下のようなクイックリファレンスリストを作成するには、プロファイルのエクスポートと同じ手順を使用します。

VoiceAttacker	VoiceAttack Profile Command Sheet
Spoken Commands	**Actions**
bags	Press and release SHIFT+B keys
desktop	Press and release Windows+D keys
fire	Say, 'i'm a firing my lazars'
focus one	Press and release CTRL+ALT+SHIFT+1 keys
focus three	Press and release CTRL+ALT+SHIFT+3 keys
focus two	Press and release CTRL+ALT+SHIFT+2 keys
map	Press and release M key
quest	Press and release L key

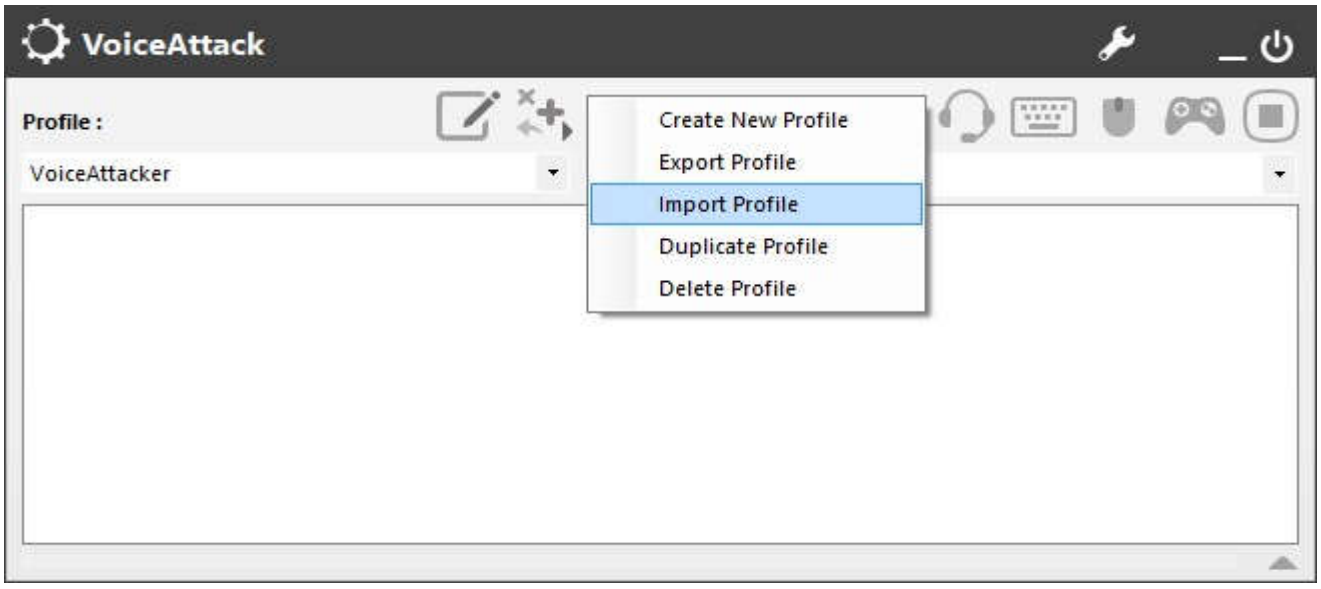
The only difference is that, instead of choosing to save your profile as a VoiceAttack Profile (.vap), you need to save your profile as an HTML file (see image below). The HTML file that is generated is viewable and printable in most compatible browsers.
唯一の違いは、プロファイルをVoiceAttackプロファイル(.vap)として保存する代わりに、プロファイルをHTMLファイルとして保存する必要があることです(下の画像を参照)。生成されるHTMLファイルは、ほとんどの互換性のあるブラウザーで表示および印刷できます。



NOTE: The Export Profile functionality (which includes creating quick reference lists) is only available to registered versions of VoiceAttack.
注:プロファイルのエクスポート機能(クイックリファレンスリストの作成を含む)は、VoiceAttackの登録済みバージョンでのみ使用できます。

Importing Profiles and Profile Packages
プロファイルとプロファイルパッケージのインポート

Importing profiles into VoiceAttack is even easier than exporting ('See Exporting Profiles'). Just go to the main screen, click on the multi-function icon and select, 'Import Profile' from the menu (or, press ALT + I):
VoiceAttackへのプロファイルのインポートは、エクスポートよりも簡単です(「プロファイルのエクスポート」を参照)。メイン画面に移動して、多機能アイコンをクリックし、メニューから「プロファイルのインポート」を選択します(または、ALT + Iを押します)。



When the 'Select a VoiceAttack Profile File To Import' dialog appears, simply browse and select your previously saved VoiceAttack profile file. A VoiceAttack profile file will have an extension of, '.vap'.
[インポートするVoiceAttackプロファイルファイルを選択]ダイアログが表示されたら、以前に保存したVoiceAttackプロファイルファイルを参照して選択します。VoiceAttackプロファイルファイルの拡張子は、'です。vap '。

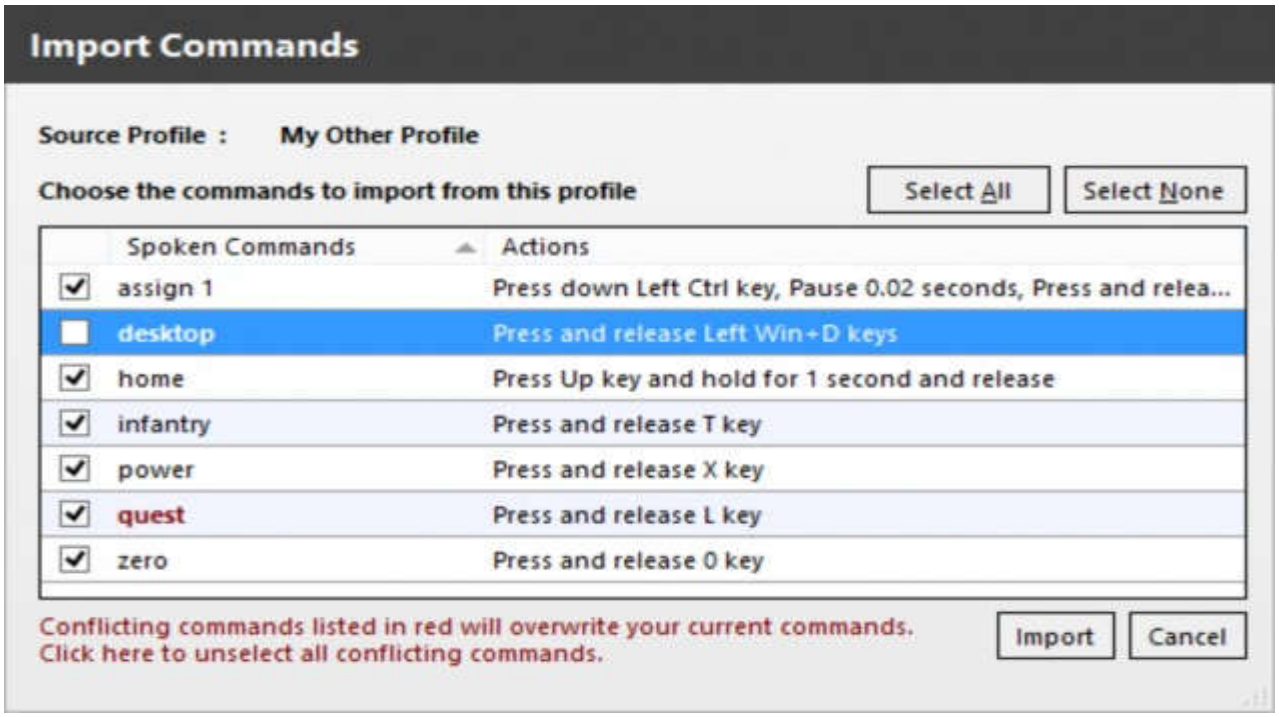
If the profile being imported has the same name as a profile that you already have, VoiceAttack will rename the new profile for you.
インポートするプロファイルの名前が既にあるプロファイルと同じ場合、VoiceAttackは新しいプロファイルの名前を変更します。

Another type of file that can be imported is the, 'VoiceAttack Profile Package' file type. These files have an extension of, '.vax'. A VoiceAttack Profile Package can contain multiple profiles, sound files, and plugin/application files. When a package file is imported, all of its contents are imported at one time (instead of having to manage multiple files).
インポートできる別のタイプのファイルは、「VoiceAttack Profile Package」ファイルタイプです。これらのファイルの拡張子は「」です。vax '。VoiceAttackプロファイルパッケージには、複数のプロファイル、サウンドファイル、およびプラグイン/アプリケーションファイルを含めることができます。パッケージファイルをインポートすると、そのコンテンツはすべて一度にインポートされます(複数のファイルを管理する必要はありません)。

Each profile it contains is imported (and renamed if duplicated as indicated above). Then, each sound and app/plugin will be copied to their respective locations in the folders that are specified on the Options screen (see, 'Sounds Folder' and, 'Apps Folder' portions of the, 'Options Screen' section earlier in this document). Note that importing a package file WILL OVERWRITE any existing sounds or app/plugin files that have the same path and file name, so make sure you know the source of your package and read their provided documentation on where the imported files will be placed. Since plugins/apps can be overwritten, the plugin feature must be disabled on the Options screen first, otherwise the import will not run (since files may be in use). For more information on VoiceAttack Profile Package files (as well as how to create them), see the section, 'VoiceAttack Profile Package Reference' later in this document. NOTE: The 'Import Profile' functionality is only available to registered versions of VoiceAttack.
含まれる各プロファイルがインポートされます(上記のように複製された場合は名前が変更されます)。その後、各サウンドとアプリ/プラグインは、オプション画面で指定されたフォルダー内のそれぞれの場所にコピーされます(この前の「オプション画面」セクションの「サウンドフォルダー」と「アプリフォルダー」の部分を参照)資料)。パッケージファイルをインポートすることに注意してください上書きされます、同じパスとファイル名を持つ既存の音やアプリ/プラグインファイルを、あなたはあなたのパッケージのソースを知っているし、インポートしたファイルが配置される場所に自分の説明書を読んで確認してください。プラグイン/アプリは上書きできるため、最初にオプション画面でプラグイン機能を無効にする必要がありますそうでない場合、インポートは実行されません(ファイルが使用されている可能性があるため)。VoiceAttackプロファイルパッケージファイルの詳細(および作成方法)については、このドキュメントで後述する「VoiceAttackプロファイルパッケージリファレンス」セクションを参照してください。注:「プロファイルのインポート」機能は、VoiceAttackの登録済みバージョンでのみ使用できます。

Importing Individual Commands
個々のコマンドのインポート

To import commands into a profile, first click on the 'Import Commands' button located at the bottom-left corner of the 'Add Profile' or 'Edit Profile' screens (See 'Profiles' section). You will then be presented with an open file dialog titled, 'Select a VoiceAttack Profile Containing Commands to Import'. Browse and select a previously saved VoiceAttack profile (VoiceAttack profile files have a, '.vap' extension). The, 'Import Commands' screen will then appear as below:
コマンドをプロファイルにインポートするには、最初に「プロファイルの追加」または「プロファイルの編集」画面の左下隅にある「コマンドのインポート」ボタンをクリックします(「プロファイル」セクションを参照)。次に、「インポートするコマンドを含むVoiceAttackプロファイルを選択してください」というタイトルのファイルを開くダイアログが表示されます。以前に保存したVoiceAttackプロファイルを参照して選択します(VoiceAttackプロファイルファイルの拡張子は「.vap」です)。[コマンドのインポート]画面が次のように表示されます。



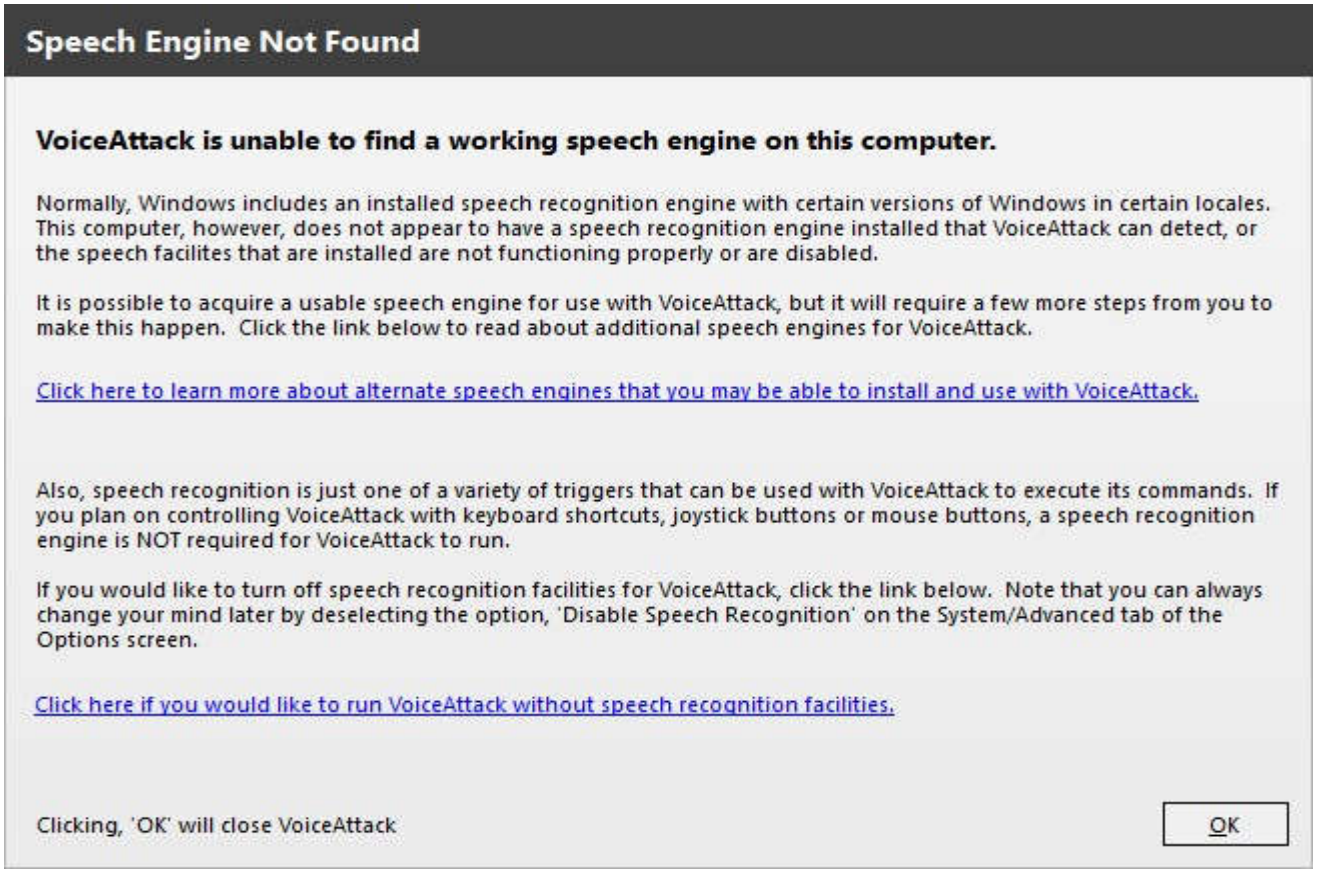
Only the commands that are checked will be imported into your profile. Commands that are in red are conflicting commands that already exist in your profile. Importing the conflicting commands will result in your profile's commands being overwritten. To clear all conflicting commands at once, simply click the gigantic label in the bottom-left corner :) When you are ready to import the selected commands, click on the, 'Import' button at the bottom-right of the screen. Remember, the commands that you import are not committed until you hit, the 'Done' button on the profile screen.
チェックされているコマンドのみがプロファイルにインポートされます。あるコマンドの赤は 、すでにあなたのプロファイルに存在するコマンドを矛盾しています。競合するコマンドをインポートすると、プロファイルのコマンドが上書きされます。競合するすべてのコマンドを一度にクリアするには、左下隅の巨大なラベルをクリックするだけです:)選択したコマンドをインポートする準備ができたら、画面右下の「インポート」ボタンをクリックします。インポートするコマンドは、プロファイル画面の「完了」ボタンを押すまでコミットされません。

NOTE: In the unregistered version of VoiceAttack, the single profile given is limited to 20 commands. If the profile ends up with more than 20 commands, only the first 20 will be shown (you may lose commands you have entered, since they may drop off).
注：VoiceAttackの未登録バージョンでは、指定された単一のプロファイルは20コマンドに制限されています。プロファイルが20を超えるコマンドで終わる場合、最初の20のみが表示されます（ドロップする可能性があるため、入力したコマンドは失われる可能性があります）。

No Speech Engine / Alternate Speech Engines in VoiceAttack
VoiceAttackの音声エンジンなし/代替音声エンジン

To be able to issue voice commands in VoiceAttack, a working speech engine is necessary. For most users with a Windows installation in English, German, French, Spanish, Chinese or Japanese locales, a speech engine in the appropriate language will be readily available. For others, Windows may not come with a speech engine installed, or their installation of Windows may default to an English speech engine.
VoiceAttackで音声コマンドを発行できるようにするには、機能する音声エンジンが必要です。英語、ドイツ語、フランス語、スペイン語、中国語、または日本語のロケールでWindowsをインストールしているほとんどのユーザーは、適切な言語の音声エンジンをすぐに利用できます。他の人にとっては、Windowsに音声エンジンがインストールされていない場合や、Windowsのインストールがデフォルトで英語の音声エンジンになっている場合があります。

If no speech engine is detected when VoiceAttack is launched (no speech engines installed on your computer or errors are encountered when asking Windows for its speech engines), VoiceAttack will present you with a message that indicates this. You will then have the option to run VoiceAttack without a speech engine, or, you can follow a link that will guide you on how to install alternate Microsoft (Microsoft Speech Platform) speech engines:
VoiceAttackの起動時に音声エンジンが検出されない場合（コンピューターに音声エンジンがインストールされていないか、音声エンジンをWindowsに要求するときにエラーが発生した場合）、VoiceAttackはこれを示すメッセージを表示します。その後、音声エンジンなしでVoiceAttackを実行するオプションがあります。または、代替のMicrosoft（Microsoft Speech Platform）音声エンジンをインストールする方法を案内するリンクをたどることができます。



If you would like to run VoiceAttack without a speech engine, simply click the link. VoiceAttack will start up as normal, but voice commands will be completely disabled (even if you have a working speech engine). In this mode, you can still execute VoiceAttack's commands via keyboard shortcuts, mouse button clicks or joystick button presses. This mode is also handy in situations where you just can't use voice commands, such as at work (which I use a LOT, as my coworkers won't miss a chance to tease… "KILL EXCEL!" lol). The next time
音声エンジンなしでVoiceAttackを実行する場合は、リンクをクリックするだけです。VoiceAttackは通常どおり起動しますが、音声コマンドは完全に無効になります（音声エンジンが動作している場合でも）。このモードでは、キーボードショートカット、マウスボタンのクリック、またはジョイスティックのボタンを押して、VoiceAttackのコマンドを実行できます。このモードは、職場などで音声コマンドを使用できない場合にも便利です（同僚がいじめる機会を逃さないように、LOTを使用しています… "KILL EXCEL！"笑）。次回

VoiceAttack launches, no check will be made for a speech engine, as the option is saved once you click the link. To turn speech recognition facilities back on again, simply deselect the, 'Disable Speech Recognition' check box on the, 'Options' screen (System/Advanced tab).
VoiceAttackが起動します。リンクをクリックするとオプションが保存されるため、音声エンジンのチェックは行われません。音声認識機能を再びオンにするには、[オプション]画面（[システム/詳細設定]タブ）の[音声認識を無効にする]チェックボックスをオフにします。

If you are without a speech engine (or want to try a new speech engine, either out of sheer need or even curiosity), the latest versions of VoiceAttack will allow the use of the Microsoft Speech Platform 11 speech engines. The link on the message will take you to the VoiceAttack site that will guide you on how to download the necessary files to make this happen. Simply follow the instructions provided on the site (which basically shows you how to download the runtime and one or more speech engines/text-to-speech synthesizers), and once VoiceAttack is restarted, the newly-installed speech engines and/or text-to-speech synthesizers will be available for use. If you have both the original (built-in) speech engine(s) and would like to toggle between the two speech platforms, there are options on the System/Advanced tab of the, 'Options' screen. Just choose the platform you wish to use (speech engines or text-to-speech synthesizers (or both)) and click, 'OK'. The next time VoiceAttack launches, you will be able to select from the speech engines or text-to-speech synthesizers from the selected platform (note that when you do change, the selected speech engine reverts to, 'System Default'). This is all new for VoiceAttack, so, once again, come and visit the VoiceAttack User Forums for more discussions on this (and, if there are none, start one!).
音声エンジンを使用していない場合（または必要性や好奇心から新しい音声エンジンを試したい場合）、VoiceAttackの最新バージョンではMicrosoft Speech Platform 11音声エンジンを使用できます。メッセージのリンクをクリックすると、VoiceAttackサイトが表示され、これを実現するために必要なファイルをダウンロードする方法が案内されます。サイトで提供される指示（基本的に、ランタイムと1つ以上の音声エンジン/音声合成シンセサイザーをダウンロードする方法を示します）に従ってください。VoiceAttackを再起動すると、新しくインストールされた音声エンジンおよび/またはテキスト音声合成シンセサイザーを使用できます。元の（組み込みの）音声エンジンの両方があり、2つの音声プラットフォームを切り替えたい場合は、[オプション]画面の[システム/詳細設定]タブにオプションがあります。使用するプラットフォーム（音声エンジンまたは音声合成シンセサイザー（またはその両方））を選択して、[OK]をクリックします。次回VoiceAttackを起動すると、選択したプラットフォームの音声エンジンまたは音声合成シンセサイザーから選択できます（変更すると、選択した音声エンジンは「システムデフォルト」に戻ります）。これはVoiceAttackのすべての新機能です。もう一度、VoiceAttackユーザーフォーラムにアクセスして、これについての詳細なディスカッションを行ってください（もしなければ、開始してください！）。選択した音声エンジンは「システムのデフォルト」に戻ります）。これはVoiceAttackのすべての新機能です。もう一度、VoiceAttackユーザーフォーラムにアクセスして、これについての詳細なディスカッションを行ってください（もしなければ、開始してください！）。選択した音声エンジンは「システムのデフォルト」に戻ります）。これはVoiceAttackのすべての新機能です。もう一度、VoiceAttackユーザーフォーラムにアクセスして、これについての詳細なディスカッションを行ってください（もしなければ、開始してください！）。

Using the Condition Builder
条件ビルダーの使用

The Condition Builder screen allows you to create compound conditions. Compound conditions contain multiple, single-test conditions that are strung together by, 'AND' or 'OR'.
条件ビルダー画面では、複合条件を作成できます。複合条件には、「AND」または「OR」で区切られた複数の単一テスト条件が含まれます。

For instance, a single-test condition (like VoiceAttack has used for some time) looks like this:
たとえば、単一のテスト条件（VoiceAttackがしばらく使用しているような）は次のようになります。

If myVariable Equals 1 Then Do Something
myVariableが1に等しい場合

End If
終了する場合

This works great, but if you want to check if myVariable is 1 OR 2 OR 3 in VoiceAttack, you would have to do something like this:
これはうまく機能しますが、VoiceAttackでmyVariable が1 または 2 または 3であるかどうかを確認する場合は、次のようにする必要があります。

If myVariable Equals 1 Then Do This
myVariableが1に等しい場合

Do That Do Other
他のことをする

Else If myVariable Equals 2 Then Do This
それ以外の場合、myVariableが2に等しい場合

Do That Do Other
他のことをする

Else If myVariable Equals 3 Then Do This
それ以外の場合myVariableが3に等しい場合

Do That Do Other
他のことをする

End If
終了する場合

Note the duplicated script... there are ways to make this shorter, but that's another subject :)
複製されたスクリプトに注意してください...これを短くする方法がありますが、それは別の主題です:)

Additionally, if you wanted to check if myVariable was 15 OR myVariable was between 1 and 10, but did not equal 6, you would need to do something like this:
あなたはmyVariable変数が15だったかどうかを確認したい場合はさらに、OR myVariable変数は1と10の間であったが、6と等しくなかった、あなたはこのような何かをする必要があります:

If myVariable Equals 15 Then Do This
myVariableが15に等しい場合

Do That Do Other
他のことをする

Else
その他

If myVariable is Greater than or Equals 1 Then If myVariable is Less than or Equals 10 Then
myVariableが1以上の場合myVariableが10以下の場合

If myVariable Does Not Equal 6 Then Do This
myVariableが6に等しくない場合、これを行う

Do That Do Other
他のことをする

End If End If
End If End If

End If
終了する場合

End If
終了する場合

As you can see, things can get a little wordy, and these are actually just small examples without a lot going on. It would be better to have scripts that read like this:
ご覧のとおり、物事は少し冗長になる可能性がありますが、これらは実際には多くのことをせずにほんの小さな例です。次のようなスクリプトを作成することをお勧めします。

If myVariable Equals 1 OR myVariable Equals 2 OR myVariable Equals 3 Then Do This
myVariableが1に等しい、またはmyVariableが2に等しい、またはmyVariableが3に等しい場合

Do That Do Other
他のことをする

End If
終了する場合

And
そして

If (myVariable Equals 15) OR (myVariable >=1 AND myVariable <= 10 AND myVariable Does Not Equal 6) Then
If(myVariable Equals 15)OR(myVariable> = 1 AND myVariable <= 10 AND myVariableが等しくない6)Then

Do This Do That Do Other
他のことをする

End If
終了する場合

To build these kinds of statements in a VoiceAttack action/graphical fashion requires the use of the Condition Builder. The Condition Builder looks scary, but once you get an idea of what it's doing, you'll get the hang of it quickly. To understand what is going on, let's start with some terminology. My apologies in advance to any programmer types out there ;)
VoiceAttackアクション/グラフィカルな方法でこれらの種類のステートメントを作成するには、条件ビルダーを使用する必要があります。条件ビルダーは恐ろしいように見えますが、それが何をしているのかがわかれば、すぐに理解できるようになります。何が起こっているのかを理解するために、いくつかの用語から始めましょう。プログラマーがそこにタイプするのを前もっておaびします;)

First is the term, 'condition'. In programming, this is often referred to as an, 'If' statement. In VoiceAttack, you'll see conditions are used for, 'If' statements, 'Else If' statements, and 'While' loops:
最初は「条件」という用語です。プログラミングでは、これはしばしば「If」ステートメントと呼ばれます。VoiceAttackでは、「If」ステートメント、「Else If」ステートメント、および「While」ループに条件が使用されていることがわかります。

If myVariable Equals 5 Then Do Something
myVariableが5に等しい場合

Else If myVariable Equals 6 Then Do Something Else
Else If myVariable Equals 6 Then Do Something Else

End If
終了する場合

Start Loop While: myVariable Is Less Than 10 Do Stuff
ループの開始：myVariableが10未満の場合

myVariable = myVariable - 1 End Loop
myVariable = myVariable-1終了ループ

A, 'compound condition' is an expression that contains more than one condition. The conditions are strung together by, 'AND' or, 'OR' and consist of logical 'sets':
A、「複合条件」は、複数の条件を含む式です。条件は、「AND」または「OR」によって結び付けられ、論理「セット」で構成されます。

If (myVariable Equals 15) OR (myVariable >=1 AND myVariable <= 10 AND myVariable Does Not Equal 6) Then
If(myVariable Equals 15)OR(myVariable> = 1 AND myVariable <= 10 AND myVariableが等しくない6)Then

Do Something End If
何かを終わらせる

The first logical set above is 'myVariable Equals 15'. If the myVariable integer variable is 15, the entire expression is evaluated as true and the control moves to the next line.
上記の最初の論理セットは「myVariable Equals 15」です。myVariable整数変数が15の場合、式全体がtrueと評価され、コントロールは次の行に移動します。

If myVariable' value is 6, the first set does not evaluate to true, so the second logical set '(myVariable >=1 AND myVariable <= 10 AND myVariable Does Not Equal 6)' is evaluated. The second set also does not evaluate to true, which means the compound condition evaluates to false and control moves past the end of the compound condition (past the, 'End If').
myVariable 'の値が6の場合、最初のセットはtrueと評価されないため、2番目の論理セット'(myVariable> = 1 AND myVariable <= 10 AND myVariable Does Not Equal 6) 'が評価されます。2番目のセットもtrueと評価されません。これは、複合条件がfalseと評価され、複合条件の終わりを過ぎて制御が移動することを意味します(過去の「End If」)。

Again, in VoiceAttack, you can build these, 'sets' using the Condition Builder. The expressions resulting from the Condition Builder can be one or more logical sets. Each set is evaluated within the expression. If all the conditions in a set evaluate to true, the set is true. If any of the conditions in a set evaluate to false, the whole set evaluates to false. If ANY set evaluates to true, the entire expression evaluates to true. Another way to state this is everything WITHIN a set is, 'AND-ed' and all sets are, 'OR-ed':
繰り返しますが、VoiceAttackでは、条件ビルダを使用してこれらの「セット」を構築できます。条件ビルダーの結果の式は、1つ以上の論理セットにすることができます。各セットは、式内で評価されます。セット内のすべての条件がtrueと評価される場合、セットはtrueです。セット内の条件のいずれかがfalseと評価された場合、セット全体がfalseと評価されます。 ANYセットがtrueと評価されると、式全体がtrueと評価されます。これを示す別の方法は、セット内のすべてが「AND」され、すべてのセットが「OR」されていることです。

Set 1
セット1

myVariable Equals 15
myVariable Equals 15

OR ß
またはß

Set 2 myVariable >=1 AND ß
2設定してmyVariable> = 1 AND ßを

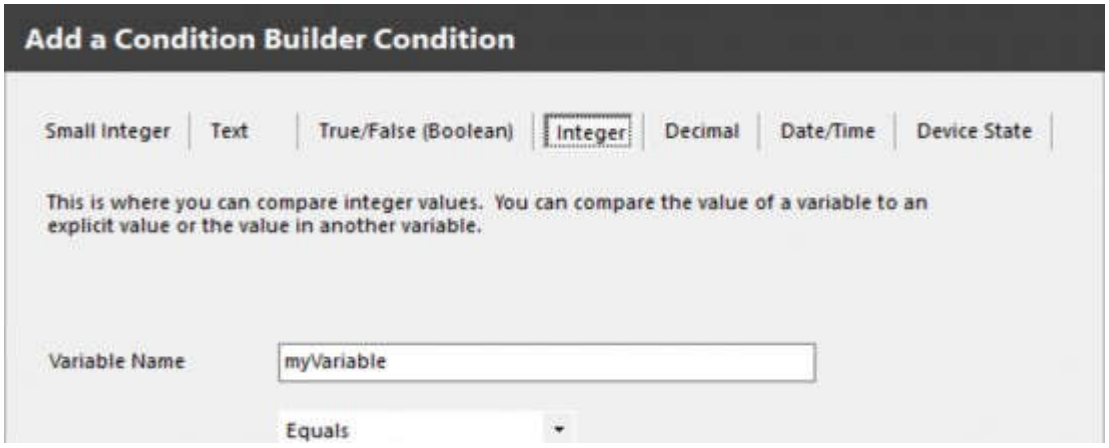
myVariable <= 10
myVariable <= 10

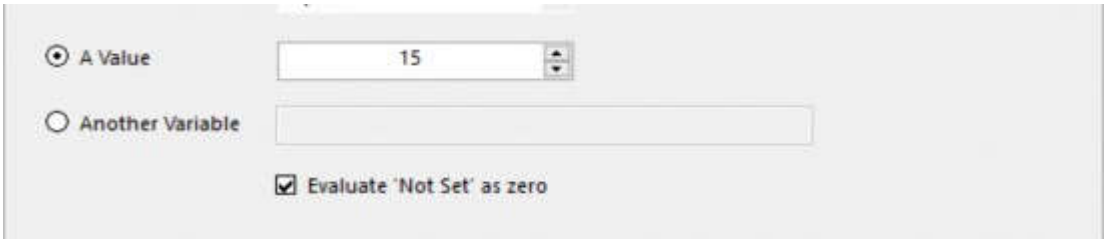
AND ß
AND ß

myVariable Does Not Equal 6
myVariableは等しくない6

The best way to explain how this is done in VoiceAttack is to just jump right in:
VoiceAttackでこれがどのように行われるかを説明する最良の方法は、次のコードに直接ジャンプすることです。

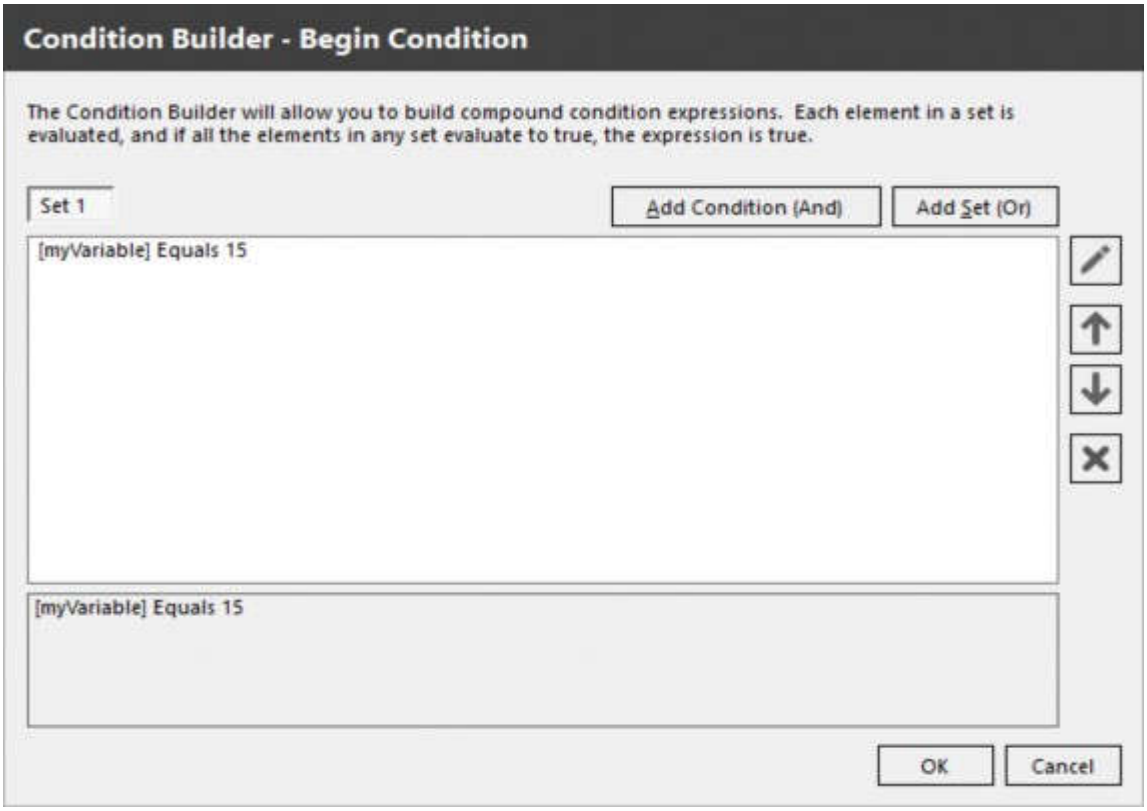
First, from the Command screen, select, Other > Advanced > Begin a Conditional (If Statement) Block > Compound Condition Builder. You will be presented with a familiar screen to create the first condition. Click on the, 'Integer' tab and then enter the values as you would to create the condition, 'myVariable Equals 15':
まず、コマンド画面から、[その他]> [詳細]> [条件付き(Ifステートメント)ブロックの開始]> [複合条件ビルダー]を選択します。最初の条件を作成するための使い慣れた画面が表示されます。「整数」タブをクリックし、条件「myVariable Equals 15」を作成する場合と同じように値を入力します。





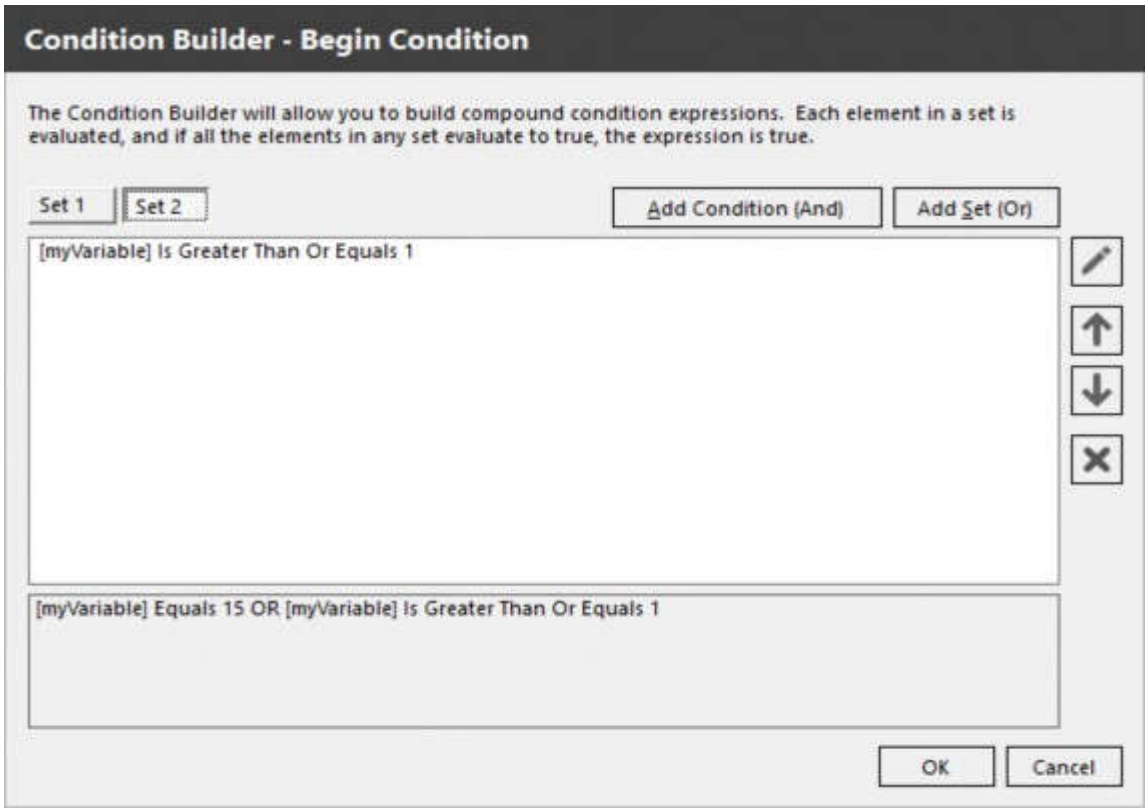
Click OK, and you'll notice that the first set (Set 1) is created for you with the condition, 'myVariable Equals 15' as the first item. Note the narrative at the bottom of the Condition
[OK]をクリックすると、最初の項目として「myVariable Equals 15」という条件で最初のセット（セット1）が作成されていることがわかります。条件の下部にある物語に注意してください

Builder screen indicates the current state of the entire expression:
ビルダー画面には、式全体の現在の状態が表示されます。



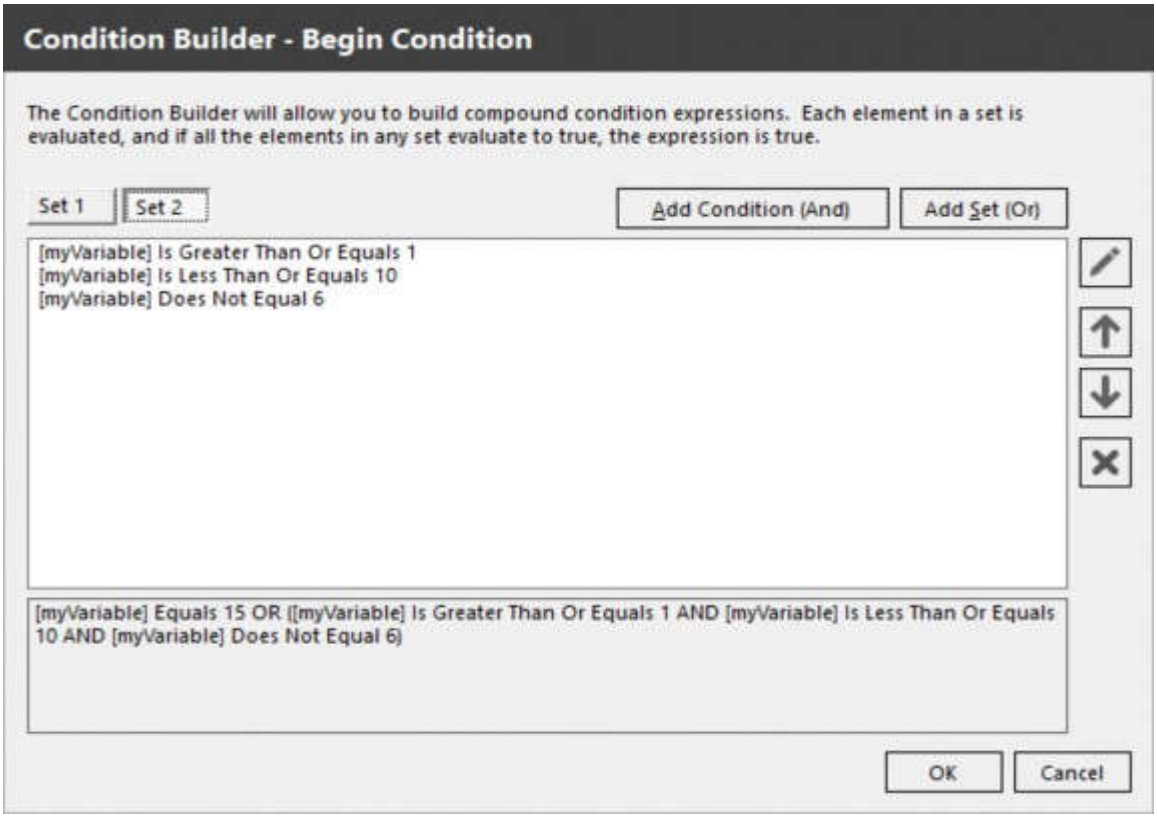
That's all there is for the first set... we're just checking first to see if myVariable is 15.
最初のセットについてはこれですべてです... myVariableが15であるかどうかを確認するために最初にチェックしています。

Next, we'll want to create the second, 'Set'. Click on the, 'Add Set (Or)' button. Again, you'll get an, 'Add a condition' screen popped up to add the first condition of the new set. Click on the, 'Integer' tab and, again, enter the values as you would to create the condition, 'myVariable is greater than or equal to 1' and click, 'OK'. Note that you are now in, 'Set 2', with 'myVariable is Greater than or Equals 1' as the first condition on the Condition Builder screen. If you look at the narrative at the bottom of the screen, you should see, 'myVariable Equals 15 OR myVariable is Greater Than or Equals 1'. Good so far:
次に、2番目の「セット」を作成します。[追加セット（または）]ボタンをクリックします。繰り返しますが、新しい条件の最初の条件を追加するための「条件の追加」画面がポップアップ表示されます。「整数」タブをクリックし、条件を作成する場合と同じように値を入力します。「myVariableは1以上」で、「OK」をクリックします。条件ビルダー画面の最初の条件として「myVariableが1以上」の「セット2」にいることに注意してください。画面の下部にある物語を見ると、「myVariable Equals 15 OR myVariable is Greater Than or Equals 1」が表示されます。これまでのところ：



We still have two more conditions to add to this set, so, this time click on the, 'Add Condition (And)' button. You'll be presented again with the, 'Add a condition' screen. Click on the integer tab and enter the values to create the condition, 'myVariable is less than or equal to 10'. Click, 'OK'. A second condition is now added to, 'Set 2': 'myVariable is Less Than or Equals 10', with the narrative now showing, 'myVariable Equals 15 OR (myVariable is Greater Than or Equals 1 AND myVariable is Less Than or Equals 10)'. One more to go...
このセットに追加する条件がまだ2つあるので、今回は[条件の追加（および）]ボタンをクリックします。「条件の追加」画面が再び表示されます。整数タブをクリックし、値を入力して「myVariable is 10以下」という条件を作成します。「OK」をクリックします。「セット2」に2番目の条件が追加されました：「myVariable is Less Than or Equals 10」、ナラティブは「myVariable Equals 15 OR（myVariable is Greater Than or Equals 1 AND myVariable is Less Than or Equals 10」 ） '。もう1つ...

Click again on the, 'Add Condition (And)' button. Again, enter the values to create the condition, 'myVariable Does Not Equal 6' and click, 'OK'. Now, you have three conditions in Set 2, with the narrative now reading, 'myVariable Equals 15 OR (myVariable is Greater Than or Equals 1 AND myVariable is Less Than or Equals 10 AND myVariable Does Not Equal 6)':
[条件の追加（および）]ボタンをもう一度クリックします。再度、値を入力して条件を作成し、「myVariable Does Not Equal 6」をクリックして、「OK」をクリックします。今、セット2には3つの条件があり、物語は「myVariable Equals 15 OR（myVariable is Greater Than or Equals 1 AND myVariable is Less Than or Equals 10 AND myVariable Does Not Equal 6）」と読みます：



Click, 'OK' on the Condition Builder screen and you'll see your new conditional (if) statement presented in the Command Screen.
[条件ビルダ]画面で[OK]をクリックすると、コマンド画面に新しい条件(if)ステートメントが表示されます。

Congratulations! You just built your first compound condition. Hope that didn't scare you off :)
おめでとうございます！最初の複合条件を作成しました。それがあなたを怖がらせなかったことを願っています:)

Note that you don't have to build compound conditions with all the same data type (in this example we are using all integers). You can have compound conditions based any number of mixed data types.
すべて同じデータ型の複合条件を作成する必要はありません(この例では、すべて整数を使用しています)。任意の数の混合データ型に基づいて複合条件を設定できます。

Note also that you can convert a single-condition conditional statement (you know… the ones you've been using all this time) to compound conditions. Just right-click on the action you want to convert and select the, 'Edit with Condition Builder' option.
また、単一条件の条件ステートメント(これまでずっと使用してきたもの)を複合条件に変換できることに注意してください。変換するアクションを右クリックして、「条件ビルダーで編集」オプションを選択します。

Command Line Options
コマンドラインオプション

VoiceAttack supports the following command line options (in basically the order in which they were added in case you're wondering):
VoiceAttackは、次のコマンドラインオプションをサポートしています（基本的には、疑問に思われる場合に追加された順序です）。

-listeningoff Turns VoiceAttack's listening off.
-listeningoff VoiceAttackのリスニングをオフにします。

-listeningon Turns VoiceAttack's listening on.
-listeningon VoiceAttackのリスニングをオンにします。

-shortcutson Turns VoiceAttack's hotkey shortcuts on.
-shortcutson VoiceAttackのホットキーショートカットをオンにします。

-shortcutsoff Turns VoiceAttack's hotkey shortcuts off.
-shortcutsoff VoiceAttackのホットキーショートカットをオフにします。

-mouseon Turns VoiceAttack's mouse shortcuts on.
-mouseon VoiceAttackのマウスショートカットをオンにします。

-mouseoff Turns VoiceAttack's mouse shortcuts off.
-mouseoff VoiceAttackのマウスショートカットをオフにします。

-joystickson Turns VoiceAttack's joystick button shortcuts on.
-joystickson VoiceAttackのジョイスティックボタンのショートカットをオンにします。

-joysticksoff Turns VoiceAttack's joystick button shortcuts off.
-joysticksoff VoiceAttackのジョイスティックボタンのショートカットをオフにします。

-minimize Starts VoiceAttack minimized.
-minimize VoiceAttackを最小化して起動します。

-nofocus Prevents VoiceAttack from popping up an already-running instance (if this is not specified, VoiceAttack pops up as the topmost window).
-nofocus VoiceAttackが既に実行中のインスタンスをポップアップするのを防ぎます（これが指定されていない場合、VoiceAttackは一番上のウィンドウとしてポップアップします）。

-profile “My Profile Name” Changes VoiceAttack's active profile. Note that double quotes are only necessary if your profile name has spaces in it.
-profile「My Profile Name」は、 VoiceAttackのアクティブなプロファイルを変更します。二重引用符は、プロファイル名にスペースが含まれている場合にのみ必要であることに注意してください。

-command “My Command Name” Executes the command specified by name in the running profile. Note that double quotes are only necessary if your command name has spaces in it.
-command “ My Command Name”実行中のプロファイルで名前で指定されたコマンドを実行します。二重引用符は、コマンド名にスペースが含まれている場合にのみ必要であることに注意してください。

-stopcommands This will work just like clicking the, 'stop commands' button.
-stopcommands これは、「停止コマンド」ボタンをクリックするのと同じように機能します。

-bypassPendingSpeech If VoiceAttack's 'not listening' mode is invoked, and you
-bypassPendingSpeech VoiceAttackの「非リスニング」モードが呼び出された場合、および

are in the middle of issuing a command, VoiceAttack will allow you to finish your phrase and not cut off what you were saying. For some, this behavior is not what is expected. What is expected is that setting VoiceAttack into, 'not listening' mode should cut off immediately, excluding anything that is currently being spoken. To allow VoiceAttack to cut you off immediately, use -bypassPendingSpeech. Note this does require VoiceAttack to be restarted (does not affect an already-running instance).
コマンドを発行している最中に、VoiceAttackを使用すると、発言を中断せずにフレーズを終了できます。一部の人にとって、この動作は予期されたものではありません。予想されるのは、VoiceAttackを「非リスニング」モードに設定すると、現在話されているものをすべて除外して、すぐに切断されることです。VoiceAttackですぐに切断できるようにするには、-bypassPendingSpeechを使用します。これにはVoiceAttackを再起動する必要があることに注意してください（既に実行中のインスタンスには影響しません）。

-ignorechildwindows This is to help with toolbox windows that are always on top. By default, VoiceAttack will seek out popup windows. This attempts to suppress that check. This is experimental right now and may become a feature later with a finer level of control if the need is there. For now, it is available as a global setting.
-ignorechildwindows これは、常に一番上にあるツールボックスウィンドウを支援するためです。デフォルトでは、VoiceAttackはポップアップウィンドウを探します。これは、そのチェックを抑制しようとします。これは現在実験段階であり、必要に応じて後でより細かな制御レベルを備えた機能になる可能性があります。現時点では、グローバル設定として使用できます。

-verifyaudio This will make VoiceAttack check all ‘play a sound’ and ‘play a random’ sound files to see if they exist. This just runs at startup and does not affect the already-running instance.
-verifyaudio これにより、VoiceAttackはすべての「サウンドを再生」および「ランダムに再生」サウンドファイルをチェックして、存在するかどうかを確認します。これは起動時に実行されるだけで、すでに実行中のインスタンスには影響しません。

-datadir “Directory Path” Allows you to indicate the directory of VoiceAttack’s data file
-datadir “ Directory Path” VoiceAttackのデータファイルのディレクトリを指定できます。

(VoiceAttack.dat) and its associated backups. If the VoiceAttack.dat file does not exist in the indicated directory when a profile has been updated, a new VoiceAttack.dat will be created. Note that double quotes are only necessary if your path has spaces in it. This does not affect the running instance.
（VoiceAttack.dat）および関連するバックアップ。プロファイルが更新されたときに、指定されたディレクトリにVoiceAttack.datファイルが存在しない場合、新しいVoiceAttack.datが作成されます。二重引用符は、パスにスペースが含まれている場合にのみ必要であることに注意してください。これは、実行中のインスタンスには影響しません。

Example: -datadir “C:¥Users¥MrAwesome¥Desktop¥MyData” will look for the VoiceAttack.dat file in the “C:¥Users¥MrAwesome¥Desktop¥MyData” directory.
例：-datadir「C：¥ Users ¥ MrAwesome ¥ Desktop ¥ MyData」は、「C：¥ Users ¥ MrAwesome ¥ Desktop ¥ MyData」ディレクトリでVoiceAttack.datファイルを探します。

-alwaysontopon This will set the running instance of VoiceAttack to be the top-most application.
-alwaysontopon これにより、VoiceAttackの実行中のインスタンスが最上位のアプリケーションに設定されます。

-alwaysontopoff This will turn off, ‘Always on Top’ if it is on, returning VoiceAttack to not be the top-most application.
-alwaysontopoff これはオフになり、「Always on Top」がオンの場合、VoiceAttackが最上位のアプリケーションではないことを返します。

-showloadtime This will show the amount of time it takes to load a profile in the log. The time taken by the speech engine is shown as well. Both times are in milliseconds. This is useful for the folks making enormous profiles to give some sort of indication of what is going on.
-showloadtime これは、ログにプロファイルをロードするのにかかる時間を表示します。音声エンジンにかかった時間も表示されます。両方の時間はミリ秒単位です。これは、巨大なプロファイルを作成している人が、何が起こっているのかを示すのに役立ちます。

-nospeech This disables VoiceAttack speech recognition initialization. This is experimental and functionally incomplete and is used primarily for testing (that is, it is not currently supported). However, this may be a last resort for those that just cannot get their speech engine to work and need to get into the software for whatever reason.
-nospeech これにより、VoiceAttack音声認識の初期化が無効になります。これは実験的で機能的に不完全であり、主にテストに使用されます（つまり、現在サポートされていません）。ただし、これは、音声エンジンを機能させることができず、何らかの理由でソフトウェアにアクセスする必要がある人にとっては最後の手段かもしれません。

-input “Device Name” This will set the Windows default multimedia recording device (table mics and headset mics for example) and set the speech engine to this device. The device name parameter value (between the required double quotes) must be spelled exactly as it is shown in the list on the VoiceAttack Options screen (see the Options screen for more information on where to find this). Sometimes with certain events (like a driver update) the name may change. This change may even be very slight (and frustrating). In order to ease that pain a bit, the device name parameter will accept wildcards. For example, this shows how to change to a device exactly as indicated:
-input “ Device Name” これは、Windowsのデフォルトのマルチメディア録音デバイス（テーブルマイクやヘッドセットマイクなど）を設定し、このデバイスに音声エンジンを設定します。デバイス名パラメーター値（必要な二重引用符の間）は、VoiceAttackオプション画面のリストに表示されているとおりに正確に入力する必要があります（この場所の詳細については、オプション画面を参照してください）。特定のイベント（ドライバーの更新など）で名前が変わる場合があります。この変更は、ごくわずかなものでさえあります（そしてイライラさせられます）。その痛みを少し緩和するために、デバイス名パラメーターはワイルドカードを受け入れます。たとえば、これは示されているとおりにデバイスに変更する方法を示しています。

C:¥Program Files(x86)¥VoiceAttack¥VoiceAttack.exe -input "Speakers (4- Sennheiser 3D G4ME1)"
C：¥ Program Files（x86）¥ VoiceAttack ¥ VoiceAttack.exe -input "Speakers（4- Sennheiser 3D G4ME1）"

This example shows how to select the first device that has a description that contains, 'G4ME1':
この例は、「G4ME1」を含む説明を持つ最初のデバイスを選択する方法を示しています。

C:¥Program Files(x86)¥VoiceAttack¥VoiceAttack.exe -input "*G4ME1*"
C：¥ Program Files（x86）¥ VoiceAttack ¥ VoiceAttack.exe -input "* G4ME1 *"

Search this help document for, ‘wildcards’ for more examples of how they are used. Note: Double quotes are required for this command line parameter. Also, this parameter will work against new or existing instances of VoiceAttack.
これらの使用方法の例については、このヘルプドキュメントで「ワイルドカード」を検索してください。注：このコマンドラインパラメーターには二重引用符が必要です。また、このパラメーターはVoiceAttackの新規または既存のインスタンスに対して機能します。

WARNING: This is not a VoiceAttack setting, rather a Windows device setting and can (and probably will) cause other applications that depend on the changed devices to appear to malfunction. It’s not THAT big of a deal, but it will definitely throw you off when your Skype or TeamSpeak is not working how you left them.
警告：これはVoiceAttackの設定ではなく、Windowsデバイスの設定であり、変更されたデバイスに依存する他のアプリケーションが誤動作するように見える可能性があります。それほど大したことではありませんが、SkypeまたはTeamSpeakがあなたがそれらを残したように機能していない場合、間違いなくあなたを失望させます。

-output “Device Name” This works exactly like the -input parameter above except it controls the default multimedia output device (like speakers and headphones). Same notes and warning apply as well ;)
-output “ Device Name” これは、デフォルトのマルチメディア出力デバイス（スピーカーやヘッドフォンなど）を制御することを除き、上記の-inputパラメーターとまったく同じように機能します。同じメモと警告も適用されます;)

-inputcomms “Device Name” This works exactly like -input (above), except this option will affect Windows’ default communications recording device.
-inputcomms “ Device Name” これは-input（上記）とまったく同じように機能しますが、このオプションはWindowsのデフォルトの通信記録デバイスに影響します。

-outputcomms “Device Name” Again, this works exactly like the parameters above except it controls the default output communications device (like speakers and headphones).
-outputcomms “ Device Name” ここでも、デフォルトの出力通信デバイス（スピーカーやヘッドフォンなど）を制御することを除いて、上記のパラメーターとまったく同じように機能します。

-inputx “Device Name”, -inputcommsx “Device Name”, -outputx “Device Name”, and
-inputx「デバイス名」、 -inputcommsx「デバイス名」、 -outputx「デバイス名」、および

-outputcommsx “Device Name” These work exactly like -input, -inputcomms, -output, and
- outputcommsx「デバイス名」、これらを正確に-input、-inputcomms、-outputのような作品、および

-outputcomms except that once the devices are changed, VoiceAttack closes immediately. This means you can make a desktop shortcut or batch file that can change your headphones to your speakers or your tabletop mic to your headset mic (or both at the same time) without fully launching VoiceAttack.
-outputcomms。ただし、デバイスが変更されると、VoiceAttackはすぐに閉じます。つまり、VoiceAttackを完全に起動することなく、ヘッドフォンをスピーカーに、またはテーブルトップマイクをヘッドセットマイクに（または両方同時に）変更できるデスクトップショートカットまたはバッチファイルを作成できます。

-nokeyboard This keeps all keyboard hooks from turning on when VoiceAttack launches. That means that global hotkeys (such as listening, stop commands, mouse capture) and command shortcuts will be disabled. Note this affects only the instance launched with this parameter and not an instance of VoiceAttack already running.
-nokeyboard これにより、VoiceAttackの起動時にすべてのキーボードフックがオンになりません。つまり、グローバルホットキー（リスニング、停止コマンド、マウスキャプチャなど）とコマンドショートカットが無効になります。これは、このパラメーターで起動されたインスタンスのみに影響し、既に実行されているVoiceAttackのインスタンスには影響しないことに注意してください。

-nomouse This keeps all mouse hooks from turning on when VoiceAttack launches. That means that global mouse clicks (such as listening) and command shortcuts will be disabled. Note this affects only the instance launched with this parameter and not an instance of VoiceAttack already running.
-nomouse これにより、VoiceAttackの起動時にすべてのマウスフックがオンになりません。つまり、グローバルなマウスクリック（リスニングなど）とコマンドショートカットは無効になります。これは、このパラメーターで起動されたインスタンスのみに影響し、既に実行されているVoiceAttackのインスタンスには影響しないことに注意してください。

-uritimeout Timeout This indicates the timeout to use when setting text variables via the URI option. This is a stopgap feature until proper user interface is created and may not be in future versions of VoiceAttack. The timeout parameter is expressed in milliseconds:
-uritimeout Timeout これは、URIオプションを介してテキスト変数を設定するときに使用するタイムアウトを示します。これは、適切なユーザーインターフェイスが作成されるまでの一時的な機能であり、VoiceAttackの将来のバージョンには含まれない可能性があります。タイムアウトパラメータはミリ秒単位で表されます。

-uritimeout 5000 sets the timeout to 5 seconds. The default value for the timeout that VoiceAttack uses is 30 seconds. Note this affects only the instance launched with this parameter and not an instance of VoiceAttack already running.
-uritimeout 5000は、タイムアウトを5秒に設定します。VoiceAttackが使用するタイムアウトのデフォルト値は30秒です。これは、このパラメーターで起動されたインスタンスのみに影響し、既に実行されているVoiceAttackのインスタンスには影響しないことに注意してください。

-reverseprofilepriority This option will reverse the priority of, 'included' profile commands. By default, when you include commands from other profiles (either from the profile options screen and/or by selecting, 'Global Profiles' from the Options screen), the commands are prioritized first by the active profile, then by the included profiles from the profile options screen, then by the global profiles. With this option, the priority is first the global profiles, then the included profiles from the profile options screen, then the active profile. This is an advanced feature that not many will use, but it is here for those that need it. This only affects the instance launched with VoiceAttack and not an instance that is already running.
-reverseprofilepriority このオプションは、「含まれる」プロファイルコマンドの優先順位を逆にします。デフォルトでは、他のプロファイルからのコマンドを含めると（プロファイルオプション画面から、および/またはオプション画面から「グローバルプロファイル」を選択することにより）、コマンドは最初にアクティブなプロファイル、次に含まれるプロファイルから優先順位が付けられますプロファイルオプション画面、次にグローバルプロファイル。このオプションでは、優先度は最初にグローバルプロファイル、次にプロファイルオプション画面に含まれるプロファイル、次にアクティブなプロファイルです。これは多くの人が使用しない高度な機能ですが、必要な人のためにここにあります。これはVoiceAttackで起動されたインスタンスにのみ影響し、既に実行されているインスタンスには影響しません。

-running This option will cause VoiceAttack to write 0 to stdout if no other instance of VoiceAttack is currently running, and 1 if an instance of VoiceAttack is running and then immediately close. Obviously, this does not affect the running instance ;)
-running このオプションにより、VoiceAttackのインスタンスが現在実行されていない場合はVoiceAttackが0をstdoutに書き込み、VoiceAttackのインスタンスが実行されている場合は1を書き込み、すぐに閉じます。明らかに、これは実行中のインスタンスには影響しません;)

-darkon -darkoff These options will turn on and off the dark mode setting for the main screen ('Cover of Darkness'). Note that these only affect the launched instance.
-darkon -darkoff これらのオプションは、メイン画面のダークモード設定（ 'Cover of Darkness'）をオンまたはオフにします。これらは起動されたインスタンスにのみ影響することに注意してください。

-showcommandnames This option will make the VA log entries display the current command name when a command is executing (for debug purposes).
-showcommandnames このオプションは、コマンドの実行時にVAログエントリに現在のコマンド名を表示します（デバッグ目的）。

-asadmin Including this command line parameter will make VoiceAttack attempt to run itself as an administrator for the current session. This does not affect an instance that is already running.
-asadmin このコマンドラインパラメーターを含めると、VoiceAttackは現在のセッションの管理者として自身を実行しようとします。これは、すでに実行されているインスタンスには影響しません。

-clearasadmin This option will clear the, 'Run as an administrator' setting from the Options screen (this is in case you can't get into VoiceAttack to clear the setting for some weird reason). This does not affect the instance that is already running.
-clearasadmin このオプションは、オプション画面から「管理者として実行」設定をクリアします（これは、何らかの奇妙な理由でVoiceAttackにアクセスして設定をクリアできない場合）。これは、すでに実行されているインスタンスには影響しません。

-resetwarnings This resets various warning messages in VoiceAttack to be visible. This does not affect the instance that is already running.
-resetwarnings これにより、VoiceAttackのさまざまな警告メッセージがリセットされて表示されます。これは、すでに実行されているインスタンスには影響しません。

-priority This adjusts the Windows process priority of VoiceAttack. The values can be, 'low', 'belownormal', 'normal', 'abovenormal', 'high', and 'realtime' (all without quotes). Note that running with a value of, 'realtime' requires VA to be run as an administrator. This only works against a newly-created instance of VA and does not work against an instance that is currently running. Example: -prority high runs VA with a high-priority process.
-priority これは、VoiceAttack のWindowsプロセスの優先度を調整します。値は、'low'、'belownormal'、'normal'、'abovenormal'、'high'、および 'realtime'（すべて引用符なし）です。「リアルタイム」の値で実行するには、VAを管理者として実行する必要があります。これは、VAの新しく作成されたインスタンスに対してのみ機能し、現在実行中のインスタンスに対しては機能しません。例： -prority high は、VAを優先度の高いプロセスで実行します。

Note: If VoiceAttack is already started, most command line parameters affect the already- running instance of VoiceAttack. This way you can create desktop shortcuts that affect VoiceAttack. Command line options are case-sensitive.
注：VoiceAttackが既に開始されている場合、ほとんどのコマンドラインパラメーターはVoiceAttackの既に実行中のインスタンスに影響します。これにより、VoiceAttackに影響するデスクトップショートカットを作成できます。コマンドラインオプションでは大文字と小文字が区別されます。

Text (and Text-To-Speech) Tokens
テキスト（およびテキスト読み上げ）トークン

A VoiceAttack Token is a tag that can be used as a placeholder to represent a value in places that use text (think of Tokens as mini functions). Tokens are surrounded by curly brackets { } and will always contain a title that is case-sensitive : {DATE} or {CMD} or
VoiceAttackトークンは、テキストを使用する場所の値を表すためのプレースホルダーとして使用できるタグです（トークンはミニ関数と考えてください）。トークンは中括弧{}で囲まれ、大文字と小文字が区別されるタイトルが常に含まれます：{DATE}または{CMD}または

{CLIP}. Tokens started their life in VoiceAttack for use in Text-To-Speech (TTS). Over time, the uses for tokens grew and now they can be used in all sorts of places that require text, such as file paths for launching apps and use in conditional statements. Some tokens you may use a lot, others you may never use at all. Some are simple (like {DATE}) and some are way out there (like {EXP}). An example of how to use the first token ({TIME}) is below. It was used in TTS to help indicate the local time to the VoiceAttack user.
{クリップ}。トークンは、Text-To-Speech（TTS）で使用するためにVoiceAttackで始まりました。時間が経つにつれて、トークンの使用が拡大し、アプリを起動するためのファイルパスや条件ステートメントでの使用など、テキストを必要とするあらゆる種類の場所で使用できるようになりました。頻繁に使用するトークンと、まったく使用しないトークンがあります。あるものは（{DATE}のような）単純なものであり、あるものは（{EXP}のような）そこから抜け出す方法です。最初のトークン（{TIME}）の使用方法の例を以下に示します。TTSで使用され、VoiceAttackユーザーに現地時間を示すのに役立ちました。

“The current time is now {TIME}”.
「現在の時刻は{TIME}です」。

This value was put in the, 'Say Something with Text-To-Speech' action. When the action is executed, VoiceAttack replaces the {TIME} tag with the local time (24-hour clock):
この値は、「テキスト読み上げで何かを言う」アクションに入れられました。アクションが実行されると、VoiceAttackは{TIME}タグを現地時間（24時間制）に置き換えます。

“The current time is thirteen hundred hours”.
「現在の時間は1300時間です」。

A variation of the {TIME} token is {time} (notice the lower case). This token will be rendered as a 12-hour clock, so, “The current time is now {time}” would be rendered as “The current time is now one o’clock” when the action is executed.
{TIME}トークンのバリエーションは{time}です（小文字に注意してください）。このトークンは12時間制で表示されるため、アクションが実行されると、「現在の時刻は{time}です」は「現在の時刻は1時です」と表示されます。

Some tokens are a little bit more complicated (or flexible, depending on how you look at it) and contain parameters. An example of this would be the {RANDOM:lowValue:highValue} token. The purpose of this token is to generate a random number within a range.
一部のトークンはもう少し複雑で（見方によっては柔軟性があり）、パラメーターを含んでいます。この例は、{RANDOM：lowValue：highValue }トークンです。このトークンの目的は、範囲内の乱数を生成することです。

Notice that this token has TWO parameters: lowValue and highValue to represent the lower and upper boundaries of the range (I believe its first use was to roll virtual, 'dice'). Notice also that the parameters are separated from the title (and other parameters) by colons. As an example, let’s say we want to generate a random number between 10 and 20. The token would look like this: {RANDOM:10:20}.
このトークンには、範囲の下限と上限を表すlowValueとhighValueの2つのパラメーターがあることに注意してください（最初の使用は仮想「サイコロ」を転がすことだったと思います）。また、パラメーターはタイトル（および他のパラメーター）からコロンで区切られています。例として、10～20の乱数を生成するとします。トークンは{RANDOM：10：20}のようになります。

Using Text-To-Speech, your action could use the following statement: “I am generating a random number and it is {RANDOM:10:20}”.
Text-To-Speechを使用すると、アクションで次のステートメントを使用できます。「私は乱数を生成しています。これは{RANDOM：10：20}です」。

When the action is executed, VoiceAttack will generate the random number and replace the tag, so the result would be something like, “I am generating a random number and it is 16”.
アクションが実行されると、VoiceAttackは乱数を生成してタグを置換するため、結果は「私は乱数を生成しています。16です」のようになります。

There are some tokens that take a variable name as a parameter, such as
次のような変数名をパラメーターとして取るトークンがいくつかあります。

{TXT:variableName}. When VoiceAttack encounters a token that accepts variable names as parameters, it replaces the tag with the value in that variable. In the case of
{TXT：variableName}。VoiceAttackは、変数名をパラメーターとして受け入れるトークンを検出すると、タグをその変数の値に置き換えます。の場合

{TXT:variableName}, VoiceAttack will get a text variable’s value. {INT:variableName} will get an integer variable’s value. {BOOL:variableName} will get a boolean variable’s value (and so on). Let’s say you set up a text variable called, ’myText’ to have a value of, ’Dave’. The token you would use to get the value from ’myText’ would be {TXT:myText}. You could use the token in a Text-To-Speech statement such as this:
{TXT：variableName}、VoiceAttackはテキスト変数の値を取得します。{INT：variableName}は整数変数の値を取得します。{BOOL：variableName}はブール変数の値を取得します（など）。値が「Dave」になるように「myText」というテキスト変数を設定したとします。'myText'から値を取得するために使用するトークンは{TXT：myText}です。次のような音声合成文でトークンを使用できます。

“I’m sorry, {TXT:myText}, I’m afraid I can’t do that.” When the action is executed, the rendered statement would be, “I’m sorry, Dave, I’m afraid I can’t do that.” Then, if you set ’myText’ to have a value of, ’Sally’ and execute the statement again, the rendered statement would be, “I’m sorry, Sally, I’m afraid I can’t do that.”.
「申し訳ありませんが、{TXT：myText}、私はそれができないのではないかと思います。」アクションが実行されると、レンダリングされたステートメントは「ごめんなさい、デイブ、私はできるのではないかと思います」次に、 'myText'の値を 'Sally'に設定し、ステートメントを再度実行すると、レンダリングされたステートメントは、「ごめん、サリー、私はできるのではないか」となるでしょう。そんなことはしません。」

With version 1.6 of VoiceAttack and later, tokens can be, ‘nested’. That means you can have tokens that accept the value of other tokens. Tokens are processed from the inner- most token to the outer-most token, reprocessing with each iteration. Also, anything contained within curly brackets { } that do not resolve to a token will be rendered as whatever is contained within the brackets. To indicate a literal curly bracket, simply prefix the curly bracket with a pipe symbol: |{ or |}.
VoiceAttackのバージョン1.6以降では、トークンを「ネスト」できます。つまり、他のトークンの値を受け入れるトークンを持つことができます。トークンは、最も内側のトークンから最も外側のトークンまで処理され、各反復で再処理されます。また、中括弧{}内に含まれるトークンに解決されないものはすべて、括弧内に含まれるものとしてレンダリングされます。リテラル中括弧を示すには、単純に中括弧の前にパイプ記号を付けます：| {または|}。

Note that since tokens are reprocessed on each iteration (and although certain measures are taken within VoiceAttack to prevent it), you can end up in an infinite loop and/or cause VoiceAttack to crash, so use nested tokens with caution. If you are using an older command and it is giving you problems, you can uncheck the option, 'Use Nested Tokens' on the options screen to see if that that helps out.
トークンは各反復で再処理されるため(また、VoiceAttack内で特定の対策が講じられますが)、無限ループに陥ったり、VoiceAttackがクラッシュする可能性があるため、ネストされたトークンは注意して使用してください。古いコマンドを使用していて問題が発生している場合は、オプション画面で「ネストされたトークンを使用」オプションのチェックを外して、それが役立つかどうかを確認できます。

Note: VoiceAttack's tokens work even when there are multiple entries for random TTS statements. Remember, tokens are case-sensitive... {Time} will not be processed properly, but, {TIME} or {time} will work just fine.
注:VoiceAttackのトークンは、ランダムTTSステートメントのエントリが複数ある場合でも機能します。トークンでは大文字と小文字が区別されることに注意してください。{Time}は適切に処理されませんが、{TIME}または{time}は正常に機能します。

Below is a haphazard list of tokens in no particular order (mostly in the order they were created… lol… sorry, I will do better at some point):
以下は、特定の順序でのトークンの無計画なリストです(ほとんどが作成された順序で...笑...申し訳ありませんが、私はいくつかの時点でより良くします):

{TIME} - The current time, expressed in a 24-hour clock.
{TIME} -24時間制で表される現在の時刻。

{TIME:dateVariableName} - The indicated date variable's time, expressed in a 24-hour clock.
{TIME:dateVariableName} -指定された日付変数の時刻。24時間制で表されます。

{time} - The current time, expressed as a 12-hour clock.
{time} -12時間制で表される現在の時刻。

{time:dateVariableName} - The indicated date variable's time, expressed as a 12-hour clock.
{time:dateVariableName} -指定された日付変数の時刻。12 時間制で表されます。

{TIMEHOUR} - The hour of the current time, expressed in a 12-hour clock.
{TIMEHOUR} -現在の時間の時間。12 時間制で表されます。

{TIMEHOUR:dateVariableName} - The hour of the indicated date variable's time, expressed in a 12-hour clock.
{TIMEHOUR:dateVariableName} -指定された日付変数の時間の時間。12 時間制で表されます。

{TIMEHOUR24} – The hour of the current time, expressed in a 24-hour clock.
{TIMEHOUR24}-24 時間制で表される現在の時間の時間。

{TIMEHOUR24:dateVariableName} – The hour of the indicated date variable's time, expressed in a 24-hour clock.
{TIMEHOUR24：dateVariableName} -指定された日付変数の時間の時間。24時間制で表されます。

{TIMEMINUTE} – The minute of the current time.
{TIMEMINUTE} -現在の時間の分。

{TIMEMINUTE:dateVariableName} – The minute of the indicated date variable's time.
{TIMEMINUTE：dateVariableName} -指定された日付変数の時刻の分。

{TIMESECOND} – The second of the current time.
{TIMESECOND} -現在の時刻の秒。

{TIMESECOND:dateVariableName} – The second of the indicated date variable's time.
{TIMESECOND：dateVariableName} -指定された日付変数の時刻の秒。

{TIMEMILLISECOND} – The second of the current time.
{TIMEMILLISECOND} -現在の時刻の秒。

{TIMEMILLISECOND:dateVariableName} – The second of the indicated date variable's time.
{TIMEMILLISECOND：dateVariableName} -指定された日付変数の時刻の秒。

{TIMEAMPM} – The AM/PM designation of the current time (or the proper designation of AM/PM in the set culture).
{TIMEAMPM} -現在の時刻のAM / PM指定（または設定されたカルチャ内のAM / PMの適切な指定）。

{TIMEAMPM:dateVariableName} – The AM/PM designation of the indicated date variable's time (or the proper designation of AM/PM in the set culture).
{TIMEAMPM：dateVariableName} -指定された日付変数の時刻のAM / PM指定（または設定されたカルチャ内のAM / PMの適切な指定）。

{TIMESTAMP} – This renders the current date/time as a four-digit year, two-digit month, two-digit day, two-digit hour (24 hour), two-digit minute, two-digit second and three-digit millisecond (like this: 20180101123020500).
{TIMESTAMP} -現在の日付/時刻を4桁の年、2桁の月、2桁の日、2桁の時間（24時間）、2桁の分、2桁の秒、3桁として表示しますミリ秒（20180101123020500など）。

{TIMESTAMP:dateVariableName} – The timestamp of the indicated date variable's date/time.
{TIMESTAMP：dateVariableName} –指定された日付変数の日付/時刻のタイムスタンプ。

{DATE} – The current date, formatted as follows: 'April 3, 2015'.
{DATE} –「2015年4月3日」の形式の現在の日付。

{DATE:dateVariableName} – The indicated date variable, formatted as follows: 'April 3, 2015'.
{DATE：dateVariableName} –次の形式の指定された日付変数：「2015年4月3日」。

{DATEYEAR} – The current year as a number (2015, 2016, 2017, etc.).
{DATEYEAR} –数値としての現在の年（ 2015、2016、2017 など）。

{DATEYEAR:dateVariableName} – The indicated date variable's year as a number (2015, 2016, 2017, etc.).
{DATEYEAR：dateVariableName} –数値として示された日付変数の年（ 2015、2016、2017 など）。

{DATEDAY} – The current day of the month as a number.
{DATEDAY} –数値としての現在の日付。

{DATEDAY:dateVariableName} – The indicated date variable's day of the month as a number.
{DATEDAY：dateVariableName} –数値として示される日付変数の日付。

{DATEMONTH} – The current month, spelled out ('April', 'May', 'June', etc.).
{DATEMONTH} –綴られた現在の月（「4月」、「5月」、「6月」など）。

{DATEMONTH:dateVariableName} – The indicated date variable's month, spelled out ('April', 'May', 'June', etc.).
{DATEMONTH：dateVariableName} –指定された日付変数の月で、綴られています（「4月」、「5月」、「6月」など）。

{DATEMONTHNUMERIC} – The current month as a number (April would be 4, May would be 5, June would be 6, etc.).
{DATEMONTHNUMERIC} –数値としての現在の月（4月は4、5月は5、6月は6など）。

{DATEMONTHNUMERIC:dateVariableName} – The indicated date variable's month as a number (April would be 4, May would be 5, June would be 6, etc.).
{DATEMONTHNUMERIC：dateVariableName} –指定された日付変数の月を数値で表します（4月は4、5月は5、6月は6など）。

{DATEDAYOFWEEK} – The current day of the week spelled out ('Monday', 'Tuesday', 'Wednesday', etc.).
{DATEDAYOFWEEK} –現在の曜日のスペル(「月曜日」、「火曜日」、「水曜日」など)。

{DATEDAYOFWEEK:dateVariableName} – The indicated date variable's day of the week spelled out ('Monday', 'Tuesday', 'Wednesday', etc.).
{DATEDAYOFWEEK:dateVariableName} –指定された日付変数の曜日が綴られています(「月曜日」、「火曜日」、「水曜日」など)。

{DATETICKS} – The current date as ticks. This will be a numeric value expressed as text that can be used for comparison.
{DATETICKS} –ティックとしての現在の日付。これは、比較に使用できるテキストとして表される数値になります。

{DATETICKS:dateVariableName} – The indicated date variable as ticks. This will be a numeric value expressed as text that can be used for comparison.
{DATETICKS:dateVariableName} –ティックとして示される日付変数。これは、比較に使用できるテキストとして表される数値になります。

{DATETIMEFORMAT:textFormatVariable} – The current date/time, formatted using standard format strings. The textFormatVariable is a text variable that is required. For example, {DATETIMEFORMAT:myFormat}, with myFormat set to 'd MMMM yyyy' and the current date as the 4th of May, 2020 will render, '4 May 2020'. May the fourth be with you ;)
{DATETIMEFORMAT:textFormatVariable} –標準のフォーマット文字列を使用してフォーマットされた現在の日付/時刻。textFormatVariableは、必須のテキスト変数です。例えば、{DATETIMEFORMAT:myFormat}、D 'MMMMのYYYY'および4などの現在の日付に設定myFormatと番目月、2020は、'4月2020'レンダリングします。4番目があなたと一緒にありますように;)

{DATETIMEFORMAT:dateVariableName:textFormatVariable} – The indicated date/time variable, formatted using standard format strings. The textFormatVariable is a text variable that is required. For example, {DATETIMEFORMAT:myDate:myFormat}, with myFormat set to 'd MMMM yyyy' and myDate set as the 4th of May, 2020 will render, '4 May 2020'.
{DATETIMEFORMAT:dateVariableName:textFormatVariable} –指定された日付/時刻変数。標準のフォーマット文字列を使用してフォーマットされます。textFormatVariableは、必須のテキスト変数です。例えば、myFormat D 'MMMMのYYYY'に設定し、MyDateに該当は4として設定と{DATETIMEFORMAT:MyDateに該当myFormat} 番目の月の、2020は、'4月2020'レンダリングします。

May the fourth be with you (always) ;)
4番目があなたと一緒にいるように(常に);)

{CMD} – The name of the currently-executing command.
{CMD} –現在実行中のコマンドの名前。

{CMDSEGMENT:segment} – If your command contains, 'dynamic command sections' (see, 'Dynamic Command Sections' in the, 'Command Screen' documentation above), you can retrieve specific portions of the spoken command, indicated by its numeric position.
{CMDSEGMENT:segment } –コマンドに「動的コマンドセクション」が含まれる場合(上記の「コマンド画面」のドキュメントの「動的コマンドセクション」を参照)、音声コマンドの特定の部分を取得できます。。

This will allow you to make more precise decisions based on what was spoken.
これにより、話された内容に基づいてより正確な決定を下すことができます。

For instance, let's say you have a complex dynamic command that you use to build several kinds of items like this:
たとえば、次のようないくつかの種類のアイテムを作成するために使用する複雑な動的コマンドがあるとします。

'build [1..10][bulldogs;wolves;strikers][please;]'
' ビルド[1..10] [ブルドッグ;オオカミ;ストライカー] [お願い;]'

Then, you say, 'build 5 bulldogs' to execute that command. To find out what was built, you can check {CMDSEGMENT:2} (note that the specified, 'segment' is zero-based. So, the first segment is 0. The second is 1, and so on). The rendered value will be, 'bulldogs'.
次に、「ブルドッグを5つ作成」して、そのコマンドを実行します。ビルドされたものを調べるには、{CMDSEGMENT:2}を確認します(指定された「セグメント」はゼロベースです。したがって、最初のセグメントは0です。2番目は1などです)。レンダリングされる値は「ブルドッグ」です。

Then, you can find out how many items to build by checking {CMDSEGMENT:1}. The rendered value will be, '5' (which you can convert and use in a loop, for instance). For bonus points, you can check {CMDSEGMENT:3} and see if, 'please' was spoken and then thank the user for being so polite (or chide them if they didn't say, 'please') ;)
次に、{CMDSEGMENT:1}をチェックして、構築するアイテムの数を確認できます。レンダリングされた値は「5」になります(たとえば、変換してループで使用できます)。ボーナスポイントについては、{CMDSEGMENT:3}を確認し、「お願い」が話されたかどうかを確認してから、ユーザーがとても礼儀正しくしてくれたことに感謝します(または「お願い」と言わなかった場合は非難します);)

{CMDACTION} - The method by which the current command was executed. The possible results are, 'Spoken', 'Keyboard', 'Joystick', 'Mouse', 'Profile', 'External', 'Unrecognized', 'ProfileUnloadChange', 'ProfileUnloadClose', 'DictationRecognized', 'Plugin' and 'Other'.
{CMDACTION} -現在のコマンドが実行された方法。可能な結果は、「音声」、「キーボード」、「ジョイスティック」、「マウス」、「プロファイル」、「外部」、「認識できない」、「ProfileUnloadChange」、「ProfileUnloadClose」、「DictationRecognized」、「プラグイン」および「その他」。

The value will be, 'Spoken' if the command was executed by a spoken phrase, 'Keyboard' if the command was executed using a keyboard shortcut, 'Joystick' if executed by a joystick button, 'Mouse' if executed by a mouse button click and 'Profile' if the command was executed on profile load (from the command indicated in the profile options screen). The value will be 'External' if the command is executed from a command line or if you right-click on a command on the profile screen and execute it from there. 'Unrecognized' will be the value if the command was executed using the unrecognized phrase catch-all command in the profile options screen. ‘ProfileUnloadChange’ and ‘ProfileUnloadClose’ will be rendered if the command is executed as part of the profile being unloaded, either by changing the profile or by VoiceAttack shutting down. ‘DictationRecognized’ is rendered if the command was invoked as a result of a dictation phrase being recognized. A value of, ‘Plugin’ is rendered if the command is executed from a plugin or inline function. 'Other' is reserved.
値は、コマンドが音声フレーズによって実行された場合は「音声」、キーボードショートカットを使用してコマンドが実行された場合は「キーボード」、ジョイスティックボタンによって実行された場合は「ジョイスティック」、マウスボタンによって実行された場合は「マウス」プロファイルのロード時にコマンドが実行された場合は、[プロファイル]をクリックします（プロファイルオプション画面に表示されるコマンドから）。コマンドがコマンドラインから実行される場合、またはプロファイル画面でコマンドを右クリックしてそこから実行する場合、値は「外部」になります。プロファイルオプション画面で認識されないフレーズcatch-allコマンドを使用してコマンドが実行された場合、「認識されない」が値になります。コマンドがアンロードされるプロファイルの一部として実行される場合、「ProfileUnloadChange」および「ProfileUnloadClose」がレンダリングされます。プロファイルを変更するか、VoiceAttackをシャットダウンしてください。ディクテーションフレーズが認識された結果としてコマンドが呼び出された場合、「DictationRecognized」がレンダリングされます。コマンドがプラグインまたはインライン関数から実行される場合、「Plugin」の値がレンダリングされます。「その他」は予約されています。

{CMD_BEFORE} - When using wildcards in spoken phrases, this is the text that occurs before the wildcard phrase. For example, if using, '*rocket*' as your wildcard phrase and you say, 'i am going for a ride in my rocket ship' {CMD_BEFORE} will contain, 'i am going for a ride in my' and {CMD_AFTER} will contain 'ship'. This token does not have to be set prior to using.
{CMD_BEFORE} -音声フレーズでワイルドカードを使用する場合、これはワイルドカードフレーズの前にあるテキストです。たとえば、ワイルドカードフレーズとして「＊ rocket ＊」を使用し、「私はロケット船に乗るつもりだ」と言う場合、{CMD_BEFORE}には「私は乗るつもりです」および{CMD_AFTER }には「ship」が含まれます。このトークンは、使用する前に設定する必要はありません。

{CMD_AFTER} - When using wildcards in spoken phrases, this is the text that occurs
{CMD_AFTER} -音声フレーズでワイルドカードを使用する場合、これは発生するテキストです

after the wildcard phrase.
ワイルドカードフレーズの後。

{CMD_WILDCARDKEY} - When using wildcards in spoken phrases, this is the text that
{CMD_WILDCARDKEY} -音声フレーズでワイルドカードを使用する場合、これは

is not variable. For example, if using, '*rocket*' as your wildcard phrase and you say, 'i am going for a ride in my rocket ship' {CMD_WILDCARDKEY} will render as, 'rocket'.
可変ではありません。たとえば、ワイルドカードフレーズとして「＊ rocket ＊」を使用し、「ロケット船に乗るつもりです」と言う場合、{CMD_WILDCARDKEY}は「rocket」としてレンダリングされます。

{CMDCONFIDENCE} – This token provides the confidence level that the speech engine provides when the speech engine detects speech. The value range for a spoken command is from “0” to “100”. This value will always be, “0” when a command is not executed as a spoken command (use the, ‘{CMDACTION}’ token to check how the command was executed). Note this value is accessible from within the specified unrecognized catch-all command.
{CMDCONFIDENCE} –このトークンは、音声エンジンが音声を検出したときに音声エンジンが提供する信頼レベルを提供します。音声コマンドの値の範囲は「0」から「100」です。コマンドが音声コマンドとして実行されない場合、この値は常に「0」になります（コマンドの実行方法を確認するには、「{CMDACTION}」トークンを使用します）。この値は、指定された認識されないcatch-allコマンド内からアクセスできることに注意してください。

{CMDMINCONFIDENCE} – This token provides the minimum confidence level set by the user as it applies to the executing command. The value range is “0” to “100”. This value will be, “0” if the minimum confidence level is not set. Note that this token can return a value even if the command is not spoken.
{CMDMINCONFIDENCE} –このトークンは、実行中のコマンドに適用されるときにユーザーが設定した最小信頼レベルを提供します。値の範囲は「0」から「100」です。最小信頼レベルが設定されていない場合、この値は「0」になります。このトークンは、コマンドが話されていなくても値を返すことができることに注意してください。

{LASTSPOKENCMD} – This provides the last-spoken command phrase. Useful from within sub-commands where you need to know what was said to invoke the root command, or if you need to know what was spoken prior to the current command (if the current command was not spoken (executed by keyboard, mouse, joystick, external, etc.).
{LASTSPOKENCMD} –最後に話されたコマンドフレーズを提供します。ルートコマンドを呼び出すために言われたことを知る必要があるサブコマンド内から、または現在のコマンドの前に話されたことを知る必要がある場合（現在のコマンドが話されていない場合（キーボード、マウス、ジョイスティックによって実行される場合）、外部など）。

{PREVIOUSSPOKENCMD} – This provides the spoken command phrase prior to the last spoken command phrase.
{PREVIOUSSPOKENCMD} –最後の音声コマンドフレーズの前の音声コマンドフレーズを提供します。

{SPOKENCMD:value} – This provides access to the history of spoken commands (up to a maximum of 1000 phrases), starting at 0. A value of zero ({SPOKENCMD:0}) is the same as getting the value of the {LASTSPOKENCMD} token. A value of 1 is the same as getting the value of {PREVIOUSSPOKENCMD} ({SPOKENCMD:1}). A value of 2 would get the spoken phrase that was issued before the previous spoken command and so on. If no history exists for the value, a blank (empty string) is rendered.
{SPOKENCMD：value } –これは、0から始まる音声コマンドの履歴（最大1000フレーズ）へのアクセスを提供します。ゼロの値（{SPOKENCMD：0}）は、{ LASTSPOKENCMD}トークン。値1は、{PREVIOUSSPOKENCMD}（{SPOKENCMD：1}）の値を取得するのと同じです。値が2の場合、前の音声コマンドなどの前に発行された音声フレーズが取得されます。値の履歴が存在しない場合、空白（空の文字列）がレンダリングされます。

{CMDISSUBCOMMAND} – This token will render as ‘1’ if the currently-executing command is executing as a subcommand (a command that is executed by another command). The rendered value will be, ‘0’ if the command is not a subcommand.
{CMDISSUBCOMMAND} –現在実行中のコマンドがサブコマンド（別のコマンドによって実行されるコマンド）として実行されている場合、このトークンは「1」としてレンダリングされます。レンダリングされた値は、コマンドがサブコマンドでない場合は「0」になります。

{CMDCOUNT} – This token provides the count of top-level commands (commands that are not subcommands) that have executed since VoiceAttack had launched.
{CMDCOUNT} –このトークンは、VoiceAttackの起動後に実行されたトップレベルコマンド（サブコマンドではないコマンド）の数を提供します。

{CMDWHENISAY} – This token provides the full value of what is indicated in the, ‘When I Say’ input box on the command screen.
{CMDWHENISAY} –このトークンは、コマンド画面の「When I Say」入力ボックスに示されている内容の完全な値を提供します。

{CMDACTIVE:name} – This provides a way to test if a command is currently active. Simply provide the command name (spoken phrase), and the rendered value will be, ‘1’ if the command is active, or ‘0’ if not. Ex: {CMDACTIVE:fire weapon} will check to see if the command, ‘fire weapon’ is active.
{CMDACTIVE：name } –これは、コマンドが現在アクティブかどうかをテストする方法を提供します。コマンド名（話し言葉）を入力するだけで、表示される値は、コマンドがアクティブな場合は「1」、アクティブでない場合は「0」になります。例：{CMDACTIVE：fire weapon}は、コマンド 'fire weapon'がアクティブかどうかを確認します。

{CMDLASTUSEREXEC} – This renders the number of seconds since the last command executed by the user – spoken phrase, keyboard key press, mouse click or joystick button press. That is, it does not include subcommands, commands executed externally, right-
{CMDLASTUSEREXEC} –ユーザーが最後にコマンドを実行してからの秒数（音声フレーズ、キーボードキーの押下、マウスクリック、またはジョイスティックボタンの押下）を表示します。つまり、サブコマンド、外部で実行されるコマンド、右

click execute, etc.
実行などをクリックします

{ISLISTENINGOVERRIDE} – If the executing command was invoked by a listening override keyword (for instance, if you executed the command by saying, 'Computer, Open Door' instead of, 'Open Door') the result will be, '1'. Otherwise, the result will be, '0'.
{ISLISTENINGOVERRIDE} –実行中のコマンドがリスニングオーバーライドキーワードによって呼び出された場合（たとえば、「Open Door」ではなく「Computer、Open Door」と言ってコマンドを実行した場合）、結果は「1」になります。それ以外の場合、結果は「0」になります。

{ISCOMPOSITE} – If an executing command is composite (where there is a prefix and a suffix), the return of this token will be, '1'. Otherwise, it will be, '0'.
{ISCOMPOSITE} –実行中のコマンドが合成（プレフィックスとサフィックスがある場合）の場合、このトークンの戻り値は「1」になります。それ以外の場合、「0」になります。

{PREFIX} – If an executing command is composite (where there is a prefix and a suffix), this token will be replaced with the prefix portion of the command. If called in a non– composite command, this will result in a blank string.
{PREFIX} –実行中のコマンドがコンポジット（プレフィックスとサフィックスがある場合）の場合、このトークンはコマンドのプレフィックス部分に置き換えられます。非複合コマンドで呼び出された場合、これは空の文字列になります。

{SUFFIX} – If an executing command is composite (where there is a prefix and a suffix), this token will be replaced with the suffix portion of the command. If called in a non– composite command, this will result in a blank string.
{SUFFIX} –実行中のコマンドがコンポジット（プレフィックスとサフィックスがある場合）の場合、このトークンはコマンドのサフィックス部分に置き換えられます。非複合コマンドで呼び出された場合、これは空の文字列になります。

{COMPOSITEGROUP} – If an executing command is composite (where there is a prefix and a suffix), this token will return the group value if it is being used.
{COMPOSITEGROUP} –実行中のコマンドがコンポジット（プレフィックスとサフィックスがある場合）である場合、このトークンは使用されている場合はグループ値を返します。

{CATEGORY} – This returns the category of the executing command. If this token is used on a composite command (a command using a prefix and suffix), the result will be the category of the prefix and the category of the suffix separated by a space. For the individual parts of a composite category, see, '{PREFIX_CATEGORY}' and '{SUFFIX_CATEGORY}'.
{CATEGORY} –実行中のコマンドのカテゴリを返します。このトークンが複合コマンド（プレフィックスとサフィックスを使用するコマンド）で使用される場合、結果は、スペースで区切られたプレフィックスのカテゴリとサフィックスのカテゴリになります。複合カテゴリの個々の部分については、「{PREFIX_CATEGORY}」および「{SUFFIX_CATEGORY}」を参照してください。

{PREFIX_CATEGORY} – This returns the prefix category of the executing command (if the executing command is a composite command).
{PREFIX_CATEGORY} –実行中のコマンドのプレフィックスカテゴリを返します（実行中のコマンドが複合コマンドの場合）。

{SUFFIX_CATEGORY} – This returns the suffix category of the executing command (if the executing command is a composite command).
{SUFFIX_CATEGORY} –実行中のコマンドのサフィックスカテゴリを返します（実行中のコマンドが複合コマンドの場合）。

{QUEUESTATUS:name} – This renders the status of a command execution queue indicated in the name parameter. The following values will be rendered:
{QUEUESTATUS：名 } –これはで示されるコマンド実行キューのステータスレンダリング名パラメータ。次の値がレンダリングされます。

‘Not Initialized’ – This will be the rendered status if the queue does not exist (that is, no commands have been enqueued into a queue by the name given).
「初期化されていません」–これは、キューが存在しない場合（つまり、指定された名前でコマンドがキューにエンキューされていない場合）のレンダリングステータスです。

‘Running’ – The queue is currently running (not paused) and there is at least one command in the queue.
「実行中」–キューは現在実行されており（一時停止されていない）、キューに少なくとも1つのコマンドがあります。

‘Idle’ – The queue is running (not paused) but there are zero items in the queue. ‘Paused’ – The queue is paused, or flagged to be paused if a command is currently executing within the queue.
「アイドル」–キューは実行されています（一時停止されていません）が、キューにはアイテムがありません。'Paused'–キューは一時停止されているか、コマンドが現在キュー内で実行されている場合に一時停止のフラグが立てられます。

‘Stopped’ – The queue is in a stopped state or has been flagged to stop if a command is currently executing within the queue.
'Stopped'–キューは停止状態にあるか、コマンドが現在キュー内で実行されている場合に停止するフラグが立てられています。

{QUEUECMDCOUNT:name} – This renders the number of commands contained within a command execution queue that is indicated in the name parameter. If the queue does not exist, ‘0’ will be rendered.
{QUEUECMDCOUNT：name } –これは、name パラメーターで示されるコマンド実行キュー内に含まれるコマンドの数を表示します。キューが存在しない場合、「0」がレンダリングされます。

{QUEUEACTIVECMD:name} – This renders the name of the executing command of a command execution queue (indicated in the name parameter). If the queue does not exist, or if there is no command currently executing within the queue, an empty value ‘’ will be rendered.
{QUEUEACTIVECMD：name } –コマンド実行キューの実行コマンドの名前をレンダリングします（name パラメーターで示されます）。キューが存在しない場合、またはキュー内で現在実行されているコマンドがない場合、空の値 ''がレンダリングされます。

{QUEUECOUNT} – This renders the number of command execution queues that are currently available.
{QUEUECOUNT}– 現在利用可能なコマンド実行キューの数を表示します。

{PROFILE} – This renders the name of the currently-loaded profile.
{PROFILE} –現在ロードされているプロファイルの名前をレンダリングします。

{PROFILE_AT1}, {PROFILE_AT2}, {PROFILE_AT3} – These render the three different AuthorTags of the currently-loaded profile (see, ‘VoiceAttack Author Flags’ later on in this document). This is an advanced feature that will probably not be used by most.
{PROFILE_AT1}、{PROFILE_AT2}、{PROFILE_AT3}– これらは、現在ロードされているプロファイルの3つの異なるAuthorTagsをレンダリングします（このドキュメントの「VoiceAttack Author Flags」を参照）。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

{PREVIOUSPROFILE} – This provides the name of the profile that was loaded prior to the currently-loaded profile.
{PREVIOUSPROFILE} –現在ロードされているプロファイルの前にロードされたプロファイルの名前を提供します。

{PREVIOUSPROFILE_AT1}, {PREVIOUSPROFILE_AT2}, {PREVIOUSPROFILE_AT3} –
{PREVIOUSPROFILE_AT1}、{PREVIOUSPROFILE_AT2}、{PREVIOUSPROFILE_AT3}–

These render the three different AuthorTags of the profile that was loaded prior to the currently-loaded profile (see, 'VoiceAttack Author Flags' later on in this document). This is an advanced feature that will probably not be used by most.
これらは、現在ロードされているプロファイルの前にロードされたプロファイルの3つの異なるAuthorTagsをレンダリングします(このドキュメントで後述する「VoiceAttack Author Flags」を参照)。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

{PROFILE:value} – This provides access to the history of the names of loaded profiles (up to a maximum of 1000 profiles), starting at 0. A value of zero ({PROFILE:0}) is the same as getting the value of the {PROFILE} token. A value of 1 is the same as getting the value of
{PROFILE : value } –これは、0から始まるロードされたプロファイル(最大1000プロファイル)の名前の履歴へのアクセスを提供します。ゼロ({PROFILE : 0})の値は、値の取得と同じです。{PROFILE}トークンの。値1は、値を取得することと同じです

{PREVIOUSPROFILE} ({PROFILE:1}). A value of 2 would get the name of the profile loaded before the previous profile and so on. If no history exists for the value, a blank (empty string) is rendered.
{PREVIOUSPROFILE}({PROFILE : 1})。値が2の場合、前のプロファイルなどの前にロードされたプロファイルの名前が取得されます。値の履歴が存在しない場合、空白(空の文字列)がレンダリングされます。

{PROFILE_AT1:value}, {PROFILE_AT2:value}, {PROFILE_AT3:value} – This provides access to the history of the three AuthorTag values of loaded profiles (up to a maximum of 1000 profiles), starting at 0. A value of zero ({PROFILE_AT1:0}) is the same as getting the value of the {PROFILE_AT1} token. A value of 1 is the same as getting the value of
{PROFILE_AT1 : value }、{PROFILE_AT2 : value }、{PROFILE_AT3 : value } –これにより、0から始まる、ロードされたプロファイル(最大1000プロファイル)の3つのAuthorTag値の履歴にアクセスできます。値0 ({PROFILE_AT1 : 0})は、{PROFILE_AT1}トークンの値を取得することと同じです。値1は、値を取得することと同じです

{PREVIOUSPROFILE_AT1} ({PROFILE_AT1:1}). A value of 2 would get the Author Tags of the profile loaded before the previous profile and so on. If no history exists for the value, a blank (empty string) is rendered. (See, 'VoiceAttack Author Flags' later on in this document). This is an advanced feature that will probably not be used by most.
{PREVIOUSPROFILE_AT1}({PROFILE_AT1 : 1})。値が2の場合、前のプロファイルなどの前にプロファイルの作成者タグがロードされます。値の履歴が存在しない場合、空白(空の文字列)がレンダリングされます。(このドキュメントで後述する「VoiceAttack Author Flags」を参照してください)。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

{NEXTPROFILE} - This provides the name of the profile that has been selected to load after the current profile is unloaded. This token will render as empty if the profile is not unloading or if the profile is unloading due to VA shutting down. Note: If you haven't been able to tell by now, this token is only available from within an unload command (see, 'Execute a command each time this profile is unloaded' earlier in this document).
{NEXTPROFILE} -これは、現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルの名前を提供します。プロファイルがアンロードされていない場合、またはVAのシャットダウンによりプロファイルがアンロードされている場合、このトークンは空としてレンダリングされます。注：今までに伝えることができなかった場合、このトークンはアンロードコマンド内からのみ使用できます（このドキュメントの「このプロファイルがアンロードされるたびにコマンドを実行する」を参照）。

{NEXTPROFILE_AT1}, {NEXTPROFILE_AT2}, {NEXTPROFILE_AT3} - These provide
{NEXTPROFILE_AT1}、{NEXTPROFILE_AT2}、{NEXTPROFILE_AT3} -これらは提供します

the three different AuthorTag values of the profile that has been selected to load after the current profile is unloaded. These tokens will render as empty if the profile is not unloading or if the profile is unloading due to VA shutting down. Note: Just like {NEXTPROFILE}, These tokens are only available from within an unload command (see, 'Execute a
現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルの3つの異なるAuthorTag値。これらのトークンは、プロファイルがアンロードされていない場合、またはVAのシャットダウンによりプロファイルがアンロードされている場合、空としてレンダリングされます。注：{NEXTPROFILE}と同様に、これらのトークンはアンロードコマンド内からのみ使用できます（「 '

command each time this profile is unloaded' earlier in this document, as well as the, 'Author Flags' section later in this document). This is an advanced feature that will probably not be used by most.
このドキュメントの前半でこのプロファイルがアンロードされるたびにコマンドを実行し、このドキュメントの後の「作成者フラグ」セクションを参照してください）。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

{RANDOM:lowValue:highValue} - A random integer between a low value and a high value (inclusive) will be generated. For example, {RANDOM:1:100} will pick a number between 1 and 100. {RANDOM:77:199} will pick a number between 77 and 199.
{RANDOM：lowValue：highValue } -低い値と高い値（両端を含む）の間のランダムな整数が生成されます。たとえば、{RANDOM：1：100}は1～100の数値を選択します。{RANDOM：77：199}は77～199の数値を選択します。

{RANDOMDEC:lowValue:highValue} - A random decimal between a low value and a high value (inclusive) will be generated. The number of decimal places provided will be determined by the values provided in lowValue or highValue. For example,
{RANDOMDEC：lowValue：highValue } -低い値と高い値（両端を含む）の間のランダムな小数が生成されます。提供される小数点以下の桁数は、lowValueまたはhighValueで提供される値によって決まります。例えば、

{RANDOM:0.2:5.4444} will pick a number between 0.2 and 5.4444. An example of a returned value from that range could be 3.1234. Note this has four decimal places, since highValue is using four decimal places.
{RANDOM：0.2：5.4444}は、0.2～5.4444の数値を選択します。その範囲から返される値の例は3.1234です。highValueは小数点以下4桁を使用しているため、これには小数点以下4桁があることに注意してください。

***Important Note: Conditions have been renamed to Small Integer. The token
***重要な注意：条件の名前はSmall Integerに変更されました。トークン

{SMALL:variableName} (see below) has been created to handle small integer variables. The {COND:conditionName} and {CONDFORMAT:conditionName} have been left in for backward compatibility (you can use either token interchangeably, since they both access the same values).
小さな整数変数を処理するために{SMALL：variableName}（以下を参照）が作成されました。{COND：conditionName}と{CONDFORMAT：conditionName}は、下位互換性のために残されています（どちらも同じ値にアクセスするため、どちらのトークンも同じように使用できます）。

{SMALL:variableName } – The numeric value stored in a small integer variable will be retrieved. For example, if you have a variable called, 'My Small Int', you would use,
{SMALL：variableName } –短整数変数に格納されている数値が取得されます。たとえば、「My Small Int」という変数がある場合は、次を使用します。

{SMALL:My Small Int}. Note that with this token, everything to the left of the colon is case– sensitive. Everything to the right is not ({SMALL:mY sMaLL InT} would work the same way
{SMALL：My Small Int}。このトークンでは、コロンの左側のすべてが大文字と小文字を区別することに注意してください。右側のすべてがそうではありません（{SMALL：mY sMaLL InT}は同じように機能します

;)) The referenced variable must be set prior to using, otherwise the value will be, 'NOT SET' when accessed. NOTE: The comma formatting of this token has been removed. See, '{SMALLFORMAT}' token, below).
;））参照される変数は、使用する前に設定する必要があります。設定しない場合、値はアクセス時に「NOT SET」になります。注：このトークンのコンマ形式は削除されました。以下の「{SMALLFORMAT}」トークンを参照してください）。

{Small:variableName:defaultValue} – The small integer value stored in a small.I integer variable will be retrieved (just like above), but, if the value results in, 'Not Set', the value in the, 'defaultValue' parameter will be used. This referenced variable does not have to be set prior to using (since the default value will be used instead).
{Small：variableName：defaultValue } –small.I整数変数に格納されている短整数値が取得されます（上記と同じ）が、値が「未設定」の場合、「defaultValue」の値パラメータが使用されます。この参照変数は、使用する前に設定する必要はありません（代わりにデフォルト値が使用されるため）。

{SMALLFORMAT:variableName } – This returns the same value as {SMALL}, but the value is formatted with commas (for TTS). This used to be the default behavior of
{SMALLFORMAT：variableName }– これは{SMALL}と同じ値を返しますが、値はコンマでフォーマットされます（TTSの場合）。これは、以前のデフォルトの動作でした

{SMALL}, but, since the introduction of the {EXP} token, this token had to be created.
{SMALL}、ただし、{EXP}トークンの導入以来、このトークンを作成する必要がありました。

{SMALLFORMAT:variableName:defaultValue} – This returns the same value as
{SMALLFORMAT：variableName：defaultValue }– これは、同じ値を返します

{SMALLFORMAT}, except if the variable is, 'NOT SET', the value indicated as the default will be used.
{SMALLFORMAT}。ただし、変数が「NOT SET」の場合を除き、デフォルトとして示されている値が使用されます。

{INT:variableName} - The numeric value stored in an integer variable will be retrieved. For example, if you have a variable called, 'My Int', you would use, {INT:My Int}. Note that with this token, everything to the left of the colon is case-sensitive. Everything to the right is not ({INT:mY InT} would work the same way ;)) The referenced variable must be set prior to using, otherwise the value will be, 'NOT SET' when accessed. NOTE: The comma formatting of this token has been removed. See, '{INTFORMAT}' token, below).
{INT:variableName } -整数変数に格納されている数値が取得されます。たとえば、「My Int」という変数がある場合、{INT:My Int}を使用します。このトークンでは、コロンの左側のすべてが大文字と小文字を区別することに注意してください。右側のすべてはそうではありません({INT:mY InT}は同じように機能します;))参照される変数は、使用する前に設定する必要があります。そうでない場合、アクセス時に値は「NOT SET」になります。注:このトークンのコンマ形式は削除されました。以下の「{INTFORMAT}」トークンを参照してください)。

{INT:variableName:defaultValue} - The integer value stored in an integer variable will be retrieved (just like above), but, if the value results in, 'Not Set', the value in the, 'defaultValue' parameter will be used. This referenced variable does not have to be set prior to using (since the default value will be used instead).
{INT:variableName:defaultValue } -整数変数に格納されている整数値が取得されます(上記と同じ)が、値が「Not Set」の場合、「defaultValue」パラメーターの値が使用されます。この参照変数は、使用する前に設定する必要はありません(代わりにデフォルト値が使用されるため)。

{INTFORMAT:variableName} - This returns the same value as {INT}, but the value is formatted with commas (for TTS). This used to be the default behavior of {INT}, but, since the introduction of the {EXP} token, this token had to be created.
{INTFORMAT:variableName }- これは{INT}と同じ値を返しますが、値はコンマでフォーマットされます(TTSの場合)。これは以前は{INT}のデフォルトの動作でしたが、{EXP}トークンの導入以来、このトークンを作成する必要がありました。

{INTFORMAT:variableName:defaultValue} - This returns the same value as
{INTFORMAT:variableName:defaultValue }- これは、次と同じ値を返します

{INTFORMAT}, except if the variable is, 'NOT SET', the value indicated as the default will be used.
{INTFORMAT}。ただし、変数が「NOT SET」の場合を除き、デフォルトとして示される値が使用されます。

{DEC:variableName} - The numeric value stored in a decimal variable will be retrieved. For example, if you have a variable called, 'My Decimal', you would use,
{DEC:variableName } -10 進変数に格納されている数値が取得されます。たとえば、「My Decimal」という変数がある場合、次を使用します。

{DEC:My Decimal}. Note that with this token, everything to the left of the colon is case- sensitive. Everything to the right is not ({DEC:mY DeCImal} would work the same way ;)) The referenced variable must be set prior to using, otherwise the value will be, 'NOT SET' when accessed. NOTE: If you plan on using this token for calculations (for example, with the {EXP} token), or, if you plan on sharing your profile with folks in other countries, you will want to use the {DECINV} token instead (see below).
{DEC：My Decimal}。このトークンでは、コロンの左側のすべてが大文字と小文字を区別することに注意してください。右側のすべてはそうではありません（{DEC：mY DeCImal}は同じように機能します;)）参照される変数は、使用する前に設定する必要があります。そうでない場合、アクセス時に値は「NOT SET」になります。注：計算にこのトークンを使用する予定がある場合（たとえば{EXP}トークンを使用する場合）、または他の国のユーザーとプロファイルを共有する予定がある場合は、代わりに{DECINV}トークンを使用することをお勧めします（下記参照）。

{DEC:variableName:defaultValue} - The decimal value stored in a decimal variable will be retrieved (just like above), but, if the value results in, 'Not Set', the value in the, 'defaultValue' parameter will be used. This referenced variable does not have to be set prior to using (since the default value will be used instead). NOTE: If you plan on using this token for calculations (for example, with the {EXP} token), or, if you plan on sharing your profile with folks in other countries, you will want to use the {DECINV} token instead (see below).
{DEC：variableName：defaultValue } -10 進変数に格納されている10進値が取得されます（上記と同じ）が、値が「未設定」の場合、「defaultValue」パラメーターの値が使用されます。この参照変数は、使用する前に設定する必要はありません（代わりにデフォルト値が使用されるため）。注：計算にこのトークンを使用する予定がある場合（たとえば{EXP}トークンを使用する場合）、または他の国のユーザーとプロファイルを共有する予定がある場合は、代わりに{DECINV}トークンを使用することをお勧めします（下記参照）。

{DECINV:variableName} and {DECINV:variableName:defaultValue} - This works exactly the same as the {DEC} tokens above, except that the value will be rendered using the invariant culture. That means that your decimal value will always be rendered with a point ('.') instead of a comma, (','). You will want to use this token when needing decimal variables to be used in calculations (such as with {EXP} and {EXPDECINV}) and if you intend on sharing your profiles with others in different countries.
{DECINV：variableName }および{DECINV：variableName：defaultValue } -これは、値が不変カルチャを使用してレンダリングされることを除いて、上記の{DEC}トークンとまったく同じように機能します。つまり、小数値は常にコンマ('、')ではなくポイント('。')でレンダリングされます。10進変数を計算で使用する必要がある場合（{EXP}および{EXPDECINV}など）および異なる国の他のユーザーとプロファイルを共有する場合は、このトークンを使用します。

{BOOL:variableName} - The text representation of the boolean value will be retrieved. If the variable value is true, the token will be replaced with, 'True'. If false, the token will be replaced with, 'False'. The referenced variable must be set prior to using, otherwise the value will be, 'NOT SET' when accessed.
{BOOL：variableName } -ブール値のテキスト表現が取得されます。変数値がtrueの場合、トークンは「True」に置き換えられます。falseの場合、トークンは「False」に置き換えられます。参照される変数は、使用する前に設定する必要があります。設定しない場合、値はアクセス時に「NOT SET」になります。

{BOOL:variableName:defaultValue} - The Boolean (true/false) value stored in a Boolean variable will be retrieved (just like above), but, if the value results in, 'Not Set', the value in the, 'defaultValue' parameter will be used. This referenced variable does not have to be set prior to using (since the default value will be used instead).
{BOOL：variableName：defaultValue } -ブール変数に格納されたブール値（true / false）は取得されます（上記と同じ）が、値が「未設定」の場合、「defaultValue」の値'パラメータが使用されます。この参照変数は、使用する前に設定する必要はありません（代わりにデフォルト値が使用されるため）。

{TXT:variableName} – The text value stored in a text variable will be retrieved. This works
{TXT：variableName } –テキスト変数に保存されているテキスト値が取得されます。これは動作します

the same way as all the other types (above), except that... well... you get the idea. The variable referenced by this token must be set prior to using, otherwise the value will be, 'NOT SET' when accessed.
他のすべてのタイプ（上記）と同じ方法ですが、それ以外は...まあ...アイデアが得られます。このトークンによって参照される変数は、使用する前に設定する必要があります。そうしないと、アクセス時に値が「NOT SET」になります。

{TXT:variableName:defaultValue} – The text value stored in a text variable will be retrieved (just like above), but, if the value results in, 'Not Set', the text in the, 'defaultValue' parameter will be used. This referenced variable does not have to be set prior to using (since the default value will be used instead).
{TXT：variableName：defaultValue } –テキスト変数に格納されたテキスト値が取得されます（上記と同様）が、値が「未設定」の場合、「defaultValue」パラメーターのテキストが使用されます。この参照変数は、使用する前に設定する必要はありません（代わりにデフォルト値が使用されるため）。

{TXTURL:variableName} – The text value stored in a text variable will be retrieved (again, just like above), but the result will be URL encoded.
{TXTURL：variableName } –テキスト変数に格納されたテキスト値が取得されます（これも上記と同様）が、結果はURLエンコードされます。

{TXTRANDOM:value} – This token will do a simple randomize of its text contents. For example, a token could be used in TTS to say, 'I love {TXTRANDOM:food;animals;jewelry}' and the result would be, 'I love animals' or, 'I love food' or, 'I love jewelry'. Note that the, 'value' portion can be another token, however, if you want to randomize randomized tokens, it is suggested that you use text value variables to store the values, otherwise it may get a little out of sorts. Experiment with this one, since the permutations for testing are a little out there :)
{TXTRANDOM：value } –このトークンは、テキストコンテンツの単純なランダム化を行います。たとえば、トークンをTTSで使用して、「私は{TXTRANDOM：food; animals; jewelry}が大好き」と言うことができ、その結果は「動物が大好き」または「食べ物が大好き」または「ジュエリーが大好き」になります。 '。「値」部分は別のトークンになりますが、ランダム化されたトークンをランダム化する場合は、テキスト値変数を使用して値を保存することをお勧めします。そうしないと、少しソートが悪くなる場合があります。テスト用の順列が少しありますので、これを試してください:)

{TXTLEN:variableName / value} – This will return the length of the text variable's value. This token can also evaluate a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, ‘myTextVariable’ is set to, ‘weapons’, {TXTLEN:myTextVariable} will evaluate to “7”.
{TXTLEN：variableName / value } –これは、テキスト変数の値の長さを返します。リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンも評価できます。たとえば、テキスト変数「myTextVariable」が「weapons」に設定されている場合、{TXTLEN：myTextVariable}は「7」に評価されます。

{TXTLEN:”{TXT:myTextVariable}”} will also evaluate to “7”. {TXTLEN:”apple pie”} will evaluate to “9”.
{TXTLEN：” {TXT：myTextVariable}”}も「7」と評価されます。{TXTLEN：” apple pie”}は「9」と評価されます。

{TXTUPPER:variableName / value} – This will return the text variable's value in upper case. This token can also render a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, 'myTextUpper' is set to, 'citadel', {TXTUPPER:myTextUpper} will render as "CITADEL".
{TXTUPPER:variableName / value } –これは、テキスト変数の値を大文字で返します。リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンもレンダリングできます。たとえば、テキスト変数「myTextUpper」が「citadel」に設定されている場合、{TXTUPPER:myTextUpper}は「CITADEL」としてレンダリングされます。

{TXTUPPER:"{TXT:myTextUpper}"} will also render as, "CITADEL". {TXTUPPER:"apple pie"} will render as, "APPLE PIE".
{TXTUPPER:" {TXT:myTextUpper}"}も「CITADEL」としてレンダリングされます。{TXTUPPER:" apple pie"}は、「APPLE PIE」としてレンダリングされます。

{TXTLOWER:variableName / value} – This will return the text variable's value in lower case. This token can also render a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, 'myTextLower' is set to, 'TeXaS', {TXTLOWER:myTextLower} will render as "texas".
{TXTLOWER:variableName / value } –これは、テキスト変数の値を小文字で返します。リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンもレンダリングできます。たとえば、テキスト変数「myTextLower」が「TeXaS」に設定されている場合、{TXTLOWER:myTextLower}は「texas」としてレンダリングされます。

{TXTLOWER:"{TXT:myTextLower}"} will also render as, "texas". {TXTLOWER:"PECAN PIE"} will render as, "pecan pie".
{TXTLOWER:" {TXT:myTextLower}"}も「texas」としてレンダリングされます。{TXTLOWER:" PECAN PIE"}は、「pecan pie」としてレンダリングされます。

{TXTTRIM:variableName / value} – This will return the text variable's value with spaces removed from the beginning and end. This token can also process a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, 'myTextTrim' is set to, ' Parachute ', {TXTTRIM:myTextTrim} will return as "Parachute". {TXTTRIM:"{TXT:myTextTrim}"} will also return as, "Parachute".
{TXTTRIM:variableName / value } –これは、テキスト変数の値を返し、先頭と末尾からスペースを削除します。リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンも処理できます。たとえば、テキスト変数「myTextTrim」が「Parachute」に設定されている場合、{TXTTRIM:myTextTrim}は「Parachute」として返されます。{TXTTRIM:" {TXT:myTextTrim}"}も「Parachute」として返されます。

{TXTTRIM:"Yellow Submarine "} will return as, "Yellow Submarine".
{TXTTRIM:" Yellow Submarine"}は、「Yellow Submarine」として戻ります。

{TXTREPLACEVAR:variableSource / value:variableFrom / value:variableTo / value} –
{TXTREPLACEVAR:variableSource / value:variableFrom / value:variableTo / value } –

This will return the text variable's value with the text indicated in variableFrom as the value in variableTo. For example: Variable, 'myVariable' value set to, 'This is a test'. Variable, 'mySearch' value set to, 'test' and variable, 'myReplacement' value set to, 'monkey'.
これは、変数変数の値を、変数 Fromで示されるテキストを変数Toの値として返します。例:変数、「myVariable」値が「This is a test」に設定されている。変数、「mySearch」に設定された値、「test」および変数、「myReplacement」に設定された値、「monkey」。

{TXTREPLACEVAR:myVariable:mySearch:myReplacement} renders as, ‘This is a monkey’. This token can also process a literal text value or token for each parameter, if each literal text or token is contained between double quotes. For example,
{TXTREPLACEVAR：myVariable：mySearch：myReplacement}は、「これは猿です」と表示されます。このトークンは、各リテラルテキストまたはトークンが二重引用符の間に含まれている場合、各パラメーターのリテラルテキスト値またはトークンも処理できます。例えば、

{TXTREPLACEVAR:myVariable:”test”:”monkey”} will also render as, “This is a monkey”.
{TXTREPLACEVAR：myVariable：” test”：” monkey”}も、「これは猿です」と表示されます。

{TXTNUM:variableName / value} – This will attempt to remove all characters except numeric (0-9, ., -). This token can also render a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, ‘myTextNum’ is set to, ‘8675309 Jenny’, {TXTNUM:myTextNum} will return as “8675309”.
{TXTNUM：variableName / value } –数値（0～9、。、-）以外のすべての文字を削除しようとします。リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンもレンダリングできます。たとえば、テキスト変数「myTextNum」が「8675309 Jenny」に設定されている場合、{TXTNUM：myTextNum}は「8675309」として返されます。

{TXTNUM:”{TXT:myTextNum}”} will also render as, “8675309”. {TXTNUM:”Brick, 08724”} will render as, “08724”.
{TXTNUM：” {TXT：myTextNum}”}も「8675309」としてレンダリングされます。{TXTNUM：” Brick、08724”}は、「08724」としてレンダリングされます。

{TXTALPHA:variableName / value} – This will attempt to remove all numeric characters (0-9). This token can also render a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, ‘myTextAlpha’ is set to, ‘8675309 Jenny’, {TXTALPHA:myTextAlpha} will return as “ Jenny”.
{TXTALPHA：variableName / value } –すべての数字（0-9）を削除しようとします。リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンもレンダリングできます。たとえば、テキスト変数「myTextAlpha」が「8675309 Jenny」に設定されている場合、{TXTALPHA：myTextAlpha}は「Jenny」として返されます。

{TXTALPHA:”{TXT:myTextNum}”} will also render as, “ Jenny”. {TXTALPHA:”Brick, 08724”} will render as, “Brick, ”.
{TXTALPHA：” {TXT：myTextNum}”}も「Jenny」としてレンダリングされます。{TXTALPHA：” Brick、08724”}は、「Brick、」としてレンダリングされます。

{TXTWORDTONUM:variableName / value : selection / value} – This will replace numeric words with integer representations. This is to assist (workaround) in the way that the speech engine interprets single-digit numbers. For instance, the speech engine interprets, “one” as, “one”, “two” as “two”, but it interprets, “ten” as “10”, “eleven” as “11” and so on.
{TXTWORDTONUM：variableName / value：selection / value} –数値の単語を整数表現に置き換えます。これは、音声エンジンが1桁の数字を解釈する方法を支援（回避）するためです。たとえば、スピーチエンジンは、「1」を「1」、「2」を「2」と解釈しますが、「10」を「10」、「11」を「11」などと解釈します。

The variableName parameter is the parameter that will hold the text to convert. The variableName parameter can accept a variable name as well as a token/literal as long as the token/literal is between double quotes. The selection parameter is optional. It can also be a variable name or token/literal if the token/literal is between double quotes. The selection parameter allows you to specify something other than English word replacements. The value for selection must be a comma-delimited list of exactly ten items (starting at “zero”) in order to work. Here are some examples:
variableNameにはパラメータが変換するテキストを保持するパラメータです。variableNameにはリテラルトークンは/であるように、パラメータは、変数名だけでなく、限りリテラルトークンを/受け入れることができ、二重引用符の間。選択パラメータはオプションです。トークン/リテラルが二重引用符の間にある場合は、変数名またはトークン/リテラルにすることもできます。選択パラメータを使用すると、英語の単語の代替品以外のものを指定することができます。選択の値は、機能するために正確に10個の項目（「ゼロ」から始まる）のコンマ区切りリストでなければなりません。ここではいくつかの例を示します。

{TXTWORDTONUM:”One thing leads two another”} will render as “1 thing leads 2 another”. Note that the second parameter is omitted, since the default is English.
{TXTWORDTONUM：″1つの事がもう2つを導く″}は、「1つの事がもう2つを導く」としてレンダリングされます。デフォルトは英語であるため、2番目のパラメーターは省略されていることに注意してください。

In this next example, myText variable value is set to “viajando al sector uno”.
次の例では、myText変数の値が「viajando al sector uno」に設定されています。

{TXTWORDTONUM:myTextVariable:“cero,uno,dos,tres,quatro,cinco,seis,siete,ocho,nueve ”} will render as “viajando al sector 1”. Note that the selection parameter is a literal value that is enclosed in double quotes, has ten items and is comma-delimited.
{TXTWORDTONUM：myTextVariable：“ cero、uno、dos、tres、quatro、cinco、seis、siete、ocho、nueve”}は、「viajando al sector 1」としてレンダリングされます。選択パラメーターは、二重引用符で囲まれ、10個のアイテムがあり、コンマ区切りのリテラル値であることに注意してください。

{TXTTITLE:variableName / value} – This will attempt to title case the value (for supported locales). For example, 'war and peace' becomes, 'War And Peace'. Yes… I know that's not proper title case (at least in English), but it's as close as the framework will allow without linguistic rules. Sorry!
{TXTTITLE：variableName / value } –これにより、大文字と小文字が区別されます（サポートされているロケールの場合）。たとえば、「戦争と平和」は「戦争と平和」になります。はい...それは適切なタイトルケースではありません（少なくとも英語では）が、フレームワークが言語規則なしで許す限り近いものです。ごめんなさい！

This token can also render a literal text value or token, if the literal text or token is contained between double quotes. For example, if text variable, ‘myTextTitle’ is set to, ‘napoleon dynamite’, {TXTTITLE:myTextTitle} will return as “Napoleon Dynamite”.
リテラルテキストまたはトークンが二重引用符の間に含まれている場合、このトークンはリテラルテキスト値またはトークンもレンダリングできます。たとえば、テキスト変数「myTextTitle」が「napoleon dynamite」に設定されている場合、{TXTTITLE：myTextTitle}は「Napoleon Dynamite」として返されます。

{TXTTITLE:”{TXT:myTextTitle}”} will also render as, “Napoleon Dynamite”.
{TXTTITLE：” {TXT：myTextTitle}”}も「Napoleon Dynamite」としてレンダリングされます。

{TXTTITLE:"nacho libre"} will render as, "Nacho Libre".
{TXTTITLE:" nacho libre"}は、「Nacho Libre」としてレンダリングされます。

{TXTCONCAT:variableName1 / value:variableName2 / value} - This will attempt to concatenate the text in variable 2 to the text of variable 1. For instance, let's say var1 is set to "Good" and var2 is set to " Morning" (note the space). {TXTCONCAT:var1:var2} will yield, "Good Morning". If variable 1 or variable 2 are "Not set", the result will be, "Not set". This token can also render a literal text value or token for either parameter, if the literal text or token is contained between double quotes. For example, if text variable, 'myTextConcat' is set to, 'Good', {TXTCONCAT:myTextConcat, " Morning"} will return as " Good Morning". {TXTCONCAT:"Good":" Morning"} will also render as, "Good Morning".
{TXTCONCAT:variableName1 / value:variableName2 / value } -変数2のテキストを変数1のテキストに連結しようとします。たとえば、var1を「Good」に設定し、var2を「Morning」に設定するとします(スペースに注意してください)。{TXTCONCAT:var1:var2}は、「おはよう」を生成します。変数1または変数2が「未設定」の場合、結果は「未設定」になります。このトークンは、リテラルテキストまたはトークンが二重引用符の間に含まれている場合、いずれかのパラメーターのリテラルテキスト値またはトークンをレンダリングすることもできます。たとえば、テキスト変数 'myTextConcat'が 'Good'に設定されている場合、{TXTCONCAT:myTextConcat、" Morning"}は" Good Morning"として返されます。{TXTCONCAT:"Good":"Morning"}も「Good Morning」と表示されます。

{TXTSUBSTR:textVariableOf / text value : intVariableBegin / int value : intVariableLength / int value} - this will return a portion (substring) of the text provided in textVariableOf, starting at intVariableBegin with a length of intVariableLength. This token uses three parameters that may contain variables, literal values or tokens. If variables are used in any position, the variables MUST BE SET prior to using. The value indicated for the beginning of the substring (second parameter, intVariableBegin) is zero-based. That means, if you want to retrieve the substring at the beginning of the provided text (txtVariableOf), set the beginning to zero. If the value of intVariableBegin resolves to, 'Not Set', zero will be assumed. To indicate the length of the substring, provide the length in the third parameter (intVariableLength variable). If the value in the third parameter resolves to, 'Not Set' or if the value in the third parameter ends up past the end of the provided text, the end of the provided text is assumed. If either the value of intVariableBegin or intVariableLength is less than zero, the result will be an empty (blank) string.
{TXTSUBSTR:textVariableOf / text value:intVariableBegin / int value:intVariableLength / int value} -これは、intVariableBeginからintVariableLengthの長さで始まるtextVariableOfで提供されるテキストの一部(サブストリング)を返します。このトークンは、変数、リテラル値、またはトークンを含む3つのパラメーターを使用します。変数が任意の位置で使用される場合、変数を設定する必要があります使用する前に。部分文字列の先頭に示される値(2番目のパラメーター、intVariableBegin)はゼロベースです。つまり、指定されたテキスト(txtVariableOf)の先頭の部分文字列を取得する場合は、先頭をゼロに設定します。intVariableBeginの値が「Not Set」に解決される場合、ゼロが想定されます。部分文字列の長さを示すには、3番目のパラメーター(intVariableLength変数)で長さを指定します。3番目のパラメーターの値が「未設定」に解決される場合、または3番目のパラメーターの値が提供されたテキストの終わりを超えた場合、提供されたテキストの終わりが想定されます。intVariableBeginまたはintVariableLengthの値がゼロより小さい場合、結果は空(空白)の文字列になります。

{TXTSUBSTR:"We are the Champions":11:9} will render as, 'Champions' (note that, 'Champions' starts at position 12, but since the beginning position is zero-based, the value is 11).
{TXTSUBSTR:"We are the Champions":11:9}は、「Champions」としてレンダリングされます(「Champions」は12桁目から始まりますが、開始位置はゼロから始まるため、値は11です)。

{TXTPOS:textVariableOf / text value : textVariableIn / text value : intVariableStart / int value} – this will return the position of text (txtVariableOf) within more text (txtVarIn) starting at a given point (intVarStart). This uses three parameters that may contain variables, literal values or tokens. If variables are used in any position, the variables MUST BE SET prior to using. The text search is case-sensitive. If the text is found, the result will be the zero- based index of the start of the text. Otherwise, if the text is not found the result will be "-1".
{TXTPOS：textVariableOf / text value：textVariableIn / text value：intVariableStart / int value} -これは、指定されたポイント（intVarStart）から始まるテキスト（txtVarIn）内のテキスト（txtVariableOf）の位置を返します。これは、変数、リテラル値、またはトークンを含む3つのパラメーターを使用します。変数が任意の位置で使用される場合、使用する前に変数を設定する必要があります。テキスト検索では大文字と小文字が区別されます。テキストが見つかった場合、結果はテキストの先頭のゼロベースのインデックスになります。それ以外の場合、テキストが見つからない場合、結果は「-1」になります。

For example (using ALL VARIABLES), if you are wanting to find the word, 'active' in the phrase, 'all systems active' you will first need to set a text variable to the word, 'active'. In this example, that variable is called, 'txtFindActive'. Next, you will need to set a second text variable to the phrase, 'all systems active'. That variable is called, 'txtStatus'. Next, you will need to set a variable to indicate the position where you would like to start searching.
たとえば、（ALL VARIABLESを使用して）フレーズで「アクティブ」という単語、「すべてのシステムがアクティブ」という単語を検索する場合、最初にテキスト変数を「アクティブ」という単語に設定する必要があります。この例では、その変数は「txtFindActive」と呼ばれます。次に、「all systems active」というフレーズに2番目のテキスト変数を設定する必要があります。その変数は「txtStatus」と呼ばれます。次に、検索を開始する位置を示す変数を設定する必要があります。

Since you would like to search the entire phrase, you will need to set this variable to zero. So, a third variable, 'intPosition' needs to be set to zero (see note on this below). You will then use this token with the three variables like this:
フレーズ全体を検索するため、この変数をゼロに設定する必要があります。そのため、3番目の変数 'intPosition'をゼロに設定する必要があります（この点に関する注意を参照してください）。次に、このトークンを次のような3つの変数で使用します。

{TXTPOS:txtFindActive:txtStatus:intPosition}. The result will be "12", as the position of 'active' appears in position 12 (the phrase actually starts at the 13th character, but we are using a zero-based positioning system).
{TXTPOS：txtFindActive：txtStatus：intPosition}。「アクティブ」の位置が位置12にあるため、結果は「12」になります（フレーズは実際には13番目の文字から始まりますが、ゼロベースのポジショニングシステムを使用しています）。

Note: if txtVarIn or txtVar of are not set, the result will be, '-1'. If intVarStart is not set, 0 is
注：txtVarInまたはtxtVarが設定されていない場合、結果は「-1」になります。intVarStartが設定されていない場合、0は

assumed (for convenience).
想定（便宜上）。

Another example, using variables AND literal values based on the example above, you can create a statement that does the same thing: {TXTPOS:"active":txtStatus:5}. This will search for the word, 'active' starting at position 5 (it still returns, "12"). Note that double quotes are required if you are using literal text in the first and second parameters (and not required for the last parameter).
上記の例に基づいた変数とリテラル値を使用する別の例では、同じことを行うステートメント{TXTPOS：" active"：txtStatus：5}を作成できます。これにより、位置5から始まる「アクティブ」という単語が検索されます（「12」が返されます）。最初と2番目のパラメーターでリテラルテキストを使用する場合は、二重引用符が必要で あることに注意してください（最後のパラメーターには必要ありません）。

{TXTLASTPOS:textVariableOf / text value : textVariableIn / text value} and
{TXTLASTPOS：textVariableOf / text value：textVariableIn / text value}および

{TXTLASTPOS:textVariableOf / text value : textVariableIn / text value : intVariableStart / int value} – this will return the last position of text (txtVariableOf) within more text (txtVarIn) starting at a given point (intVarStart). This uses three parameters that may contain variables, literal values or tokens. If variables are used in any position, the variables MUST BE SET prior to using. The text search is case-sensitive. If the text is found, the result will be the zero-based index of the start of the text. Otherwise, if the text is not found the result will be "-1". One thing you will need to understand about this token is that, unlike {TXTPOS:}, the start position is from the end of the text to the beginning. So, the search will begin at the length of the text and move to zero. For example,
{TXTLASTPOS：textVariableOf / text value：textVariableIn / text value：intVariableStart / int value} -これは、特定のポイント（intVarStart）から始まるテキスト（txtVarIn）内のテキストの最後の位置（txtVariableOf）を返します。これは、変数、リテラル値、またはトークンを含む3つのパラメーターを使用します。変数が任意の位置で使用される場合、使用する前に変数を設定する必要があります。テキスト検索では大文字と小文字が区別されます。テキストが見つかった場合、結果はテキストの先頭のゼロベースのインデックスになります。それ以外の場合、テキストが見つからない場合、結果は「-1」になります。このトークンについて理解しておく必要があることの1つは、{TXTPOS：}とは異なり、開始位置はテキストの終わりから始まりです。したがって、検索はテキストの長さから始まり、ゼロに移動します。例えば、

{TXTLASTPOS:"abc", "123abc":0} will return "-1" because the search begins and ends at 0 (the beginning).
{TXTLASTPOS:"abc", "abc123abc":9} will return "6" because the search begins at 9 (the end of the text) and moves left. Note that start positions that are beyond the length of the search string will be adjusted to the length of the search string.
{TXTLASTPOS：" abc"、" 123abc"：0}は、検索が0（先頭）で開始および終了するため、「-1」を返します。
{TXTLASTPOS：" abc"、" abc123abc"：9}は、検索が9（テキストの終わり）から始まり左に移動するため、「6」を返します。検索文字列の長さを超える開始位置は、検索文字列の長さに合わせて調整されることに注意してください。

Note that there are two variations of this token. One with a numeric start position and one without. If you use the option without a start position, the end of the text is assumed.
このトークンには2つのバリエーションがあることに注意してください。開始位置が数値であるものとないもの。開始位置なしでオプションを使用すると、テキストの終わりが想定されます。

{TXTLASTPOS:"abc", "abc123abc"} will again return "6".
{TXTLASTPOS：" abc"、" abc123abc"}は再び" 6"を返します。

For brevity, the variable usage examples for this token are the same as they are for
簡潔にするため、このトークンの変数の使用例は、次の場合と同じです。

{TXTPOS:}, only that the search begins at the end of the search string instead of the start.
{TXTPOS：}、検索は検索文字列の先頭ではなく末尾から開始することのみ。

{SPACE} – Renders as a single space. This really only useful in very specific circumstances when a literal value will not do.
{SPACE} -単一のスペースとしてレンダリングされます。これは、リテラル値が機能しない非常に特定の状況でのみ有効です。

{NEWLINE} – Renders as a new line. Again, this really only useful in very specific circumstances when a literal value will not do.
{NEWLINE} –新しい行として表示されます。繰り返しますが、これは、リテラル値が機能しない非常に特定の状況でのみ有効です。

{GUID} – Renders as a GUID (36-character unique identifier).
{GUID} –GUID（36文字の一意の識別子）として表示されます。

{GUIDCLEAN} – Renders as a GUID without the dashes (32-character unique identifier).
{GUIDCLEAN} –ダッシュなしのGUID（32文字の一意の識別子）としてレンダリングします。

{ACTIVEWINDOWTITLE} – Returns the active window's title text.
{ACTIVEWINDOWTITLE} –アクティブウィンドウのタイトルテキストを返します。

{ACTIVEWINDOWPROCESSNAME} – Returns the active window's process name (the one you see in Task Manager).
{ACTIVEWINDOWPROCESSNAME} –アクティブウィンドウのプロセス名（タスクマネージャーに表示されるプロセス名）を返します。

{ACTIVEWINDOWPROCESSID} – Returns the active window's process id (the one you see in Task Manager details).
{ACTIVEWINDOWPROCESSID} –アクティブウィンドウのプロセスID（タスクマネージャーの詳細に表示されるプロセスID）を返します。

{ACTIVEWINDOWPATH} – Returns the path of the active window's executable.
{ACTIVEWINDOWPATH} –アクティブウィンドウの実行可能ファイルのパスを返します。

{ACTIVEWINDOWWIDTH} – Returns the active window's width. Helps with resizing/moving.
{ACTIVEWINDOWWIDTH} –アクティブウィンドウの幅を返します。サイズ変更/移動を支援します。

{ACTIVEWINDOWHEIGHT} – Returns the active window's height. Helps with resizing/moving.
{ACTIVEWINDOWHEIGHT} –アクティブウィンドウの高さを返します。サイズ変更/移動を支援します。

{ACTIVEWINDOWTOP} – Returns the active window's top (Y coordinate). Helps with resizing/moving.
{ACTIVEWINDOWTOP} –アクティブウィンドウの上部（Y座標）を返します。サイズ変更/移動を支援します。

{ACTIVEWINDOWLEFT} – Returns the active window's left (X coordinate). Helps with resizing/moving.
{ACTIVEWINDOWLEFT} –アクティブウィンドウの左（X座標）を返します。サイズ変更/移動を支援します。

{ACTIVEWINDOWRIGHT} – Returns the active window's right (left + width). Helps with resizing/moving.
{ACTIVEWINDOWRIGHT} –アクティブウィンドウの右（左+幅）を返します。サイズ変更/移動を支援します。

{ACTIVEWINDOWBOTTOM} – Returns the active window's bottom (top + height). Helps with resizing/moving.
{ACTIVEWINDOWBOTTOM} –アクティブウィンドウの下部（上部+高さ）を返します。サイズ変更/移動を支援します。

{PROCESSEXISTS:textVariable} – Returns "1" if a process with the name specified in textVariable exists. Returns "0" if not. Note that this can take wildcards (*). For instance, if textVariable contains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
{PROCESSEXISTS : textVariable } – textVariableで指定された名前のプロセスが存在する場合、「1」を返します。そうでない場合は「0」を返します。これにはワイルドカード（*）を使用できることに注意してください。たとえば、textVariableに「* notepad *」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

{PROCESSCOUNT:textVariable} – Returns "1" or more if processes with the name specified in textVariable exist. Returns "0" if there are none. Note that this can take wildcards (*). For instance, if textVariable contains '*notepad*' (without quotes), the search will look for processes that have a name that contains 'notepad'.
{PROCESSCOUNT : textVariable } – textVariableで 指定された名前のプロセスが存在する場合、「1」以上を返します。存在しない場合は「0」を返します。これにはワイルドカード（*）を使用できることに注意してください。たとえば、textVariableに「* notepad *」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

{PROCESSFOREGROUND:textVariable} – Returns "1" if a process with a main window with the title specified in textVariable is the foreground window. Returns "0" if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
{PROCESSFOREGROUND : textVariable } – textVariableで指定されたタイトルのメインウィンドウを持つプロセスがフォアグラウンドウィンドウである場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、textVariable に「* notepad *」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

{PROCESSMINIMIZED:textVariable} – Returns "1" if a process with a main window with the title specified in textVariable is minimized. Returns "0" if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariablecontains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
{PROCESSMINIMIZED : textVariable } – textVariableで指定されたタイトルのメインウィンドウを持つプロセスが最小化されている場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、textVariable に「* notepad *」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

{PROCESSMAXIMIZED:textVariable} – Returns “1” if a process with a main window with the title specified in textVariable is maximized. Returns “0” if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariablecontains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
{PROCESSMAXIMIZED：textVariable } – textVariableで指定されたタイトルのメインウィンドウを持つプロセスが最大化されている場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード(*)を使用できることに注意してください。たとえば、textVariable に「* notepad *」(引用符なし)が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

{WINDOWEXISTS:textVariable} – Returns “1” if a window with the title specified in textVariable exists. Returns “0” if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search
{WINDOWEXISTS：textVariable } – textVariableで指定されたタイトルのウィンドウが存在する場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード(*)を使用できることに注意してください。たとえば、textVariableに '* notepad *'(引用符なし)が含まれている場合、検索

will look for a window that has a title that contains 'notepad'.
「notepad」を含むタイトルを持つウィンドウを探します。

{WINDOWFOREGROUND:textVariable} – Returns “1” if a window with the title specified in textVariable is the foreground window. Returns “0” if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
{WINDOWFOREGROUND：textVariable } – textVariableで指定されたタイトルを持つウィンドウがフォアグラウンドウィンドウである場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード(*)を使用できることに注意してください。たとえば、textVariableに「* notepad *」(引用符なし)が含まれている場合、検索では「notepad」を含むタイトルを持つウィンドウが検索されます。

{WINDOWMINIMIZED:textVariable} – Returns “1” if a window with the title specified in textVariable is minimized. Returns “0” if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
{WINDOWMINIMIZED：textVariable } – textVariableで指定されたタイトルのウィンドウが最小化されている場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード(*)を使用できることに注意してください。たとえば、textVariableに「* notepad *」(引用符なし)が含まれている場合、検索では「notepad」を含むタイトルを持つウィンドウが検索されます。

{WINDOWMAXIMIZED:textVariable} – Returns “1” if a window with the title specified in textVariable is maximized. Returns “0” if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
{WINDOWMAXIMIZED：textVariable } – textVariableで指定されたタイトルのウィンドウが最大化されている場合、「1」を返します。そうでない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード(*)を使用できることに注意してください。たとえば、textVariableに「* notepad *」(引用符なし)が含まれている場合、検索では「notepad」を含むタイトルを持つウィンドウが検索されます。

{WINDOWCOUNT:textVariable} – Returns “1” or more if windows with the title specified in textVariable exist. Returns “0” if there are none. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search will look for windows that have a title that contains 'notepad'.
{WINDOWCOUNT : textVariable } – textVariableで 指定されたタイトルのウィンドウが存在する場合、「1」以上を返します。存在しない場合は「0」を返します。変更するウィンドウタイトルにはワイルドカード(*)を使用できることに注意してください。たとえば、textVariableに「* notepad *」(引用符なし)が含まれている場合、検索では「notepad」を含むタイトルを持つウィンドウが検索されます。

{WINDOWTITLEUNDERMOUSE} – Returns the title for the window that is currently located under the mouse.
{WINDOWTITLEUNDERMOUSE} –現在マウスの下にあるウィンドウのタイトルを返します。

{WINDOWPROCESSUNDERMOUSE} – Returns the process name for the window that is currently located under the mouse.
{WINDOWPROCESSUNDERMOUSE} –現在マウスの下にあるウィンドウのプロセス名を返します。

{CMDTARGETFOREGROUND} – Renders “1” if the current command’s target is the foreground window. Renders “0” if the target is not the foreground window or if the target does not exist.
{CMDTARGETFOREGROUND} –現在のコマンドのターゲットがフォアグラウンドウィンドウの場合、「1」をレンダリングします。ターゲットがフォアグラウンドウィンドウでない場合、またはターゲットが存在しない場合、「0」をレンダリングします。

{CMDTARGETMINIMIZED} – Renders “1” if the current command’s target is minimized. Renders “0” if the target is not minimized or if the target does not exist.
{CMDTARGETMINIMIZED} –現在のコマンドのターゲットが最小化されている場合、「1」をレンダリングします。ターゲットが最小化されていない場合、またはターゲットが存在しない場合、「0」をレンダリングします。

{CMDTARGETMAXIMIZED} – Renders “1” if the current command’s target is maximized. Renders “0” if the target is not maximized or if the target does not exist.
{CMDTARGETMAXIMIZED} –現在のコマンドのターゲットが最大化されている場合、「1」をレンダリングします。ターゲットが最大化されていない場合、またはターゲットが存在しない場合、「0」をレンダリングします。

{MOUSESCREENX} – Returns the X coordinate of the mouse position as it relates to the screen. Zero would be the top-left corner of the screen.
{MOUSESCREENX} –画面に関連するマウス位置のX座標を返します。ゼロは、画面の左上隅になります。

{MOUSESCREENY} – Returns the Y coordinate of the mouse position as it relates to the screen. Zero would be the top-left corner of the screen.
{MOUSESCREENY} –画面に関連するマウス位置のY座標を返します。ゼロは、画面の左上隅になります。

{MOUSEWINDOWX} - Returns the X coordinate of the mouse position as it relates to the active window. Zero would be the top-left corner of the window.
{MOUSEWINDOWX} -アクティブウィンドウに関連するマウス位置のX座標を返します。ゼロは、ウィンドウの左上隅になります。

{MOUSEWINDOWY} - Returns the Y coordinate of the mouse position as it relates to the active window. Zero would be the top-left corner of the window.
{MOUSEWINDOWY} -アクティブウィンドウに関連するマウス位置のY座標を返します。ゼロは、ウィンドウの左上隅になります。

{CAPSLOCKON} - Returns “1” if the caps lock key is locked. Returns “0” if not.
{CAPSLOCKON}-Caps Lockキーがロックされている場合、「1」を返します。そうでない場合は「0」を返します。

{NUMLOCKON} - Returns “1” if the numlock key is locked. Returns “0” if not.
{NUMLOCKON} -numlockキーがロックされている場合、「1」を返します。そうでない場合は「0」を返します。

{SCROLLLOCKON} - Returns “1” if the scroll lock key is locked. Returns “0” if not.
{SCROLLLOCKON}- スクロールロックキーがロックされている場合、「1」を返します。そうでない場合は「0」を返します。

{CLIP} - This token will be replaced by whatever is in the Windows clipboard, as long as what is in the clipboard is a text value. Note: This value can also be set within VoiceAttack by using the 'Set a text value to the Windows clipboard' action.
{CLIP} -クリップボードにあるものがテキスト値である限り、このトークンはWindowsクリップボードにあるものに置き換えられます。注：この値は、「Windowsクリップボードにテキスト値を設定」アクションを使用してVoiceAttack内で設定することもできます。

{DICTATION} - This token will be replaced by whatever is in the dictation buffer, without any formatting. So, if you said, 'This is a test of the emergency broadcast system' and then later said 'this is only a test', the result would be 'This is a test of the emergency broadcast system this is only a test'.
{DICTATION} -このトークンは、フォーマットせずにディクテーションバッファにあるものに置き換えられます。したがって、「これは緊急放送システムのテストです」と言ってから、「これは単なるテストです」と言った場合、結果は「これは緊急放送システムのテストであり、これは単なるテストです」となります。

{DICTATIONON} - This token returns “1” if dictation mode is on, and “0” if it is off.
{DICTATIONON} -ディクテーションモードがオンの場合、このトークンは「1」を返し、オフの場合は「0」を返します。

{DICTATION:options} – This token is an attempt to offer some basic touch up to whatever is in the dictation buffer. The speech recognition engine may do all kinds of heinous things to your text (as you will see... lol ;))
First, let's talk about the options. The option examples below will be used with as if your dictation buffer was filled first by saying 'This is a test of the emergency broadcast system' and then later saying 'this is only a test'.
{DICTATION：options } –このトークンは、ディクテーションバッファにあるものに基本的なタッチを提供しようとする試みです。音声認識エンジンは、あなたのテキストにあらゆる種類の凶悪なことをするかもしれません（あなたが見るように...笑;））まず、オプションについて話しましょう。以下のオプションの例は、ディクテーションバッファが最初に「これは緊急放送システムのテストです」と言ってから、「これはテストに過ぎない」と言って使用されるように使用されます。

The options are:
オプションは次のとおりです。

PERIOD – This puts a period in for you at the end of each line (so you don't have to constantly say, 'period' to the speech engine). The rendered output would be:
PERIOD– これにより、各行の終わりにピリオドが挿入されます（したがって、スピーチエンジンに対して常に「ピリオド」と言う必要はありません）。レンダリングされた出力は次のようになります。

'This is a test of the emergency broadcast system. this is only a test.'.
「これは緊急放送システムのテストです。これは単なるテストです。」

CAPITAL – Capitalizes the first character of every line (since the speech engine may or may not do it for you... o_O). The rendered output would be:
CAPITAL– すべての行の最初の文字を大文字にします（音声エンジンがあなたのためにそれをするかもしれないし、しないかもしれないので... o_O）。レンダリングされた出力は次のようになります。

'This is a test of the emergency broadcast system This is only a test'.
「これは緊急放送システムのテストですこれは単なるテストです」

LATEST – Only display the last thing you said. In the example, you would get 'this is only a test'.
最新–最後に言ったことだけを表示します。この例では、「これは単なるテストです」と表示されます。

UPPERCASE – Converts all characters to upper case:
大文字–すべての文字を大文字に変換します。

'THIS IS A TEST OF THE EMERGENCY BROADCAST SYSTEM THIS IS ONLY A TEST'
「これは緊急放送システムのテストですこれはテストのみです」

LOWERCASE – Converts all characters to lower case:
LOWERCASE– すべての文字を小文字に変換します：

'this is a test of the emergency broadcast system this is only a test'
「これは緊急放送システムのテストであり、これは単なるテストです」

NEWLINE – Makes sure each phrase is on its own line (note that it won't show up this way if you write to the log... the log doesn't care about new lines). Rendered output would be: 'This is a test of the emergency broadcast system
NEWLINE– 各フレーズがそれぞれの行にあることを確認します（ログに書き込むと、この方法では表示されないことに注意してください... レンダリング出力は次のようになります。「これは緊急放送システムのテストです

this is only a test.'.
これは単なるテストです。」

SPACEX – Replace the, 'X' with an integer number, and that many extra spaces will be
SPACEX–「X」を整数に置き換えると、多くの余分なスペースが

placed at the end of each phrase. The default is one space. So, if you use
各フレーズの最後に配置されます。デフォルトは1スペースです。したがって、使用する場合

{DICTATION:SPACE4}, the rendered output will be
{DICTATION：SPACE4}、レンダリングされた出力は

'this is a test of the emergency broadcast system this is only a test'.
「これは緊急放送システムのテストであり、これは単なるテストです」。

You can use any or all of the options, in any order, separated by a colon. They are not case-sensitive. So, if you simply wanted to add a period to the end of each spoken dictation phrase and ensure that the first character is always capitalized, you would use
オプションのいずれかまたはすべてを、コロンで区切って任意の順序で使用できます。大文字と小文字は区別されません。したがって、口頭の各ディクテーションフレーズの最後にピリオドを追加し、最初の文字が常に大文字になるようにしたい場合は、

{DICTATION:PERIOD:CAPITAL}. The above example would be rendered as: 'This is a test of the emergency broadcast system. This is only a test.'.
{DICTATION：PERIOD：CAPITAL}。上記の例は、「これは緊急放送システムのテストです。これは単なるテストです。」

If you wanted to also make it so that a line feed is placed between lines, you can use {DICTATION:PERIOD:CAPITAL:NEWLINE}. You will then get: 'This is a test of the emergency broadcast system.
改行を行間に配置するようにしたい場合は、{DICTATION：PERIOD：CAPITAL：NEWLINE}を使用できます。その後、次のようになります。「これは緊急放送システムのテストです。

This is only a test.'.
これは単なるテストです。」

{EXP:expression} and {EXPDECINV:expression} –
{EXP：式 }および{EXPDECINV：式} –

UPDATE: If you plan on using the {EXP} token for decimal calculations, it is now recommended that you use {EXPDECINV} instead, especially if you plan on sharing your profile with others in different countries. The {EXPDECINV} token works exactly the same as the {EXP} token, except that the value will be rendered using the invariant culture. That means that your resulting decimal value will always be rendered with a point ('.') instead of a comma, (',').
更新：10進計算に{EXP}トークンを使用する予定の場合、特に異なる国の他のユーザーとプロファイルを共有する予定の場合は、代わりに{EXPDECINV}を使用することをお勧めします。{EXPDECINV}トークンは、値が不変カルチャを使用してレンダリングされることを除いて、{EXP}トークンとまったく同じように機能します。つまり、結果の10進値は常に、コンマ(',')ではなくポイント('.')でレンダリングされます。

Currently experimental, the expression token will evaluate a typed-out expression (parenthesis and all, for order of operation) and return the value as text. Comparison expressions can be used on both text and numeric values (including numeric variables such as Integer, Small Integer and Decimal). Arithmetic expressions can be performed on numeric values. Values intended to be used for numeric variables can then be converted using their various, 'Convert Text/Token' option in each of the set value screens (for use in condition blocks (note that the result values can be decimal values when dividing).
現在実験的であるため、式トークンは入力された式（括弧とすべて、操作順）を評価し、値をテキストとして返します。比較式は、テキストと数値（整数、小整数、小数などの数値変数を含む）の両方で使用できます。数値に対して算術式を実行できます。数値変数に使用する予定の値は、各設定値画面のさまざまな「テキスト/トークンの変換」オプションを使用して変換できます（条件ブロックで使用するためです（除算する場合、結果値は10進数値になる場合があります）。

IMPORTANT: {EXP} and {EXPDECINV} only accept decimal values expressed in what is called the, 'invariant culture'. That means that it will only accept decimal values using a point ('.') instead of a comma (','). So, if two and a half is expressed as, '2,5', you will need to enter the value as, '2.5' or your result will end up as an expression error.
重要：{EXP}および{EXPDECINV}は、「インバリアントカルチャ」と呼ばれるもので表現された10進値のみを受け入れます。つまり、コンマ(',')の代わりにポイント('.')を使用した10進値のみを受け入れます。したがって、2.5が「2,5」と表現されている場合、値を「2.5」と入力する必要があります。そうしないと、結果が式エラーとなります。

There are several ways to use expressions as outlined below.
以下に概説するように、式を使用する方法はいくつかあります。

Evaluating Arithmetic Expressions
算術式の評価

The expression token can be used to evaluate arithmetic expressions like so:
式トークンは、次のような算術式を評価するために使用できます。

{EXP: ((5 + 5) − 2) * 10} (evaluates to, '80')
{EXP：((5 + 5)−2)* 10}（評価、「80」）

Accepted arithmetic operators: +, −, * (multiplication), / (division), % (modulus).
使用可能な算術演算子：+、−、*（乗算）、/（除算）、%（モジュラス）。

This token also accepts tokens to be evaluated. So, if you have a text value myText1 and it is set to '100' and an integer variable myInt1 that is 200, you can have a mixed
このトークンは、評価されるトークンも受け入れます。したがって、テキスト値myText1があり、それが「100」に設定され、整数変数myInt1が200である場合、混合値を持つことができます

expression of, {EXP:({TXT:myText1} + {INT:myInt1}) * 2} that results in '600'.
{EXP：({TXT：myText1} + {INT：myInt1})* 2}の式は、結果として「600」になります。

Numeric Comparisons
数値比較

You can do numeric comparisons using expressions as well:
式を使用して数値比較を行うこともできます。

{EXP: {TXT:myText1} = {INT:myInt1} } (evaluates to, '0' (false)) Comparison expressions return either '1' for true, or '0' for false.
{EXP：{TXT：myText1} = {INT：myInt1}}（評価、「0」（偽））比較式は、真の場合は「1」、偽の場合は「0」を返します。

Accepted comparison operators: =, <, >, <= (less than or equal), >= (greater than or equal),
受け入れられる比較演算子：=、<、>、<=（以下）、> =（以上）、

<> (not equal).
<>（等しくない）。

You may also use, 'And', 'Or', 'Not':
「And」、「Or」、「Not」を使用することもできます。

{EXP: ('cat' = 'dog') And Not (20 = 30) } (evaluates to, '0' (false))
{EXP : ('cat' = 'dog') And Not (20 = 30)}(評価、 '0'(false))

{EXP: ('cat' = 'dog') Or (20 = 30) } (evaluates to, '0' (false))
{EXP : ('cat' = 'dog')または(20 = 30)}(評価、 '0'(false))

{EXP: Not ('cat' = 'dog') } (evaluates to, '1' (true))
{EXP : Not('cat' = 'dog')}(評価、 '1'(true))

Evaluating Text Comparisons
テキスト比較の評価

The expression token can evaluate text.
式トークンはテキストを評価できます。

{EXP: 'all your base are belong to us' = 'hello'} (evaluates to, '0' (false)).
{EXP : 'すべてのベースは私たちのものです' = 'hello'}(評価、 '0'(false))。

{EXP: 'all your base are belong to us' <> 'hello'} (evaluates to, '1' (true)).
{EXP : 'すべてのベースは私たちのものです' <> 'hello'}(評価、 '1'(true))。

Just like numeric comparisons, comparison expressions return either '1' for true, or '0' for false.
数値比較と同様に、比較式はtrueの場合は「1」、falseの場合は「0」を返します。

Note that you have to enclose the text in single quotes (the quotes even need to be around tokens if the token value itself does not have quotes). If the text contains single quotes, they need to be doubled up to be used in expression tokens:
テキストを一重引用符で囲む必要があることに注意してください(トークン値自体に引用符がない場合、引用符はトークンを囲む必要さえあります)。テキストに一重引用符が含まれる場合、式トークンで使用するには二重引用符で囲む必要があります。

{EXP:'catcher''s mitt' = 'pitcher''s mitt'} Text comparisons are not case sensitive.
{EXP : 'catcher''s mitt '=' pitcher ''s mitt'}テキスト比較では大文字と小文字は区別されません。

Accepted comparison operators: =, <, >, <= (less than or equal), >= (greater than or equal),
受け入れられる比較演算子:=、<、>、<=(以下)、> =(以上)、

<> (not equal). You may also use, 'And', 'Or', 'Not'.
<>（等しくない）。「And」、「Or」、「Not」を使用することもできます。

You can use, 'LIKE' as part of your text comparison expressions. The text that you are comparing to needs to have asterisks in various places in order to indicate the type of comparison to make (wildcards).
テキスト比較式の一部として「LIKE」を使用できます。比較するテキスト（ワイルドカード）を比較するために、さまざまな場所にアスタリスクを付ける必要があります。

Asterisks around the text indicate, 'contains':
テキストの周りのアスタリスクは、「含む」を示します。

{EXP: 'rocket ship' LIKE '*rocket*'} (evaluates to '1' (true) because the text contains, 'rocket')
{EXP: 'rocket ship' LIKE '* rocket *'}（テキストに 'rocket'が含まれているため、 '1'（true）に評価されます）

Asterisks at the end indicate, 'starts with' :
末尾のアスタリスクは、「で始まる」を示します。

{EXP: 'rocket ship' LIKE 'ship*'} (evaluates to '0' (false), since the text does not start with, 'ship')
{EXP: 'ロケット船' LIKE '船*'}('0'（偽）と評価されます。テキストは '船'で始まらないため）

Asterisks at the beginning indicate, 'ends with':
先頭のアスタリスクは、「で終わる」を示します。

{EXP: 'rocket ship' LIKE '*ship'} (evaluates to '1' (true), since the text ends with 'ship') No asterisks indicate exact match (just like using, '=').
{EXP: 'ロケット船' LIKE '* ship'}（テキストが 'ship'で終わるため、 '1'（true）と評価されます）完全一致を示すアスタリスクはありません（ '='を使用するように）。

Other Expression Abilities
その他の表現能力

You can concatenate text by using, '+':
「+」を使用してテキストを連結できます。

{EXP: 'welcome' + ' ' + 'captain' } evaluates to 'welcome captain'. You can use, 'TRIM', 'SUBSTRING', 'LEN' and, 'IIF' (immediate, 'if')).
{EXP: 'welcome' + '' + 'captain'}は「welcome captain」と評価されます。「TRIM」、「SUBSTRING」、「LEN」、および「IIF」（イミディエート、「if」））を使用できます。

TRIM removes any blank spaces around text:
TRIMは、テキストの周囲の空白スペースを削除します。

{EXP: TRIM(' my trimmed text ') } evaluates to 'my trimmed text'
{EXP：TRIM（ 'my trimmed text'）}は 'my trimmed text'と評価されます

SUBSTRING(start position, length) gets just the portion of the text you are wanting:
SUBSTRING（start position、length）は、必要なテキストの一部のみを取得します。

{EXP: SUBSTRING('let the good times roll', 9, 4) } evaluates to, 'good'. Note that the start is 1-based (not zero-based).
{EXP：SUBSTRING（ 'let the good times roll'、9、4）}は、 'good'と評価されます。開始は1ベース（ゼロベースではない）であることに注意してください。

LEN gives the length of the text:
LENはテキストの長さを示します。

{EXP: LEN('let the good times roll') } evaluates to 23.
{EXP：LEN（ 'let the good times roll'）}は23と評価されます。

IIF(expression, true part, false part) allows you to evaluate an expression and get one of two values depending on the result:
IIF（expression、true part、false part）を使用すると、式を評価し、結果に応じて2つの値のいずれかを取得できます。

{EXP: IIF('cat' = 'dog', 10, 20) } evaluates to, '20'
{EXP：IIF（ 'cat' = 'dog'、10、20）}は '20'と評価されます

{EXP: IIF('cat' <> 'dog', 10, 20) } evaluates to, '10' myCondition1 is 100 and myCondition2 is 200 :
{EXP：IIF（ 'cat' <> 'dog'、10、20）}は、'10 'と評価されます。myCondition1は100で、myCondition2は200です。

{EXP: IIF({INT:myInt1} > {INT:myInt2}, 'Blue', 'Green') } evaluates to, 'Green'.
{EXP：IIF（{INT：myInt1}> {INT：myInt2}、 'Blue'、 'Green'）}は、 'Green'と評価されます。

VoiceAttack State Tokens
VoiceAttack状態トークン

State tokens will allow you to test various conditions that are occurring within VoiceAttack. These can be handy by allowing you to modify the flow of your command actions based on the state of your devices. For instance, you can check (with a conditional (if) statement) if the right mouse button is down when I say, 'fire weapons', missiles are fired instead of photons (lol). Also, if the 'x' button is down, fire both photons and missiles. You can also monitor the position of your sliders or X, Y and Z positions of the joystick itself to create some interesting, 'triggers' using loops and conditional (if) statements.
状態トークンを使用すると、VoiceAttack内で発生しているさまざまな条件をテストできます。これらは、デバイスの状態に基づいてコマンドアクションのフローを変更できるため、便利です。たとえば、「武器を発射する」と言ったときにマウスの右ボタンが押されているかどうかを条件付き(if)ステートメントで確認できます。光子の代わりにミサイルが発射されます(笑)。また、「x」ボタンが押されている場合は、光子とミサイルの両方を発射します。スライダーの位置またはジョイスティック自体のX、Y、Z位置を監視して、ループと条件(if)ステートメントを使用して、興味深い「トリガー」を作成することもできます。

{STATE_KEYSTATE:key} - This token checks to see if a particular key is pressed down or not. The, 'key' parameter can be any key you can type in a token: 'A', 'B', 'C', 'ß', 'ö', 'ñ', 'ç', as well as keys you can't type in a token: ENTER, TAB, LCTRL, ARROWR (see section later in this document titled, 'Key State Token Parameter Values' for the full list).
{STATE_KEYSTATE:key } -このトークンは、特定のキーが押されているかどうかを確認します。「key」パラメーターには、トークンに入力できる任意のキーを指定できます。「A」、「B」、「C」、「ß」、「ö」、「ñ」、「ç」、およびキートークンを入力することはできません:ENTER、TAB、LCTRL、ARROWR(完全なリストについては、このドキュメントの「キー状態トークンパラメータ値」というタイトルのセクションを参照してください)。

So, if you want to test to see if the F10 key is down, just use the following token:
したがって、F10キーがダウンしているかどうかをテストする場合は、次のトークンを使用します。

{STATE_KEYSTATE:F10}. To test for the letter 'A', just use this token:
{STATE_KEYSTATE:F10}。文字「A」をテストするには、次のトークンを使用します。

{STATE_KEYSTATE:A}. If a keyboard key is down, the replaced value of the token will be "1". If the key is not down, the value will be "0".
{STATE_KEYSTATE:A}。キーボードのキーが押されている場合、置き換えられたトークンの値は「1」になります。キーがダウンしていない場合、値は「0」になります。

{STATE_ANYKEYDOWN} - This token checks to see if any keyboard key is currently pressed. If any keyboard key is down, the replaced value of the token will be "1". If there are no keys pressed down, the value will be "0".
{STATE_ANYKEYDOWN} -このトークンは、現在キーボードキーが押されているかどうかを確認します。キーボードのキーが押されている場合、トークンの置き換えられた値は「1」になります。押されたキーがない場合、値は「0」になります。

{STATE_LEFTMOUSEBUTTON}
{STATE_LEFTMOUSEBUTTON}

{STATE_RIGHTMOUSEBUTTON}
{STATE_RIGHTMOUSEBUTTON}

{STATE_MIDDLEMOUSEBUTTON}
{STATE_MIDDLEMOUSEBUTTON}

{STATE_FORWARDMOUSEBUTTON}
{STATE_FORWARDMOUSEBUTTON}

{STATE_BACKMOUSEBUTTON} – Each of these tokens test to see if a mouse button is being pressed. If you want to test for the right mouse button, use token
{STATE_BACKMOUSEBUTTON} –これらの各トークンは、マウスボタンが押されているかどうかをテストします。マウスの右ボタンをテストする場合は、トークンを使用します

{STATE_RIGHTMOUSEBUTTON}. If the mouse button is pressed down, the replaced value will be ″1″. If the mouse button is not pressed, the value will be ″0″.
{STATE_RIGHTMOUSEBUTTON}。マウスボタンが押された場合、置き換えられた値は「1」になります。マウスボタンが押されていない場合、値は「0」になります。

{STATE_ANYMOUSEDOWN} – This token checks to see if any of the five standard mouse buttons are currently pressed. If any mouse button is down, the replaced value of the token will be ″1″. If there are no mouse buttons pressed down, the value will be ″0″.
{STATE_ANYMOUSEDOWN} –このトークンは、5つの標準マウスボタンのいずれかが現在押されているかどうかを確認します。いずれかのマウスボタンが押されている場合、置き換えられたトークンの値は「1」になります。マウスボタンが押されていない場合、値は「0」になります。

{STATE_MOUSESHORTCUTS} – This token tests to see if VoiceAttack's mouse button shortcuts are enabled. If they are enabled, the replaced value will be ″1″. Otherwise, the replaced value will be ″0″.
{STATE_MOUSESHORTCUTS} –このトークンは、VoiceAttackのマウスボタンショートカットが有効になっているかどうかをテストします。有効な場合、置き換えられる値は「1」になります。それ以外の場合、置き換えられる値は「0」になります。

{STATE_LISTENING} – This token tests to see if VoiceAttack is, 'listening'. If, 'listening' is on, the replaced value will be ″1″. If, 'listening' is off, the replaced value will be ″0″.
{STATE_LISTENING} –このトークンは、VoiceAttackが「リスニング」しているかどうかをテストします。「リスニング」がオンの場合、置き換えられる値は「1」になります。「listening」がオフの場合、置き換えられる値は「0」になります。

{STATE_SHORTCUTS} – This token tests to see if VoiceAttack's keyboard shortcuts are enabled. If shortcuts are enabled, the replaced value will be ″1″. Otherwise, the replaced value will be ″0″.
{STATE_SHORTCUTS} –このトークンは、VoiceAttackのキーボードショートカットが有効になっているかどうかをテストします。ショートカットが有効になっている場合、置き換えられる値は「1」になります。それ以外の場合、置き換えられる値は「0」になります。

{STATE_JOYSTICKSHORTCUTS} – This token tests to see if VoiceAttack's joystick button shortcuts are enabled. If shortcuts are enabled, the replaced value will be ″1″. Otherwise, the replaced value will be ″0″.
{STATE_JOYSTICKSHORTCUTS} –このトークンは、VoiceAttackのジョイスティックボタンショートカットが有効になっているかどうかをテストします。ショートカットが有効になっている場合、置き換えられる値は「1」になります。それ以外の場合、置き換えられる値は「0」になります。

{STATE_CPU:coreNumber}
{STATE_CPU：coreNumber}

{STATE_CPU} – These will return your cpu usage. {STATE_CPU} will return the average for all cores. The value returned will be from "0" to "100". {STATE_CPU:coreNumber} will allow you to specify a particular core. For instance, {STATE_CPU:5} will get the cpu usage for core 5.
{STATE_CPU} –これらはCPU使用量を返します。{STATE_CPU}は、すべてのコアの平均を返します。返される値は「0」から「100」です。{STATE_CPU：coreNumber}を使用すると、特定のコアを指定できます。たとえば、{STATE_CPU：5}はコア5のCPU使用率を取得します。

{STATE_RAMTOTAL} – This will return the total RAM on your system in bytes.
{STATE_RAMTOTAL} –システムの合計RAMをバイト単位で返します。

{STATE_RAMAVAILABLE} – This will return the available RAM on your system in bytes.
{STATE_RAMAVAILABLE} –システムで使用可能なRAMをバイト単位で返します。

{STATE_FILEEXISTS:textVariable} – This will return "1" if the file indicated in the text variable exists, or "0" if it does not.
{STATE_FILEEXISTS：textVariable } –テキスト変数で指定されたファイルが存在する場合は「1」を返し、存在しない場合は「0」を返します。

{STATE_DIRECTORYEXISTS:textVariable} – This will return "1" if the directory indicated in the text variable exists, or "0" if it does not.
{STATE_DIRECTORYEXISTS：textVariable } –テキスト変数で指定されたディレクトリが存在する場合は「1」を返し、存在しない場合は「0」を返します。

{STATE_DIRECTORYHASFILES:textVariable} – This will return "1" if the directory
{STATE_DIRECTORYHASFILES：textVariable } –ディレクトリが

indicated in the text variable has files in it, or "0" if it does not. Note that if the directory does not exist, "0" will be returned.
テキスト変数に示されているファイルにはファイルがあり、含まれていない場合は「0」です。ディレクトリが存在しない場合、「0」が返されることに注意してください。

{STATE_AUDIOLEVEL} – This token indicates the currently reported audio level from the speech engine. The replaced value will be from “0” to “100”.
{STATE_AUDIOLEVEL} –このトークンは、音声エンジンから現在報告されている音声レベルを示します。置き換えられる値は「0」から「100」です。

{STATE_AUDIOLASTFILE} – This token will render the path of the last audio file that is played.
{STATE_AUDIOLASTFILE} –このトークンは、最後に再生されるオーディオファイルのパスをレンダリングします。

{STATE_AUDIOCOUNT} – This returns the number of all currently-playing audio files. If legacy audio mode is on this value will always be, “0”.
{STATE_AUDIOCOUNT} –現在再生中のすべてのオーディオファイルの数を返します。レガシーオーディオモードがオンの場合、この値は常に「0」になります。

{STATE_AUDIOCOUNT: variableName / value } – This returns the number of currently- playing instances of an audio file with a given file path. The parameter can be a text variable name (without quotes: {STATE_AUDIOCOUNT:mySoundVariable}, or it can be a token or a literal if contained within double quotes:
{STATE_AUDIOCOUNT：variableName / value } –これは、指定されたファイルパスを持つオーディオファイルの現在再生中のインスタンスの数を返します。パラメーターはテキスト変数名（引用符なしの場合：{STATE_AUDIOCOUNT：mySoundVariable}、または二重引用符で囲まれている場合はトークンまたはリテラルの場合があります：

{STATE_AUDIOCOUNT:″{TXT:someTextVariable}″} or
{STATE_AUDIOCOUNT：″{TXT：someTextVariable}″}または

{STATE_AUDIOCOUNT:″C:¥Sounds¥Robot.wav″}.
{STATE_AUDIOCOUNT：″C：¥ Sounds ¥ Robot.wav″}。

Notes: Since sounds run asynchronously in VoiceAttack, there is a slight chance that if you use this token IMMEDIATELY after executing a ‘Play Sound’ action the file may not yet have had a chance to queue or load up and will not be included in the count. This is technically correct, but may not be a proper count depending on what you are trying to accomplish.
注：音がVoiceAttackで非同期に実行されるため、わずかなチャンスがあることを、あなたはこのトークンを使用する場合は、すぐに実行した後、ファイルがまだキューまたはロードアップする機会を持っていない可能性があり、中には含まれませんアクション「サウンド再生」をカウント。これは技術的には正しいですが、達成しようとしている内容によっては適切なカウントではない場合があります。

This token will allow you to put a literal in that contains colons (for file paths). Use with caution (or use variables/tokens).
このトークンを使用すると、コロンを含むリテラル（ファイルパス用）を配置できます。注意して使用してください（または変数/トークンを使用してください）。

If legacy audio mode is on this value will always be, “0”.
レガシーオーディオモードがオンの場合、この値は常に「0」になります。

{STATE_AUDIOPOS: variableName / value } – This returns the position of currently– playing audio file with a given file path, expressed as seconds. The parameter can be a text variable name (without quotes: {STATE_AUDIOPOS:mySoundVariable}, or it can be a token or a literal if contained within double quotes: {STATE_AUDIOPOS:″{TXT:someText}″} or {STATE_AUDIOPOS:″C:¥Sounds¥TargetEwoks.wav″}.
{STATE_AUDIOPOS：variableName / value } –これは、指定されたファイルパスを持つ現在再生中のオーディオファイルの位置を秒数で返します。パラメーターはテキスト変数名（引用符なし：{STATE_AUDIOPOS：mySoundVariable}）、または二重引用符内に含まれる場合はトークンまたはリテラルにすることができます：{STATE_AUDIOPOS：″{TXT：someText}″}または{STATE_AUDIOPOS：″C ：¥ Sounds ¥ TargetEwoks.wav ″}。

Notes: Since sounds run asynchronously in VoiceAttack, there is a slight chance that if you use this token IMMEDIATELY after executing a ‘Play Sound’ action the file may not yet have had a chance to queue or load up and will return a position of “0”. This is technically correct, but may not be a proper value depending on what you are trying to accomplish.
注：音がVoiceAttackで非同期に実行されるため、わずかなチャンスがあることを、あなたはこのトークンを使用する場合は、すぐに実行した後、ファイルがまだキューまたはロードアップする機会を持っていないかもしれないと「の位置を返しますアクション「サウンド再生」を0」。これは技術的には正しいですが、達成しようとしている内容によっては適切な値ではない場合があります。

Since you can run multiple instances of a sound file at once, if there is more than one instance currently playing, the rendered value will be, “0” (assumptions cannot be made on which instance to be chosen). To help with this, check the
サウンドファイルの複数のインスタンスを一度に実行できるため、現在再生中のインスタンスが複数ある場合、レンダリング値は「0」になります（どのインスタンスを選択するかは想定できません）。これを支援するには、

{STATE_AUDIOCOUNT:variable} token outlined above prior to using the
上記の{STATE_AUDIOCOUNT：variable}トークンを使用する前に

{STATE_AUDIOPOS} token.
{STATE_AUDIOPOS}トークン。

If no instances of the indicated file are currently playing, “0” will be returned.
指定されたファイルのインスタンスが現在再生されていない場合、「0」が返されます。

This token will allow you to put a literal in that contains colons (for file paths). Use with caution (or use variables/tokens).
このトークンを使用すると、コロンを含むリテラル（ファイルパス用）を配置できます。注意して使用してください（または変数/トークンを使用してください）。

If legacy audio mode is on this value will always be, “0”.
レガシーオーディオモードがオンの場合、この値は常に「0」になります。

{STATE_DEFAULTPLAYBACK} – This returns the device name of the default multimedia audio playback device as indicated by Windows. Note that there is a very minor memory leak when accessing the multimedia device property store, so this will be reflected in VoiceAttack (using this token sparingly will not present a problem… running it over and over in a loop will chew up memory… the search for a better way continues).
{STATE_DEFAULTPLAYBACK}–Windows が示すデフォルトのマルチメディアオーディオ再生デバイスのデバイス名を返します。マルチメディアデバイスプロパティストアへのアクセス時に非常に小さなメモリリークがあるため、これはVoiceAttackに反映されます（このトークンを控えめに使用しても問題は発生しません。ループで繰り返し実行するとメモリが噛みついてしまいます…検索より良い方法が続きます）。

{STATE_DEFAULTPLAYBACKCOMMS} – This works just like
{STATE_DEFAULTPLAYBACKCOMMS} –これは次のように機能します

{STATE_DEFAULTPLAYBACK}, except the default communications playblack device name will be rendered.
{STATE_DEFAULTPLAYBACK}。ただし、デフォルトの通信playblackデバイス名がレンダリングされます。

{STATE_DEFAULTRECORDING} – This works just like {STATE_DEFAULTPLAYBACK},
{STATE_DEFAULTRECORDING} –これは{STATE_DEFAULTPLAYBACK}と同様に機能します。

except the default multimedia recording device name will be rendered.
デフォルトのマルチメディア記録デバイス名がレンダリングされることを除きます。

{STATE_DEFAULTRECORDINGCOMMS} – This works just like
{STATE_DEFAULTRECORDINGCOMMS} –これは次のように機能します

{STATE_DEFAULTPLAYBACK}, except the default communications recording device name will be rendered.
{STATE_DEFAULTPLAYBACK}。ただし、デフォルトの通信記録デバイス名がレンダリングされます。

{STATE_SYSDIR} – This renders the path of the system directory (e.g. ‘C:¥Windows¥System32’).
{STATE_SYSDIR} –これにより、システムディレクトリのパスがレンダリングされます（例：’C：¥ Windows ¥ System32’）。

{STATE_WINDIR} – This renders the path of the Windows directory (e.g. ‘C:¥Windows’).
{STATE_WINDIR} –これは、Windowsディレクトリのパスをレンダリングします（例：’C：¥ Windows’）。

{STATE_ENV:textValue} – This renders the Windows environment variable specified in, ‘textValue’. For example, ‘{STATE_ENV:programfiles}’ would (usually) render ‘C:¥Program Files’.
{STATE_ENV：textValue } –これは、 'textValue'で指定されたWindows環境変数をレンダリングします。たとえば、「{STATE_ENV：programfiles}」は（通常）「C：¥ Program Files」をレンダリングします。

{STATE_CULTURE} – This renders the default user locale of the system.
{STATE_CULTURE} –システムのデフォルトのユーザーロケールを表示します。

{STATE_UICULTURE} – This renders the default user interface language.
{STATE_UICULTURE} –デフォルトのユーザーインターフェイス言語をレンダリングします。

{STATE_SYSVOL} – This returns the system volume (default playback device) rendered as a value from “0” to “100”.
{STATE_SYSVOL} –これは、「0」から「100」までの値としてレンダリングされたシステムボリューム（デフォルトの再生デバイス）を返します。

{STATE_SYSMUTE} – If the system volume (default playback device) is muted, this token is rendered as “1”. If not, the value is rendered as “0”.
{STATE_SYSMUTE} –システムボリューム（デフォルトの再生デバイス）がミュートされている場合、このトークンは「1」としてレンダリングされます。そうでない場合、値は「0」としてレンダリングされます。

{STATE_MICVOL} – This returns the microphone volume (default recording device) rendered as a value from “0” to “100”.
{STATE_MICVOL} –これは、「0」から「100」までの値としてレンダリングされたマイクの音量（デフォルトの録音デバイス）を返します。

{STATE_MICMUTE} – If the microphone volume (default recording device) is muted, this token is rendered as “1”. If not, the value is rendered as “0”.
{STATE_MICMUTE} –マイクの音量（デフォルトの録音デバイス）がミュートされている場合、このトークンは「1」としてレンダリングされます。そうでない場合、値は「0」としてレンダリングされます。

{STATE_APPVOL: variableName / value} – This renders a value between “0” and “100”
{STATE_APPVOL：variableName / value } –これにより、「0」から「100」の間の値がレンダリングされます

based on the indicated app volume as it relates to the System Volume Mixer. If the volume cannot be accessed (app closed or no audio is playing for instance), a value of ″-1″ will be returned. The parameter for this token should be the window title, process name or window class name of the target application (see “Set Audio Level” section of the, “Other Stuff” screen for more information on targeting the application). The parameter can be a text variable name (without quotes: {STATE_APPVOL:myTextVariable}, or it can be a token or a literal if contained within double quotes: {STATE_APPVOL:″*windows media*″} or
System Volume Mixerに関連する指定されたアプリのボリュームに基づきます。ボリュームにアクセスできない場合（アプリを閉じるか、オーディオが再生されていない場合など）、「-1」の値が返されます。このトークンのパラメーターは、ターゲットアプリケーションのウィンドウタイトル、プロセス名、またはウィンドウクラス名である必要があります（アプリケーションのターゲット設定の詳細については、「その他のもの」画面の「オーディオレベルの設定」セクションを参照してください）。パラメーターはテキスト変数名（引用符なしの場合：{STATE_APPVOL：myTextVariable}、または二重引用符内に含まれる場合はトークンまたはリテラルの場合：{STATE_APPVOL：″* windows media *″}または

{STATE_APPVOL:″{TXT:myTextVariable}″}.
{STATE_APPVOL：″{TXT：myTextVariable}″}。

{STATE_APPMUTE: variableName / value} – This works exactly the same as the
{STATE_APPMUTE：variableName / value }– これは、

{STATE_APPVOL} token above, except this renders a value of “1” if the indicated application is muted as it relates to the System Volume Mixer, and “0” if the application is not muted. “-1” is rendered if the information cannot be accessed.
上記の{STATE_APPVOL}トークン。ただし、指定されたアプリケーションがSystem Volume Mixerに関連してミュートされている場合は「1」、アプリケーションがミュートされていない場合は「0」の値をレンダリングします。情報にアクセスできない場合、「-1」が表示されます。

{STATE_SPEECHDEVICEMUTE} – This token tests if the recording device that the speech engine is currently using is muted. If the device is muted, “1” will be rendered. Otherwise, the rendered value will be “0”.
{STATE_SPEECHDEVICEMUTE} –このトークンは、音声エンジンが現在使用している録音デバイスがミュートされているかどうかをテストします。デバイスがミュートされている場合、「1」がレンダリングされます。それ以外の場合、レンダリングされた値は「0」になります。

{STATE_SPEECHDEVICEVOL} – This returns the volume of the recording device that the speech engine is currently using rendered as a value from “0” to “100”.
{STATE_SPEECHDEVICEVOL} –これは、音声エンジンが現在使用している録音デバイスの音量を「0」から「100」までの値として返します。

{STATE_SPEECHACTIVE} – This token tests to see if the speech engine is detecting speech. If speech is currently detected, the replaced value will be ″1″. If, not, the replaced value will be ″0″.
{STATE_SPEECHACTIVE} –このトークンは、音声エンジンが音声を検出しているかどうかをテストします。音声が現在検出されている場合、置き換えられる値は「1」になります。そうでない場合、置き換えられる値は「0」になります。

{STATE_VA_VERSION} – Displays the full VoiceAttack version: “1.5.12.15”.
{STATE_VA_VERSION} –完全なVoiceAttackバージョンを表示します：「1.5.12.15」。

{STATE_VA_VERSION_MAJOR} - Displays the version major component: “1”.
{STATE_VA_VERSION_MAJOR} -バージョンメジャーコンポーネント「1」を表示します。

{STATE_VA_VERSION_MINOR} - Minor component: “5”.
{STATE_VA_VERSION_MINOR} -マイナーコンポーネント:「5」。

{STATE_VA_VERSION_BUILD} - Build component: “12”.
{STATE_VA_VERSION_BUILD} -ビルドコンポーネント:「12」。

{STATE_VA_VERSION_REVISION} - Revision component: “15”.
{STATE_VA_VERSION_REVISION} -改訂コンポーネント:「15」。

{STATE_VA_VERSION_ISRELEASE} - Returns “1” if the version is a release version. “0” if it is not (beta release).
{STATE_VA_VERSION_ISRELEASE} -バージョンがリリースバージョンの場合、「1」を返します。そうでない場合は「0」（ベータリリース）。

{STATE_VA_VERSION_COMPARE:textVariable} - Returns “1” if the VoiceAttack version number is greater than or equal to the version number indicated in the text variable.
{STATE_VA_VERSION_COMPARE:textVariable } -VoiceAttackのバージョン番号がテキスト変数で示されるバージョン番号以上の場合、「1」を返します。

“0” if the indicated version is earlier than the VoiceAttack version number.
指定されたバージョンがVoiceAttackのバージョン番号より前の場合は「0」。

{STATE_VA_PLUGINSENABLED} - Returns “1” if plugin support is enabled. “0” if not.
{STATE_VA_PLUGINSENABLED}- プラグインサポートが有効な場合、「1」を返します。そうでない場合は「0」。

{STATE_VA_NESTEDTOKENSENABLED} - Returns “1” if the nested tokens option is enabled. “0” if not.
{STATE_VA_NESTEDTOKENSENABLED}- ネストされたトークンオプションが有効な場合、「1」を返します。そうでない場合は「0」。

Joystick State Token Reference
ジョイスティックステートトークンリファレンス

This section is for the joystick state tokens. Everything below the buttons and POV are lifted directly from DirectX states. Each state value may or may not be available for your device, so, your mileage may vary.
このセクションは、ジョイスティック状態トークン用です。ボタンとPOVの下にあるものはすべて、DirectXの状態から直接持ち上げられます。各州の値は、お使いのデバイスで使用できる場合と使用できない場合があります。そのため、走行距離は異なる場合があります。

{STATE_JOYSTICKBUTTONS} - This token has been replaced with
{STATE_JOYSTICKBUTTONS} -このトークンは置き換えられました

{STATE_JOYSTICKSHORTCUTS} (see above).
{STATE_JOYSTICKSHORTCUTS}(上記を参照)。

{STATE_JOYSTICK1ENABLED}
{STATE_JOYSTICK1ENABLED}

{STATE_JOYSTICK2ENABLED} - these two tokens test whether or not joystick 1 or joystick 2 are enabled in VoiceAttack. If the joystick is enabled, the rendered value will be “1”. If the indicated joystick is not enabled, the value will be “0”.
{STATE_JOYSTICK2ENABLED} -これら2つのトークンは、VoiceAttackでジョイスティック1またはジョイスティック2が有効になっているかどうかをテストします。ジョイスティックが有効な場合、レンダリングされた値は「1」になります。指定されたジョイスティックが有効になっていない場合、値は「0」になります。

{STATE_JOYSTICK1BUTTON:buttonNumber}
{STATE_JOYSTICK1BUTTON:buttonNumber }

{STATE_JOYSTICK2BUTTON:buttonNumber} - these two tokens test to see if particular joystick's button is down or not. The, 'buttonNumber' parameter is the number of the button on the desired stick. To test if button 10 is down on joystick 2, just use this token:
{STATE_JOYSTICK2BUTTON:buttonNumber } -これら2つのトークンは、特定のジョイスティックのボタンが押されているかどうかをテストします。'buttonNumber'パラメーターは、目的のスティック上のボタンの番号です。ジョイスティック2でボタン10が押されているかどうかをテストするには、次のトークンを使用します。

{STATE_JOYSTICK2BUTTON:10}. Once again, if the button is down on the tested stick, the replaced value will be ″1″. If the button is not down, the value will be ″0″.
{STATE_JOYSTICK2BUTTON:10}。繰り返しますが、テストされたスティックでボタンが押されている場合、置き換えられた値は「1」になります。ボタンが押されていない場合、値は「0」になります。

{STATE_JOYSTICK1ANYBUTTON}
{STATE_JOYSTICK1ANYBUTTON}

{STATE_JOYSTICK2ANYBUTTON} - these two tokens test whether or not any button is pressed on either joystick 1 or joystick 2. If any button is down, the rendered value will be “1”. If no buttons are pressed down, the value will be “0”.
{STATE_JOYSTICK2ANYBUTTON} -これらの2つのトークンは、ジョイスティック1またはジョイスティック2のいずれかのボタンが押されているかどうかをテストします。いずれかのボタンが押されている場合、レンダリング値は「1」になります。ボタンが押されていない場合、値は「0」になります。

Point of View (Hat) Controllers - (Note: there are up to 4 available POV controllers per stick)
視点(ハット)コントローラー-(注:スティックごとに最大4つのPOVコントローラーが利用可能です)

{STATE_JOYSTICK1POVENABLED}
{STATE_JOYSTICK1POVENABLED}

{STATE_JOYSTICK2POVENABLED} – this token indicates whether or not POV is enabled for the indicated stick. If enabled the replaced value will be “1”. Otherwise, the replaced value will be “0”.
{STATE_JOYSTICK2POVENABLED} –このトークンは、示されたスティックに対してPOVが有効になっているかどうかを示します。有効にすると、置き換えられる値は「1」になります。それ以外の場合、置き換えられる値は「0」になります。

{STATE_JOYSTICKXPOVYTYPE} – Use this token to find out how the POV is being used by VoiceAttack (as indicated in the joystick options on the options page). X indicates the stick number (1 or 2) and Y indicates the POV controller (1–4). So, to get the POV type for the second POV controller on stick 1, use: {STATE_JOYSTICK1POV2TYPE}. The replaced value will be one of the following:
{STATE_JOYSTICK X POV Y TYPE}– このトークンを使用して、POVがVoiceAttackでどのように使用されているかを確認します（オプションページのジョイスティックオプションに示されています）。Xはスティック番号（1または2）を示し、YはPOVコントローラー（1～4）を示します。したがって、スティック1の2番目のPOVコントローラーのPOVタイプを取得するには、{STATE_JOYSTICK1POV2TYPE}を使用します。置き換えられる値は次のいずれかです。

“–1” – POV not available.
「–1」–POVは使用できません。

“0” – POV available, not turned on in settings.
「0」–POVが利用可能、設定でオンになっていない。

“1” – POV acts as on/off switch (any direction will cause the POV to indicate as switched. “2” – POV acts as two–directional switch (up or down).
「1」–POVはオン/オフスイッチとして機能します（どの方向でもPOVが切り替えられたことを示します。「2」–POVは双方向スイッチ（上または下）として機能します。

“3” – POV acts as two–directional switch (left or right).
「3」–POVは双方向スイッチとして機能します（左または右）。

“4” – POV acts as four–directional switch (up, down, left, right).
「4」–POVは4方向スイッチ（上、下、左、右）として機能します。

“8” – POV acts as eight–directional switch (up, up right, right, down right, down, down left, left, up left).
「8」–POVは8方向スイッチとして機能します（上、右上、右、下右、下、左下、左、左上）。

{STATE_JOYSTICKXPOVY} – This token is used to get the direction pressed by the POV
{STATE_JOYSTICK X POV Y } –このトークンは、POVによって押された方向を取得するために使用されます

controller. X indicates the stick (1 or 2). Y indicates the POV controller (1-4). So, to get the position value for stick 1, POV controller 2, use: {STATE_JOYSTICK1POV2}.
コントローラ。Xはスティック(1または2)を示します。YはPOVコントローラー(1～4)を示します。したがって、スティック1、POVコントローラー2の位置値を取得するには、{STATE_JOYSTICK1POV2}を使用します。

The replaced value will be "CENTER", "UP", "UPRIGHT", "RIGHT", "DOWNRIGHT", "DOWN", "DOWNLEFT", "LEFT", "UPLEFT" depending on the direction pushed, or, "-1" if the POV controller is unavailable. Note that certain directions will only be available due to the type of POV setup in options (see the token above for getting the POV type). For POV type "1" (POV is on/off switch), the only position that will be indicated and that you should test for is, "UP" (this is for speed reasons). For "2" (POV is up/down only), "UP" and "DOWN" are indicated. For "3" (POV is left/right only), "LEFT" and "RIGHT" are indicated. For "4" (four-direction), "LEFT", "RIGHT", "UP" and "DOWN" are indicated. For "8" (eight- direction), all eight directions are available.
置き換えられた値は、押された方向に応じて、「CENTER」、「UP」、「UPRIGHT」、「RIGHT」、「DOWNRIGHT」、「DOWN」、「DOWNLEFT」、「LEFT」、「UPLEFT」、または「- POVコントローラーが使用できない場合は1インチ。オプションでのPOVセットアップのタイプが原因で、特定の指示のみが利用可能になることに注意してください(POVタイプを取得するには、上記のトークンを参照してください)。POVタイプが「1」の場合(POVはオン/オフスイッチ)、示され、テストする必要がある唯一の位置は「UP」です(これは速度上の理由によるものです)。「2」(POVはアップ/ダウンのみ)の場合、「UP」と「DOWN」が示されます。「3」(POVは左/右のみ)の場合、「左」と「右」が示されます。「4」(4方向)の場合、「左」、「右」、「上」、「下」が表示されます。「8」(8方向)の場合、8方向すべてが使用可能です。

DirectX Joystick States (simple DirectX query, not tracked by VoiceAttack for managing events):
DirectXジョイスティックの状態(単純なDirectXクエリ、イベントを管理するためにVoiceAttackによって追跡されない):

{STATE_JOYSTICKXPOVY_NUMERIC} - This token is used to get the numeric value presented by the POV controller. X indicates the stick (1 or 2). Y indicates the POV controller (1-4). So, to get the position value for stick 1, POV controller 2, use:
{STATE_JOYSTICK X POV Y _NUMERIC} -このトークンは、POVコントローラーによって提示される数値を取得するために使用されます。Xはスティック(1または2)を示します。YはPOVコントローラー(1～4)を示します。したがって、スティック1、POVコントローラ2の位置値を取得するには、次を使用します。

{STATE_JOYSTICK1POV2}. The value will usually be "-1" (some drivers may report "65535") if the POV is centered, or "0" to "35999" as the POV is pressed in a direction. You can use this token if the VoiceAttack, 'switch' model does not fit your needs.
{STATE_JOYSTICK1POV2}。値は通常、POVが中央にある場合は「-1」(一部のドライバーは「65535」と報告する場合があります)、またはPOVが一方向に押されると「0」から「35999」になります。VoiceAttackの「スイッチ」モデルがニーズに合わない場合、このトークンを使用できます。

{STATE_JOYSTICK1X}
{STATE_JOYSTICK1X}

{STATE_JOYSTICK2X} - This token is used to indicate the X value of the indicated joystick. The replaced value is "0" at minimum and "65535" at maximum. This value will be "-1" if the stick is unavailable.
{STATE_JOYSTICK2X} -このトークンは、示されたジョイスティックのX値を示すために使用されます。置き換えられる値は、最小で「0」、最大で「65535」です。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1Y}
{STATE_JOYSTICK1Y}

{STATE_JOYSTICK2Y} - This token is used to indicate the Y value of the indicated joystick. The replaced value is “0” at minimum and “65535” at maximum. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2Y} -このトークンは、示されたジョイスティックのY値を示すために使用されます。置き換えられる値は、最小で「0」、最大で「65535」です。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1Z}
{STATE_JOYSTICK1Z}

{STATE_JOYSTICK2Z} - This token is used to indicate the Z value of the indicated joystick. The replaced value is “0” at minimum and “65535” at maximum. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2Z} -このトークンは、示されたジョイスティックのZ値を示すために使用されます。置き換えられる値は、最小で「0」、最大で「65535」です。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ROTATIONX}
{STATE_JOYSTICK1ROTATIONX}

{STATE_JOYSTICK2ROTATIONX} - This token is used to indicate the rotation X value of the indicated joystick. The replaced value is “0” at minimum and “65535” at maximum.
{STATE_JOYSTICK2ROTATIONX} -このトークンは、示されたジョイスティックの回転X値を示すために使用されます。置き換えられる値は、最小で「0」、最大で「65535」です。

This value will be “-1” if the stick is unavailable.
スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ROTATIONY}
{STATE_JOYSTICK1ROTATIONY}

{STATE_JOYSTICK2ROTATIONY} - This token is used to indicate the rotation Y value of the indicated joystick. The replaced value is “0” at minimum and “65535” at maximum.
{STATE_JOYSTICK2ROTATIONY} -このトークンは、示されたジョイスティックの回転Y値を示すために使用されます。置き換えられる値は、最小で「0」、最大で「65535」です。

This value will be “-1” if the stick is unavailable.
スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ROTATIONZ}
{STATE_JOYSTICK1ROTATIONZ}

{STATE_JOYSTICK2ROTATIONZ} – This token is used to indicate the rotation Z value of
{STATE_JOYSTICK2ROTATIONZ} –このトークンは、の回転Z値を示すために使用されます

the indicated joystick. The replaced value is “0” at minimum and “65535” at maximum. This value will be “-1” if the stick is unavailable.
指定されたジョイスティック。置き換えられる値は、最小で「0」、最大で「65535」です。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ACCELERATIONX}
{STATE_JOYSTICK1ACCELERATIONX}

{STATE_JOYSTICK2ACCELERATIONX} – This token is used to indicate the acceleration X value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2ACCELERATIONX} –このトークンは、示されたジョイスティックの加速度X値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ACCELERATIONY}
{STATE_JOYSTICK1ACCELERATIONY}

{STATE_JOYSTICK2ACCELERATIONY} – This token is used to indicate the acceleration Y value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2ACCELERATIONY} –このトークンは、示されたジョイスティックの加速度Y値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ACCELERATIONZ}
{STATE_JOYSTICK1ACCELERATIONZ}

{STATE_JOYSTICK2ACCELERATIONZ} – This token is used to indicate the acceleration Z value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2ACCELERATIONZ} –このトークンは、示されたジョイスティックの加速度Z値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1ANGULARACCELERATIONX}
{STATE_JOYSTICK1ANGULARACCELERATIONX}

{STATE_JOYSTICK2ANGULARACCELERATIONX} – This token is used to indicate the angular acceleration X value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2ANGULARACCELERATIONX} –このトークンは、示されたジョイスティックの角加速度X値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1ANGULARACCELERATIONY}
{STATE_JOYSTICK1ANGULARACCELERATIONY}

{STATE_JOYSTICK2ANGULARACCELERATIONY} – This token is used to indicate the angular acceleration Y value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2ANGULARACCELERATIONY} –このトークンは、示されたジョイスティックの角加速度Y値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1ANGULARACCELERATIONZ}
{STATE_JOYSTICK1ANGULARACCELERATIONZ}

{STATE_JOYSTICK2ANGULARACCELERATIONZ} – This token is used to indicate the angular acceleration Z value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2ANGULARACCELERATIONZ} –このトークンは、示されたジョイスティックの角加速度Z値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1ANGULARVELOCITYX}
{STATE_JOYSTICK1ANGULARVELOCITYX}

{STATE_JOYSTICK2ANGULARVELOCITYX} – This token is used to indicate the angular velocity X value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2ANGULARVELOCITYX} –このトークンは、示されたジョイスティックの角速度X値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1ANGULARVELOCITYY}
{STATE_JOYSTICK1ANGULARVELOCITYY}

{STATE_JOYSTICK2ANGULARVELOCITYY} – This token is used to indicate the angular velocity Y value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2ANGULARVELOCITYY} –このトークンは、示されたジョイスティックの角速度Y値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1ANGULARVELOCITYZ}
{STATE_JOYSTICK1ANGULARVELOCITYZ}

{STATE_JOYSTICK2ANGULARVELOCITYZ} – This token is used to indicate the angular velocity Z value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2ANGULARVELOCITYZ} –このトークンは、示されたジョイスティックの角速度Z値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1FORCEX}
{STATE_JOYSTICK1FORCEX}

{STATE_JOYSTICK2FORCEX} – This token is used to indicate the force X value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2FORCEX} –このトークンは、示されたジョイスティックの力X値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1FORCEY}
{STATE_JOYSTICK1FORCEY}

{STATE_JOYSTICK2FORCEY} – This token is used to indicate the force Y value of the
{STATE_JOYSTICK2FORCEY} –このトークンは、

indicated joystick. This value will be “–1” if the stick is unavailable.
示されたジョイスティック。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1FORCEZ}
{STATE_JOYSTICK1FORCEZ}

{STATE_JOYSTICK2FORCEZ} – This token is used to indicate the force Z value of the indicated joystick. This value will be “–1” if the stick is unavailable.
{STATE_JOYSTICK2FORCEZ} –このトークンは、示されたジョイスティックの強制Z値を示すために使用されます。スティックが使用できない場合、この値は「–1」になります。

{STATE_JOYSTICK1TORQUEX}
{STATE_JOYSTICK1TORQUEX}

{STATE_JOYSTICK2TORQUEX} - This token is used to indicate the torque X value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2TORQUEX} -このトークンは、示されたジョイスティックのトルクX値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1TORQUEY}
{STATE_JOYSTICK1TORQUEY}

{STATE_JOYSTICK2TORQUEY} - This token is used to indicate the torque Y value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2TORQUEY} -このトークンは、示されたジョイスティックのトルクY値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1TORQUEZ}
{STATE_JOYSTICK1TORQUEZ}

{STATE_JOYSTICK2TORQUEZ} - This token is used to indicate the torque Z value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2TORQUEZ} -このトークンは、示されたジョイスティックのトルクZ値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1VELOCITYX}
{STATE_JOYSTICK1VELOCITYX}

{STATE_JOYSTICK2VELOCITYX} - This token is used to indicate the velocity X value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2VELOCITYX} -このトークンは、示されたジョイスティックの速度X値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1VELOCITYY}
{STATE_JOYSTICK1VELOCITYY}

{STATE_JOYSTICK2VELOCITYY} - This token is used to indicate the velocity Y value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2VELOCITYY} -このトークンは、示されたジョイスティックの速度Y値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICK1VELOCITYZ}
{STATE_JOYSTICK1VELOCITYZ}

{STATE_JOYSTICK2VELOCITYZ} - This token is used to indicate the velocity Z value of the indicated joystick. This value will be “-1” if the stick is unavailable.
{STATE_JOYSTICK2VELOCITYZ} -このトークンは、示されたジョイスティックの速度Z値を示すために使用されます。スティックが使用できない場合、この値は「-1」になります。

{STATE_JOYSTICKXSLIDERY} - This token is used to indicate the slider value. X indicates the stick number (1 or 2) and Y indicates the control number (1 or 2). If the control is not available, the replaced value will be “-1”.
{STATE_JOYSTICK X SLIDER Y } -このトークンは、スライダーの値を示すために使用されます。Xはスティック番号（1または2）を示し、Yはコントロール番号（1または2）を示します。コントロールが使用できない場合、置き換えられる値は「-1」になります。

{STATE_JOYSTICKXACCELERATIONSLIDERY} - This token is used to indicate the acceleration slider value. X indicates the stick number (1 or 2) and Y indicates the control number (1 or 2). If the control is not available, the replaced value will be “-1”.
{STATE_JOYSTICK X ACCELERATIONSLIDER Y } -このトークンは、加速スライダーの値を示すために使用されます。Xはスティック番号（1または2）を示し、Yはコントロール番号（1または2）を示します。コントロールが使用できない場合、置き換えられる値は「-1」になります。

{STATE_JOYSTICKXFORCESLIDERY} - This token is used to indicate the force slider value. X indicates the stick number (1 or 2) and Y indicates the control number (1 or 2). If the control is not available, the replaced value will be “-1”.
{STATE_JOYSTICK X FORCESLIDER Y } -このトークンは、フォーススライダーの値を示すために使用されます。Xはスティック番号（1または2）を示し、Yはコントロール番号（1または2）を示します。コントロールが使用できない場合、置き換えられる値は「-1」になります。

{STATE_JOYSTICKXVELOCITYSLIDERY} - This token is used to indicate the velocity slider value. X indicates the stick number (1 or 2) and Y indicates the control number (1 or 2). If the control is not available, the replaced value will be “-1”.
{STATE_JOYSTICK X VELOCITYSLIDER Y } -このトークンは、速度スライダーの値を示すために使用されます。Xはスティック番号（1または2）を示し、Yはコントロール番号（1または2）を示します。コントロールが使用できない場合、置き換えられる値は「-1」になります。

VoiceAttack Path Tokens
VoiceAttackパストークン

VoiceAttack has a few tokens that can be used in places that require a file path (sound file locations and application locations). There are certain cases where it is helpful to keep certain files together, especially when it comes to sharing profiles.
VoiceAttackには、ファイルパス（サウンドファイルの場所とアプリケーションの場所）が必要な場所で使用できるトークンがいくつかあります。特にプロファイルの共有に関しては、特定のファイルをまとめておくと役立つ場合があります。

{VA_DIR} – This is the VoiceAttack installation directory.
{VA_DIR} –これはVoiceAttackのインストールディレクトリです。

{VA_ASSEMBLIES} – This is the VoiceAttack Shared¥Assemblies folder that is (should be) located in the VoiceAttack installation folder. Note that no check is made to see if the folder exists.
{VA_ASSEMBLIES} –これは、VoiceAttackインストールフォルダーにある(必要な)VoiceAttack Shared ¥ Assemblies フォルダーです。フォルダーが存在するかどうかを確認するチェックは行われないことに注意してください。

{VA_SOUNDS} – By default, this is the folder named, 'Sounds' under the VoiceAttack root directory. This is a place where you can store your VoiceAttack sound packs. If you do not want to use the 'Sounds' folder in the VoiceAttack installation directory, you can change this to be whatever folder you want it to be on the VoiceAttack Options page. Note that no check is made to see if this folder exists.
{VA_SOUNDS} –デフォルトでは、これはVoiceAttackルートディレクトリの下の「Sounds」という名前のフォルダーです。これは、VoiceAttackサウンドパックを保存できる場所です。VoiceAttackインストールディレクトリの「サウンド」フォルダを使用したくない場合は、VoiceAttackオプションページで任意のフォルダに変更できます。このフォルダが存在するかどうかを確認するチェックは行われないことに注意してください。

{VA_APPS} – This is the folder named, 'Apps' under the VoiceAttack root directory. This is the place where you can store apps that you use with VoiceAttack (.exe files) and VoiceAttack plugins (.dll files). Just like the, 'Sounds' folder, you can change the location of the VoiceAttack Apps folder in the VoiceAttack Options page. Note that no check is made to see if this folder exists.
{VA_APPS} –これは、VoiceAttackルートディレクトリの下の「Apps」という名前のフォルダーです。これは、VoiceAttack(.exeファイル)およびVoiceAttackプラグイン(.dllファイル)で使用するアプリを保存できる場所です。「サウンド」フォルダーと同様に、VoiceAttackオプションページでVoiceAttackアプリフォルダーの場所を変更できます。このフォルダが存在するかどうかを確認するチェックは行われないことに注意してください。

Quick Input, Variable Keypress and Hotkey Key Indicators
クイック入力、可変キープレス、ホットキーキーインジケーター

As explained earlier in this document, you can include special indicators in your Quick Input, keypress variable text and variable hotkey text to represent keys that do not have a character representation (such as, 'Enter', 'Tab', 'F1', etc.).
このドキュメントで前述したように、クイック入力、キープレス変数テキスト、および変数ホットキーテキストに特別なインジケーターを含めて、文字表現を持たないキー(「Enter」、「Tab」、「F1」、等。)。

Below is a list of the acceptable key indicators, some with a brief description. Remember that key indicators need to be enclosed in square brackets: []. At the bottom of this list is the Quick Input function list (currently only contains, 'PAUSE', but will grow as needed).
以下は、受け入れられる重要な指標のリストであり、簡単な説明もあります。重要なインジケータは角括弧で囲む必要があることに注意してください:[]。このリストの下部には、クイック入力機能リストがあります(現在は「PAUSE」のみが含まれていますが、必要に応じて拡大します)。

For information on what these are for, see the section titled, 'Quick Input' in the, 'Other Stuff' screen documentation, or the section detailing variable keypresses in the, 'Keypress' screen documentation.
これらの用途に関する情報については、「その他の項目」画面のドキュメントの「クイック入力」というタイトルのセクション、または「キープレス」画面のドキュメントの変数キー押下の詳細セクションを参照してください。

Note: Items below that are marked with a blue asterisk (*) are for Quick Input only. Items that are marked with a red asterisk (*) are for keypress variables only and are not available for Quick Input.
注：青いアスタリスク（＊）でマークされている以下の項目は、クイック入力専用です。赤いアスタリスク（＊）でマークされたアイテムは、キー入力変数専用であり、クイック入力では使用できません。

ENTER – presses the enter key TAB – presses the tab key
ENTER-Enterキーを押しますTAB-Tabキーを押します

ESC – presses the escape key ESCAPE – works the same as, 'esc' above BACK – press the backspace button BACKSPACE – works the same as, 'back' above
ESC-エスケープキーを押すESCAPE-同じように機能し、「esc」を押すBACK-バックスペースボタンを押す
BACKSPACE-同じように機能する、「戻る」を押す

SPACE – presses the space bar (note that you can also just have a space in your keypress variable or Quick Input value.
スペース-スペースバーを押します（キープレス変数またはクイック入力値にスペースを入れることもできます）。

SHIFTDOWN* – holds the left shift key down SHIFTUP* – releases the left shift key RSHIFTDOWN* – right shift if you need it RSHIFTUP*
SHIFTDOWN ＊ -左シフトキーを押したままSHIFTUP ＊ -左シフトキーを放しますRSHIFTDOWN ＊ -必要に応じて右シフトRSHIFTUP ＊

LSHIFTDOWN* – works the same as shiftdown LSHIFTUP* – works the same as shiftup
LSHIFTDOWN ＊ -シフトダウンと同じ動作LSHIFTUP ＊ - シフトアップと同じ動作

SHIFT* – press left shift key
SHIFT ＊ -左シフトキーを押します

LSHIFT* – press left shift key
LSHIFT ＊ -左シフトキーを押します

RSHIFT* – press right shift key
RSHIFT ＊ -右シフトキーを押します

ALTDOWN* – holds down the left alt key ALTUP* – releases the left alt key RALTDOWN* – holds down the right alt key RALTUP* – releases the right alt key LALTDOWN* – works the same as altdown LALTUP* – works the same as altup
ALTDOWN *は–左AltキーALTUPダウン保持* リリース左AltキーRALTDOWN – *を–右AltキーRALTUPを押したまま* –リリースを右AltキーLALTDOWN * – LALTUP altdownと同様に機能*を– altupと同じように動作します

ALT* – keypress variable – press left alt key LALT* – keypress variable – press left alt key RALT* – keypress variable – press right alt key
ALT * –キーを押す変数–左のaltキーを押すLALT * –キーを押す変数–左のaltキーを押すRALT * –キーを押す変数–右のaltキーを押す

CTRLDOWN* – holds down the left ctrl key
CTRLDOWN * –左のctrlキーを押したままにします

CTRLUP* – releases the left ctrl key RCTRLDOWN* – holds down right ctrl key RCTRLUP* – releases right ctrl key LCTRLDOWN*– works the same as ctrldown LCTRLUP* – works the same as ctrlup
CTRLUP * –左のctrlキーを解放RCTRLDOWN * –右のctrlキーを押し下げRCTRLUP * –右のctrlキーを解放LCTRLDOWN * –ctrldownと同じ働きLCTRLUP * –ctrlupと同じ働き

CTRL* – keypress variable only – press left ctrl key LCTRL* – keypress variable only – press left ctrl key RCTRL* – keypress variable only – press right ctrl key
CTRL * –キーを押す変数のみ–左Ctrlキーを押すLCTRL * –キーを押す変数のみ–左Ctrlキーを押すRCTRL * –キーを押す変数のみ–右Ctrlキーを押す

WINDOWN* – holds down the left win key WINUP* – releases the left win key RWINDOWN* – holds down the right win key RWINUP* – releases the right win key LWINDOWN* – works the same as windown LWINUP* – works the same as winup
WINDOWN * –左の勝利キーを押しますWINUP * –左の勝利キーを押しますRWINDOWN * –右の勝利キーを押しますRWINUP * –右の勝利キーを押しますLWINDOWN * –windown LWINUPと同じ働きをします* –winupと同じ働きをします

WIN* – keypress variable only – press left win key LWIN* – keypress variable only – press left win key RWIN* – keypress variable only – press right win key
WIN * –キー押下変数のみ–左のwinキーを押すLWIN * –キー押下変数のみ–左のwinキーを押すRWIN * –キー押下変数のみ–右のwinキーを押す

DEAD – The, ‘dead’ key that is available on German keyboards (located next to the ‘1’ key) and various French keyboards (located next to the ‘P’ key).
DEAD–ドイツ語キーボード(「1」キーの隣にある)およびさまざまなフランス語キーボード(「P」キーの隣にある)で使用可能な「デッド」キー。

NUM0 - numeric pad 0-9 NUM1
NUM0-テンキー0-9 NUM1

NUM2 NUM3 NUM4 NUM5 NUM6 NUM7 NUM8 NUM9
NUM2 NUM3 NUM4 NUM5 NUM6 NUM7 NUM8 NUM9

NUM* - numeric pad *
NUM *-テンキー*

NUM+ - numeric pad +
NUM +-テンキー+

NUM- - numeric pad -
NUM--テンキー-

NUM. - numeric pad .
NUM -テンキー。

NUM/ - numeric pad / NUMENTER - numeric pad enter NUMINSERT - numeric pad insert NUMHOME - numeric pad home NUMDELETE - numeric pad delete NUMPAGEUP - numeric pad page up NUMPAGEDOWN - numeric pad page down NUMEND - numeric pad end NUMRIGHT - numeric pad right arrow NUMLEFT - numeric pad left arrow
NUM /-テンキー/テンキー-テンキー入力NUMINSERT-テンキーの挿入NUMHOME-テンキーのホームNUMDELETE-テンキーの削除NUMPAGEUP-テンキーのページ上NUMPAGEDOWN-テンキーのページ下NUMEND-テンキーの末尾NUMRIGHT-テンキーの右矢印NUMLEFT -テンキーの左矢印

NUMUP - numeric pad up arrow NUMDOWN - numeric pad down arrow
NUMUP-テンキーの上矢印NUMDOWN-テンキー下矢印

F1-F24 - press F(unction) keys... F1-F24
F1-F24-F(機能)キーを押します... F1-F24

ARROWD - arrow down
矢印-下矢印

ARROWL - arrow left
矢印-左矢印

ARROWR - arrow right
矢印-右矢印

ARROWU – arrow up CAPSLOCK – toggle capslock DEL – press delete key
ARROWU–上矢印CAPSLOCK–capslock DELを切り替えます–削除キーを押します

DELETE – works the same as del
DELETE–delと同じように機能します

END – press end key
END–終了キーを押します

HOME – press home key
HOME–ホームキーを押します

INS – press insert key
INS–挿入キーを押します

INSERT – works the same as ins NUMLOCK – toggle numlock PAGEUP – press page up PAGEDOWN – press page down
INSERT–INSと同じように機能しますNUMLOCK–NUMLOCKを切り替えますPAGEUP–ページアップを押します
PAGEDOWN–ページダウンを押します

~~PAUSE – press the pause/break button (use, 'BREAK' instead) BREAK – press the pause/break button ー時停止–ー時停止/中断ボタンを押します(代わりに「BREAK」を使用します)BREAK–ー時停止/中断ボタンを押します~~

PRINTSCREEN – press printscreen button SCRLOCK – toggle scroll lock SCROLLLOCK – works the same as scrlock
PRINTSCREEN–印刷画面ボタンを押すSCRLOCK–スクロールロックの切り替えSCROLLLOCK–scrlockと同じように動作します

VOLUMEMUTE – media volume mute key VOLUMEDOWN – media volume down key VOLUMEUP – media volume up key NEXTTRACK – media next track key PREVTRACk – media previous track key STOP – media stop key PLAYPAUSE – media play/pause key
VOLUMEMUTE–メディア音量ミュートキーVOLUMEDOWN–メディア音量ダウンキーVOLUMEUP–メディア音量アップキーNEXTTRACK–メディア次のトラックキーPREVTRACk–メディア前のトラックキーSTOP–メディア停止キー
PLAYPAUSE–メディア再生/一時停止キー

0 – 255 – As an additional helper, if you *happen* to know the virtual key value, you can put it between square brackets and it will be used. For instance, the virtual key value for the, 'A' key is 65, 'B' is 66 and 'C' is 67. If you put [SHIFTDOWN][65][66][67][SHIFTUP] in Quick Input, 'ABC' will be typed out (note the uppercase characters). Note that you don't save any time by using this… it's just used behind the scenes in other ways and is exposed in this way for you to use ;)
0から255 –あなたは*仮想キーの値を知っている*起こる場合は、追加のヘルパーとして、あなたは角括弧の間にそれを置くことができ、それが使用されます。たとえば、「A」キーの仮想キー値は65、「B」は66、「C」は67です。[SHIFTDOWN][65] [66] [67] [SHIFTUP]をクイック入力に入力した場合、「ABC」が入力されます(大文字に注意してください)。これを使用して時間を節約しないことに注意してください…それは他の方法で舞台裏で使用され、使用するためにこの方法で公開されています;)

Quick Input inline functions
クイック入力インライン関数

[PAUSE:seconds] – Using this indicator will allow you to insert a pause between characters. Simply use the term, ‘PAUSE’, followed by a colon, then the amount of time in seconds that you would like to pause. For example, A[PAUSE:2.5]B[PAUSE:0.5]C will press, ‘A’, then pause 2.5 seconds, then press, ‘B’, pause one half second, then press, ‘C’.
[PAUSE：seconds]–このインジケーターを使用すると、キャラクター間にポーズを挿入できます。単に「PAUSE」という用語の後にコロンを使用してから、一時停止する時間を秒単位で使用します。たとえば、A [PAUSE：2.5] B [PAUSE：0.5] Cは「A」を押してから2.5秒休止し、「B」を押して0.5秒休止してから「C」を押します。

Key State Token Parameter Values
キー状態トークンのパラメーター値

Key state token parameters are used with the {STATE_KEYSTATE:key} token. The, 'key' parameter can be any key you can type into a token (A, B, C, #, @, etc). For keys that you can't type into a token, use the items from the list below. For instance, if you wanted to see if the F10 key is down, simply use the following token: {STATE_KEYSTATE:F10}.
キー状態トークンパラメータは、{STATE_KEYSTATE：key }トークンで使用されます。'key'パラメーターには、トークンに入力できる任意のキー（A、B、C、＃、@など）を指定できます。トークンに入力できないキーについては、以下のリストのアイテムを使用してください。たとえば、F10キーがダウンしているかどうかを確認するには、次のトークンを使用します：{STATE_KEYSTATE：F10}。

See the section on tokens elsewhere in this document for more info. ENTER
詳細については、このドキュメントのトークンに関するセクションを参照してください。入る

TAB ESC ESCAPE BACK
TAB ESC ESCAPE BACK

BACKSPACE SPACE
バックスペーススペース

DEAD – The, ‘dead’ key that is available on German keyboards (located next to the ‘1’ key) and various French keyboards (located next to the ‘P’ key).
DEAD–ドイツ語キーボード（「1」キーの隣にある）およびさまざまなフランス語キーボード（「P」キーの隣にある）で使用可能な「デッド」キー。

LCTRL – Left control key
LCTRL–左コントロールキー

CTRL – Same as left control key
CTRL–左コントロールキーと同じ

RCTRL – Right control key
RCTRL–右コントロールキー

LALT – Left ALT key
LALT–左Altキー

ALT – Same as left ALT key
ALT–左のALTキーと同じ

RALT – Right ALT key
RALT–右Altキー

LSHIFT – Left shift key
LSHIFT–左シフトキー

SHIFT – Same as left shift key
SHIFT–左シフトキーと同じ

RSHIFT – Right shift key
RSHIFT–右シフトキー

LWIN – Left Windows key
LWIN–左Windowsキー

WIN – Same as left Windows key
WIN–左Windowsキーと同じ

RWIN – Right Windows key
RWIN–右Windowsキー

NUM0 – numeric pad 0–9 NUM1
NUM0–テンキー0–9 NUM1

NUM2 NUM3 NUM4 NUM5 NUM6 NUM7 NUM8 NUM9
NUM2 NUM3 NUM4 NUM5 NUM6 NUM7 NUM8 NUM9

NUM* - numeric pad *
NUM *-テンキー*

NUM+ - numeric pad +
NUM +-テンキー+

NUM- - numeric pad -
NUM--テンキー-

NUM. - numeric pad .
NUM -テンキー。

NUM/ - numeric pad / NUMENTER - numeric pad enter
NUM /-テンキー/ NUMENTER-テンキー入力

F1 - F1-F24 F2
F1-F1-F24 F2

F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15 F16 F17 F18 F19 F20 F21 F22 F23 F24
F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15 F16 F17 F18 F19 F20 F21 F22 F23 F24

ARROWD - arrow down
矢印-下矢印

ARROWL - arrow left
矢印-左矢印

ARROWR - arrow right
矢印-右矢印

ARROWU - arrow up CAPSLOCK - caps lock DEL - delete key
ARROWU-上矢印CAPSLOCK-Caps Lock DEL-削除キー

DELETE - works the same as del END - end key
DELETE-ENDと同じように機能します-終了キー

HOME - home key
HOME-ホームキー

INS - insert key
INS-キーを挿入

INSERT - works the same as ins NUMLOCK - numlock
INSERT-ins NUMLOCK-numlockと同じように機能します

PAGEUP -page up PAGEDOWN - page down
PAGEUP-ページアップPAGEDOWN-ページダウン

PAUSE -pause/break button PRINTSCREEN - printscreen button SCRLOCK - scroll lock
一時停止-一時停止/中断ボタンPRINTSCREEN-印刷画面ボタンSCRLOCK-スクロールロック

SCROLLLOCK - works the same as scrlock
SCROLLLOCK-scrlockと同じように機能します

VoiceAttack Plugins (for the truly mad)
VoiceAttackプラグイン（真に狂った人向け）

A VoiceAttack plugin is code that resides in a dynamic-link library (.dll) that VoiceAttack can call at any point in a command. The plugin can be code that you (or somebody else) can write to enhance the capabilities of VoiceAttack. A plugin will allow you to pass information from VoiceAttack into your code, execute whatever features you want, then allow your code to interact with VoiceAttack or pass information back to VoiceAttack for further processing within your commands.
VoiceAttackプラグインは、VoiceAttackがコマンドの任意の時点で呼び出すことができるダイナミックリンクライブラリ（.dll）にあるコードです。プラグインは、VoiceAttackの機能を強化するためにユーザー（または他の誰か）が作成できるコードにすることができます。プラグインを使用すると、VoiceAttackからコードに情報を渡し、必要な機能を実行してから、コードでVoiceAttackとやり取りしたり、VoiceAttackに情報を渡してコマンド内でさらに処理したりできます。

This is an experimental part of VoiceAttack and is really not intended for everyone (that's why the documentation is all the way down here near the end). This is a means to be able to make VoiceAttack do pretty much whatever we want it to do without affecting its core functionality. The interface for the plugins will be available to anybody that wants to jump in, and I am hoping that what is provided will be easy to understand. Please note that this is an evolving endeavor, and not every aspect of this feature is (or will be) explained here. More aspects will be uncovered/discovered as (or if) more people decide to use this feature. This documentation may or may not be accurate or up to date, depending on the version of VoiceAttack you are using. This document covers version 4 of the plugin interface. To read about previous versions of this interface, you will need to obtain an earlier version of VoiceAttack or post a request in the forum as somebody may have a copy lying around.
これはVoiceAttackの実験的な部分であり、実際にすべての人を対象とするわけではありません（そのため、ドキュメントは最後近くにあります）。これは、VoiceAttackのコア機能に影響を与えずに、VoiceAttackが実行したいことをほとんど何でもできるようにする手段です。プラグインのインターフェースは、飛び込みたい人なら誰でも利用できます。そして、提供されているものが理解しやすいことを望んでいます。これは進化している努力であり、この機能のすべての側面がここで説明されているわけではないことに注意してください。より多くの人々がこの機能を使用することを決定した場合（またはその場合）、より多くの側面が明らかになります。このドキュメントは、使用しているVoiceAttackのバージョンに応じて、正確であったり最新であったりする場合があります。このドキュメントでは、プラグインインターフェイスのバージョン4について説明します。

Setup
セットアップ

If you have the right tools, a VoiceAttack plugin should be fairly easy to construct. The only rub is that there is an interface that you must adhere to (more later). Sample code will be placed in a folder called, 'Plugin Samples' in the VoiceAttack installation directory. Note that there is no library for you to reference. Note: Although having a library with interfaces to reference would speed things up slightly, the decision was made to not require you have to tote around additional files to get your plugin to work. As you will see, there's very little you'll have to do or remember to get information back and forth to VA, or to make VA perform some type of action.
適切なツールがあれば、VoiceAttackプラグインはかなり簡単に構築できます。唯一のこすり方は、あなたが順守しなければならないインターフェースがあるということです（後で）サンプルコードは、VoiceAttackインストールディレクトリの「Plugin Samples」というフォルダーに配置されます。参照するライブラリがないことに注意してください。注：参照するインターフェースを備えたライブラリーを使用すると、処理速度がわずかに向上しますが、プラグインを機能させるために追加のファイルを持ち歩く必要がないという決定が下されました。後でわかるように、情報をVAとやり取りしたり、VAに何らかのアクションを実行させたりするために必要なことはほとんどありません。

Basically, all you will need at this point in time is a version of Visual Studio that supports at least .Net Framework 4.0 (you pick the language). If you do not have this, you can go to Microsoft's site and download an Express/Community edition for free. I won't go into any more detail about that here, since if you've gotten this far without your eyes glazing over, you're being told something you probably already know :)
基本的に、この時点で必要なのは、少なくとも.Net Framework 4.0をサポートするVisual Studioのバージョンだけです（言語を選択します）。これがない場合は、Microsoftのサイトにアクセスして、Express / Communityエディションを無料でダウンロードできます。ここでこれについて詳しくは説明しません。目を凝視せずにここまで進んだ場合、おそらく既に知っていることを伝えられるからです:)

If you are planning on sharing your plugin with others, it is suggested that you compile your
プラグインを他の人と共有することを計画している場合は、コンパイルすることをお勧めします

.dll using the, 'Any CPU' platform target. Also, you may even want to consider making your plugins open-source, due to trust/security concerns. Some people will not run a compiled
「任意のCPU」プラットフォームターゲットを使用した.dll。また、信頼性やセキュリティ上の懸念から、プラグインをオープンソースにすることを検討することもできます。一部の人々はコンパイルされたを実行しません

.dll unless they know that the source is trusted and/or know what the code looks like (I am one of those people).
ソースが信頼されていること、および/またはコードがどのように見えるかを知っている場合を除き、.dll（私はそうした人々の一人です）。

Turning on plugin support in VoiceAttack
VoiceAttackでプラグインサポートを有効にする

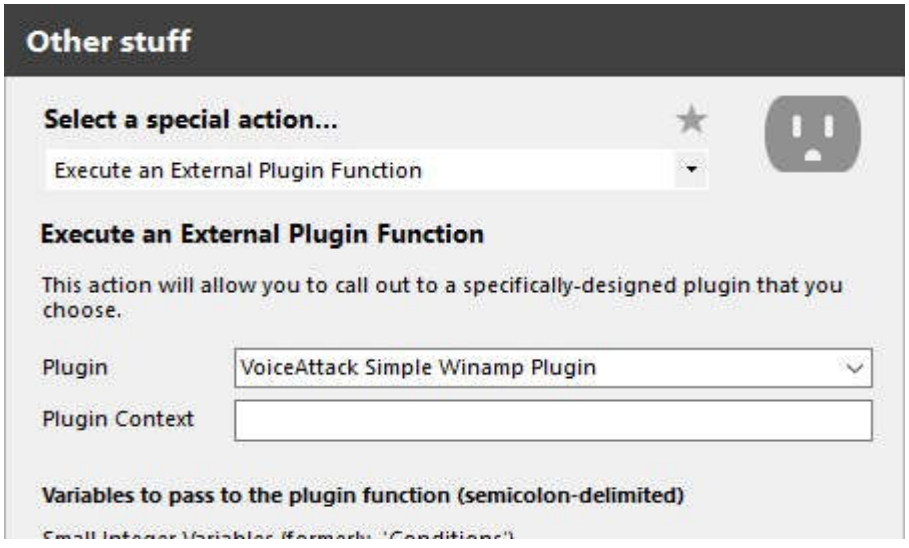
To turn on plugin support in VoiceAttack, you need to go into the options screen and check the, 'Enable Plugin Support' checkbox. A nice warning message will come up and advise you of the dangers of using plugins that somebody else creates. If you accept and continue, you'll get another box that indicates that VoiceAttack will need to be restarted to initialize any plugins. When you run VoiceAttack with plugin support enabled, you will see log messages indicating the state of each plugin that was found and validated.
VoiceAttackでプラグインサポートを有効にするには、オプション画面に移動し、[プラグインサポートを有効にする]チェックボックスをオンにする必要があります。素敵な警告メッセージが表示され、他の人が作成したプラグインを使用することの危険性が通知されます。同意して続行すると、VoiceAttackを再起動してプラグインを初期化する必要があることを示す別のボックスが表示されます。プラグインサポートを有効にしてVoiceAttackを実行すると、検出および検証された各プラグインの状態を示すログメッセージが表示されます。

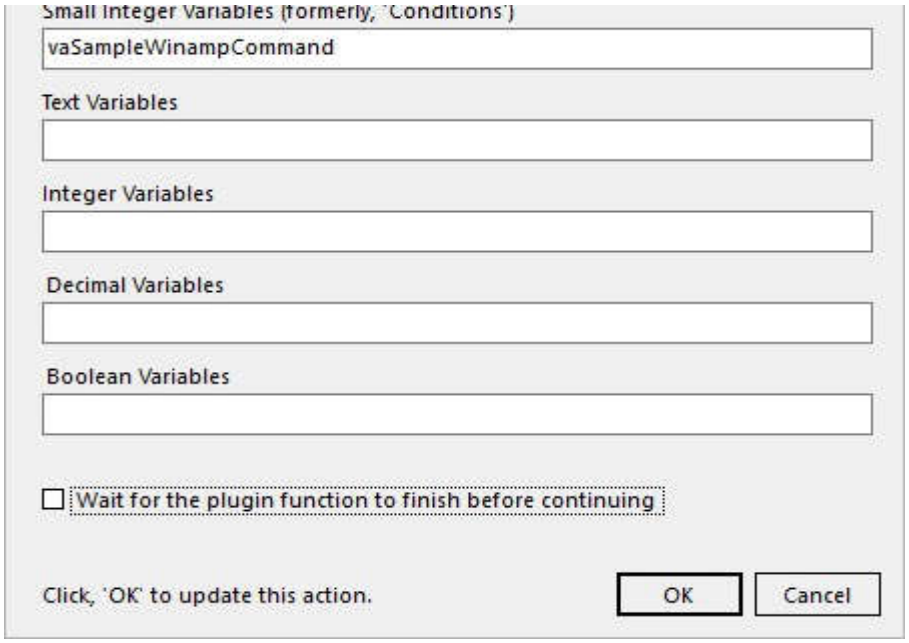
Turning off plugin support
プラグインサポートをオフにする

There are two ways to disable plugin support. You can deselect the box described above, or, you can launch VoiceAttack with the control and shift buttons pressed at the same time. This is a way to catch VoiceAttack before the plugin initializers can be called (in case of danger o_O). You can always just wipe your apps directory if you REALLY want to make sure nothing is going to happen ;)
プラグインのサポートを無効にする方法は2つあります。上記のボックスを選択解除するか、コントロールとシフトボタンを同時に押しながらVoiceAttackを起動できます。これは、プラグイン初期化子を呼び出す前にVoiceAttackをキャッチする方法です（危険なo_Oの場合）。何も起こらないことを本当に確認したい場合は、いつでもアプリのディレクトリを消去することができます;)

Running a plugin from within VoiceAttack
VoiceAttack内からプラグインを実行する

To invoke a VoiceAttack plugin, you will need to put an, 'Execute an External Plugin Function' (long name... might change... lol) command action in your command:
VoiceAttackプラグインを呼び出すには、コマンドに「外部プラグイン関数の実行」（長い名前...変更される可能性があります...笑）コマンドアクションを追加する必要があります。





On this screen, you will pick the plugin that you want to use and any condition or text values you want passed in. You can also indicate a context value and indicate whether or not you want VoiceAttack to wait for the plugin function to return.
この画面では、使用するプラグインと渡す条件またはテキスト値を選択します。また、コンテキスト値を示し、VoiceAttackがプラグイン関数の戻りを待つかどうかを示すこともできます。

NOTE: Any reference below this point regarding the variable input boxes (Small Integer, Text, Integer, Decimal and Boolean) are there for backward compatibility prior to version 4 of the plugin interface. They are not necessary for use after version 4 (VoiceAttack version 1.6+), as you will be able to directly access variables
注：変数入力ボックスに関するこのポイントの下の参照（小さい整数、テキスト、整数、10進およびブール）は、プラグインインターフェイスのバージョン4より前の下位互換性のためにあります。これらは、変数に直接アクセスできるため、バージョン4（VoiceAttackバージョン1.6以降）以降の使用には必要ありません。

from within your plugin.
プラグイン内から。

****
******** When the command action is executed, the plugin context value, any small integer variables (indicated by name in a semicolon-delimited list), any text, integer, decimal, Boolean or date/time variables (indicated the same way), and plugin state are passed to the plugin's invoke function (currently, 'VA_Invoke1').
****
********** ときコマンドアクションが実行され、プラグインコンテキスト値、小さな整数変数（セミコロンで区切られたリスト内の名前で示される）、テキスト、整数、10進数、ブールまたは日付/時間変数（同じ方法で示される）、およびプラグインの状態はプラグインの呼び出し関数に渡されます（現在は「VA_Invoke1」）。

At this point, this is where most of the business will (should) occur. The plugin context that is passed to the plugin can be any string value that you want it to be or any combination of tokens (fairly simple). Small integer (formerly, 'Condition') variables that are passed in can be altered within the plugin or you can add more small integer values (setting a small integer value to null will remove the small integer on return). Same goes for the other variable types (text, integer, decimal, Boolean, date/time). Small integer, text, integer, decimal, Boolean and date/time variables are global and available to all plugins, so play nice. The dictionary that contains state values can be modified however you want it. It's private to your plugin (no other plugin has access to it, and only your plugin can modify it). Since your plugin class is using static functions, this is provided to simply maintain state privately if you need it. I'm sure you can come up with all kinds of ways to persist state between calls, but this might make things a little bit easier.
この時点で、これはほとんどのビジネスが発生する（する）必要がある場所です。プラグインコンテキストプラグインに渡されるのは、必要な任意の文字列値またはトークンの任意の組み合わせです（かなり単純です）。渡される小さな整数（以前の「Condition」）変数は、プラグイン内で変更できます。または、さらに小さな整数値を追加できます（小さな整数値をnullに設定すると、戻り時に小さな整数が削除されます）。他の変数タイプ（テキスト、整数、10進数、ブール、日付/時刻）についても同様です。小さい整数、テキスト、整数、10進数、ブール、および日付/時刻変数はグローバルであり、すべてのプラグインで使用できるため、うまくプレイできます。状態値を含む辞書は、必要に応じて変更できます。プラグインに対してプライベートです（他のプラグインはアクセスできず、プラグインのみが変更できます）。プラグインクラスは静的関数を使用しているため、これは、必要に応じて状態をプライベートに単純に維持するために提供されています。呼び出し間で状態を保持するためのあらゆる種類の方法を考え出すことができると確信していますが、これにより物事が少し簡単になるかもしれません。

After your code goes out of scope and control returns to VoiceAttack, VoiceAttack will update its copy of your plugin state, update any variables you altered and remove any variables set to null.
コードがスコープ外になり、制御がVoiceAttackに戻ると、VoiceAttackはプラグインの状態のコピーを更新し、変更した変数を更新し、nullに設定された変数を削除します。

If you did not check the, 'wait until function returns' checkbox, the call to the plugin will be tossed into another thread and VoiceAttack will continue processing actions immediately.
「関数が戻るまで待機する」チェックボックスをチェックしなかった場合、プラグインへの呼び出しは別のスレッドに投げ込まれ、VoiceAttackはすぐにアクションの処理を続行します。

If you do check the, 'wait until function returns' checkbox, VoiceAttack will do just that... wait until your code finishes before continuing. At this point you have a chance to work with any of the values you had modified/added in subsequent actions within the command. For instance, maybe your plugin goes out to the internet to retrieve information. Your plugin can set that information to a value that is passed back to VoiceAttack and then VoiceAttack can use that value (for TTS or for conditional flow).
あなたがチェックした場合は、「関数が戻るまで待つチェックボックス」、VoiceAttackはまさにそれを行います...あなたのコードを続行する前に終了するまで待ちます。この時点で、コマンド内の後続のアクションで変更/追加した値を操作する機会があります。たとえば、プラグインが情報を取得するためにインターネットにアクセスする場合があります。プラグインはその情報をVoiceAttackに返される値に設定でき、VoiceAttackはその値を使用できます（TTSまたは条件付きフロー用）。

VoiceAttack Plugin Requirements and Interface (this will be reorganized properly at some point)
VoiceAttack Pluginの要件とインターフェイス（これは、ある時点で適切に再編成されます）

The code that you compile must be a .net framework 4.0+ .dll. The .dll must be in a specific location.
コンパイルするコードは、.net framework 4.0+ .dllである必要があります。.dllは特定の場所になければなりません。

The VoiceAttack Apps directory is located (by default) in the VoiceAttack installation directory (this location can be changed in the Options page). I called it, 'Apps' because it's not just for plugins. It can be for your out-of-process (.exe) stuff as well... just thought you should know that for some reason o_O.
VoiceAttack Appsディレクトリは、デフォルトでVoiceAttackインストールディレクトリにあります（この場所は[オプション]ページで変更できます）。プラグイン専用ではないため、「アプリ」と呼びました。アウトプロセス（.exe）にも使用できます...何らかの理由でo_Oであることを知っておくべきだと思っただけです。

VoiceAttack will crawl all subdirectories of the Apps directory. VoiceAttack will ONLY
VoiceAttackは、Apps ディレクトリのすべてのサブディレクトリをクロールします。VoiceAttackのみ

crawl subdirectories of the Apps directory, and not the Apps directory itself (that means that any .dll files that are just sitting in the Apps directory will be skipped... this is for housekeeping reasons... your plugins need to be in their own directories). Again... The compiled plugin .dll must reside IN A SUBDIRECTORY of the VoiceAttack Apps directory:
Appsディレクトリそのものではなく、Appsディレクトリのサブディレクトリをクロールします（つまり、Appsディレクトリにある.dllファイルはスキップされます...これはハウスキーピングの理由のためです...プラグインは独自のディレクトリ）。繰り返しますが、コンパイルされたプラグイン.dllはVoiceAttack Appsディレクトリのサブディレクトリに存在する必要があります。

“C:¥Program Files (x86)¥VoiceAttack¥Apps¥myPlugin.dll” will be skipped.
「C：¥ Program Files（x86）¥ VoiceAttack ¥ Apps ¥ myPlugin.dll」はスキップされます。

“C:¥Program Files (x86)¥VoiceAttack¥Apps¥MyPluginDirectory¥myPlugin.dll” will be picked up.
「C：¥ Program Files（x86）¥ VoiceAttack ¥ Apps ¥ MyPluginDirectory ¥ myPlugin.dll」が選択されます。

When VoiceAttack discovers a .dll, it interrogates it and sees if it is compatible with the current version of VoiceAttack. If it is not, you will see a log entry indicating this.
VoiceAttackが.dllを検出すると、それを照会し、VoiceAttackの現在のバージョンと互換性があるかどうかを確認します。そうでない場合は、これを示すログエントリが表示されます。

Functions your plugin must provide...
プラグインが提供しなければならない機能...

VoiceAttack will not be instantiating any objects in your plugins, so all the required functions must be static. If you have objects that need to be instantiated, feel free to do that from within the static functions. You can call your classes anything you want.
VoiceAttackはプラグインのオブジェクトをインスタンス化しないため、必要な機能はすべてstaticでなければなりません。インスタンス化する必要があるオブジェクトがある場合は、静的関数内から自由にインスタンス化できます。クラスは好きな名前で呼び出すことができます。

VoiceAttack will find all the classes in your .dll that meet the criteria and reference each class as a separate plugin (this way if you have multiple plugins per assembly (.dll)).
VoiceAttackは、基準を満たす.dll内のすべてのクラスを検索し、各クラスを個別のプラグインとして参照します（アセンブリごとに複数のプラグイン（.dll）がある場合はこの方法）。

As stated above, your class must contain static functions. For version 4 (the current version), seven of them are currently necessary. The functions are for display name, plugin id, extended display info, initialization, invoke and application exit, and command stop. Each of the functions must have a specific set of parameters to work.
上記のように、クラスには静的関数が含まれている必要があります。バージョン4（現在のバージョン）の場合、そのうちの7つが現在必要です。これらの機能は、表示名、プラグインID、拡張表示情報、初期化、起動とアプリケーションの終了、およびコマンドの停止用です。各機能には、機能する特定のパラメーターセットが必要です。

The first required plugin function is called VA_Id. This returns a Guid to VoiceAttack and uniquely identifies your plugin so that it can actually be called when needed.
最初に必要なプラグイン関数はVA_Idと呼ばれます。これにより、GuidがVoiceAttackに返され、プラグインが一意に識別されるため、必要なときに実際に呼び出すことができます。

This is a Guid that must be generated by you. Please find a proper tool (Visual Studio has one built in) that will generate the value for you. Note that if your project has multiple classes that present themselves as plugins, the Guid must be different for each class.
これは、あなたが生成しなければならない GUID です。値を生成する適切なツール（Visual Studioに組み込まれているもの）を見つけてください。プロジェクトにプラグインとして表示される複数のクラスがある場合、Guidはクラスごとに異なる必要があることに注意してください。

The next plugin function is called, VA_DisplayName. This returns a string value and is what VoiceAttack will display when referring to your plugin. You will see this value in the selection lists and log entries.
次のプラグイン関数VA_DisplayNameが呼び出されます。これは文字列値を返し、プラグインを参照するときにVoiceAttackが表示するものです。この値は、選択リストとログエントリに表示されます。

The third plugin function is called, VA_DisplayInfo. This is another simple string value that can be used to display extra information about your plugin (such as the author or helpful info regarding the plugin). Make sure the string that is returned contains proper formatting. This function can return an empty string.
3番目のプラグイン関数VA_DisplayInfoが呼び出されます。これは、プラグインに関する追加情報（作成者やプラグインに関する役立つ情報など）を表示するために使用できる別の単純な文字列値です。返される文字列に適切な書式が含まれていることを確認してください。この関数は空の文字列を返すことができます。

The fourth function is VA_Init1 (UPDATED IN VERSION 4). This function gets called when VoiceAttack starts, before the main screen form is loaded. This function does not return a value (void), however, it has a single dynamic parameter called, 'vaProxy'. This parameter takes the place of the eight parameters that were used before version 4. Note that this function is called asynchronously and there is no guarantee of the order that plugins will be initialized. Note that through the vaProxy variable, you can access the state values, any VoiceAttack variables as well as be able to execute commands or change profiles (and lots more… more about vaProxy later).
4番目の関数は VA_Init1(バージョン4で更新)です。この関数は、メイン画面フォームがロードされる前に、VoiceAttackの起動時に呼び出されます。この関数は値(void)を返しませんが、「vaProxy」と呼ばれる単一の動的パラメーターがあります。このパラメーターは、バージョン4より前に使用されていた8つのパラメーターの代わりになります。この関数は非同期に呼び出され、プラグインが初期化される順序は保証されません。vaProxy変数を使用して、状態値、VoiceAttack変数にアクセスしたり、コマンドを実行したり、プロファイルを変更したりすることができることに注意してください(vaProxyについては後で詳しく説明します)。

The fifth function is VA_Exit1 (UPDATED IN VERSION 4). This function gets called when VoiceAttack closes. This will give you an opportunity to clean up anything that you want to upon application exit. This function does not return a value (void), however, it has a single dynamic parameter called, 'vaProxy'. This parameter takes the place of the state dictionary that was used before version 4. You can use the vaProxy variable to access the state information that is passed back and forth to VoiceAttack (again, more about vaProxy later). This function is called asynchronously and there is no guarantee of the order that plugins will be called on exit.
5番目の関数は VA_Exit1(バージョン4で更新)です。この関数は、VoiceAttackが閉じると呼び出されます。これにより、アプリケーションの終了時に必要なものをすべてクリーンアップできます。この関数は値(void)を返しませんが、「vaProxy」と呼ばれる単一の動的パラメーターがあります。このパラメーターは、バージョン4より前に使用された状態ディクショナリーの代わりに使用します。vaProxy変数を使用して、VoiceAttackとやり取りされる状態情報にアクセスできます(再度、vaProxyについて詳しく説明します)。この関数は非同期に呼び出され、終了時にプラグインが呼び出される順序の保証はありません。

Note: The, 'vaProxy' that is available in VA_Exit1 is limited to being only able to get state information. Any other use of vaProxy in VA_Exit1 may result in an exception being raised and VoiceAttack becoming hung on exit (possibly requiring you to terminate VoiceAttack via the Task Manager).
注:VA_Exit1で使用可能な「vaProxy」は、状態情報のみを取得できることに制限されています。VA_Exit1でvaProxyを他に使用すると、例外が発生し、VoiceAttackが終了時にハングする可能性があります(タスクマネージャーを使用してVoiceAttackを終了する必要がある場合があります)。

The sixth required static function is VA_StopCommand (NEW IN VERSION 4). This function takes no parameters and is called any time the user executes a, 'stop all commands' action or clicks on the, 'stop all commands' button on the main screen. This function's purpose is to provide a means for the plugin to know that this event as occurred. Again, this will be called asynchronously and there is no guarantee that of the order that plugins will be called.
6番目に必要な静的関数は VA_StopCommand(バージョン4の新機能)です。この関数はパラメーターを使用せず、ユーザーがメイン画面の「すべてのコマンドを停止」アクションを実行するか、「すべてのコマンドを停止」ボタンをクリックしたときに呼び出されます。この関数の目的は、このイベントが発生したことをプラグインが知る手段を提供することです。繰り返しますが、これは非同期で呼び出され、プラグインが呼び出される順序の保証はありません。

The seventh required function is VA_Invoke1 (UPDATED IN VERSION 4). This function gets called via a special command action in VoiceAttack ('Execute external plugin function'). The VA_Invoke1 function does not have a return value (void). This function has a single dynamic parameter called, 'vaProxy' which replaces all of the parameters for this function that existed prior to version 4.
7番目に必要な関数は VA_Invoke1(バージョン4で更新)です。この関数は、VoiceAttackの特別なコマンドアクション(「外部プラグイン関数の実行」)を介して呼び出されます。VA_Invoke1関数には戻り値(void)はありません。この関数には「vaProxy」という単一の動的パラメーターがあり、バージョン4より前に存在していたこの関数のすべてのパラメーターを置き換えます。

Plugin parameter notes
プラグインパラメーターのメモ

Note that this section is now in two parts: version 4 and above and version 3 and below.
このセクションは、バージョン4以降とバージョン3以下の2つのパートに分かれていることに注意してください。

Version 4 and above plugin parameter notes vaProxy
バージョン4以降のプラグインパラメーターノートvaProxy

Version 4 of the plugin interface brings an attempt to simplify things somewhat by eliminating an ever-growing list of parameters for each function. In version 3 and versions prior, there was a parameter for the context, the state and each and every data type to be passed in (see v3 notes below on how to access vaProxy in v3 and prior). In this version, we have the single dynamic parameter called, 'vaProxy'. It's not as easy to see what is going on with this approach, but it will make things far more flexible going forward. That is, the vaProxy's capabilities can expand indefinitely without having to rework this interface every time there is a change. Below is an outline of the properties and methods available to vaProxy with a description of how to use each one. Also, if the property or method relates to functionality available in previous versions of the interface you will find notes on that as well.
プラグインインターフェイスのバージョン4は、各関数のパラメーターのリストが増え続けることを排除することで、物事をいくらか簡素化する試みをもたらします。バージョン3以前のバージョンでは、コンテキスト、状態、渡される各データタイプのパラメーターがありました（v3以前のvaProxyにアクセスする方法については、以下のv3ノートを参照してください）。このバージョンでは、「vaProxy」と呼ばれる単一の動的パラメーターがあります。このアプローチで何が起こっているかを見るのは簡単ではありませんが、今後ははるかに柔軟になります。つまり、vaProxyの機能は、変更があるたびにこのインターフェイスを作り直すことなく、無期限に拡張できます。以下は、vaProxyで使用できるプロパティとメソッドの概要と、それぞれの使用方法の説明です。また、

To use the vaProxy variable, simply access its attributes just as you would any other class. It will be a little awkward at first, as the type of vaProxy is, 'dynamic'. That means that the attributes are not available until runtime (also known as, 'late bound') and you will not have the assistance of Intellisense. You will generally not receive compiler errors when using dynamic variables. If something is wrong, you will receive a runtime error when that line is hit. Again, if you are down here reading this, I am most likely telling you stuff you already know. Rock on ;)
vaProxy変数を使用するには、他のクラスと同じように単純にその属性にアクセスします。vaProxyのタイプは「動的」なので、最初は少し厄介です。つまり、属性は実行時（「後期バインド」とも呼ばれます）まで利用できず、Intellisenseの支援を受けられません。通常、ダイナミック変数を使用する場合、コンパイラエラーは発生しません。何かが間違っている場合、その行にヒットするとランタイムエラーが表示されます。繰り返しますが、もしあなたがこれを読んでいるなら、私はおそらくあなたがすでに知っていることをあなたに話しているでしょう。ロックオン;）

vaProxy Attributes
vaProxy属性

VoiceAttack's proxy object, 'vaProxy', has a bunch of methods and properties. Some of these methods and properties are accessed directly from the proxy object itself: vaProxy.Context, vaProxy.GetText(string VariableName). The proxy object also has a set of objects to help organize its properties and methods into logical groups. These objects are currently Command, Profile, Queue, Utility and State. Each of these will be outlined below.
VoiceAttackのプロキシオブジェクト「vaProxy」には、多数のメソッドとプロパティがあります。これらのメソッドおよびプロパティの一部は、プロキシオブジェクト自体から直接アクセスされます：vaProxy.Context、vaProxy.GetText（string VariableName）。プロキシオブジェクトには、プロパティとメソッドを論理グループに整理するのに役立つ一連のオブジェクトもあります。これらのオブジェクトは現在、コマンド、プロファイル、キュー、ユーティリティ、および状態です。これらのそれぞれの概要を以下に示します。

vaProxy Base Attributes
vaProxy基本属性

These are the attributes that belong directly to the vaProxy object. They represent the basic functionality of the proxy object that you've been using for like forever. You call them straight from the vaProxy object:
これらは、vaProxyオブジェクトに直接属する属性です。これらは、永遠に使用してきたプロキシオブジェクトの基本機能を表します。vaProxyオブジェクトから直接呼び出します：

Context – This is a read-only string property which is set via the, 'Plugin Context' input box on the, 'Execute External Plugin Function' screen. This property is only available when you
コンテキスト-これは、「外部プラグイン関数の実行」画面の「プラグインコンテキスト」入力ボックスを介して設定される読み取り専用の文字列プロパティです。このプロパティは、

are using VA_Invoke1 (it is not accessible in any other function).
VA_Invoke1を使用しています（他の関数ではアクセスできません）。

Since there is only one function (VA_Invoke1) that is accessed by commands, you need a way to differentiate between different types of requests. This is just a simple way to get information to your plugin without having to assign a variable. You will probably use this property the most.
コマンドによってアクセスされる関数（VA_Invoke1）が1つしかないため、異なるタイプの要求を区別する方法が必要です。これは、変数を割り当てることなくプラグインに情報を取得する簡単な方法です。このプロパティをおそらく最も使用するでしょう。

```
public static void VA_Invoke1(dynamic vaProxy)
public static void VA_Invoke1（dynamic vaProxy）

{
{

if (vaProxy.Context == "lights on")
if （vaProxy.Context =="点灯"）
```

```
//do some stuff here
//ここで何かをする


else if (vaProxy.Context == "lights off")
else if (vaProxy.Context =="消灯")


//do something different
//何か違うことをする


}
}
```

This property will be available in VA_Invoke1 only and will be null if accessed elsewhere (including inline functions (see, 'Execute an Inline Function' section earlier in this document)).
このプロパティはVA_Invoke1でのみ使用でき、他の場所(インライン関数を含む(このドキュメントの「インライン関数の実行」を参照)でアクセスした場合はnullになります)。

The Context property takes the place of the Context parameter that was available in VA_Invoke1 for version 3 and before.
Contextプロパティは、バージョン3以前のVA_Invoke1で使用可能であったContextパラメーターの代わりになります。

SessionState – this is a read/write Dictionary of (String, Object). This property is accessible through VA_Init1, VA_Invoke1 and VA_Exit1. This property is for your own private use within the plugin you create. No other plugin has access to the values. It serves as a kind of session, so that you can easily maintain information between calls without having to do any kind of persistence. The dictionary is (String, Object) so you can name your values whatever you want, as well as store any kind of value type. Whatever you do to this dictionary will be reflected in VoiceAttack upon return (VA_Invoke1 and VA_Init1 only). So, if you empty this dictionary, VoiceAttack's copy of this dictionary will be emptied. Null values will not be cleaned out. Note: When the state dictionary is initialized for a plugin, there are three key/value pairs that are included for use. The keys are below (the key names are the same as the token values used elsewhere):
SessionState –これは、(String、Object)の読み取り/書き込み辞書です。このプロパティには、VA_Init1、VA_Invoke1、およびVA_Exit1からアクセスできます。このプロパティは、作成するプラグイン内で個人的に使用するためのものです。他のプラグインは値にアクセスできません。これは一種のセッションとして機能するため、コール間で情報を維持するために、いかなる種類の永続性も必要ありません。ディクショナリは(String、Object)であるため、任意の値に名前を付けたり、任意の種類の値タイプを保存したりできます。このディクショナリに対して行うことは、返されるとVoiceAttackに反映されます(VA_Invoke1およびVA_Init1のみ)。したがって、この辞書を空にすると、この辞書のVoiceAttackのコピーは空になります。NULL値は意志ではありません掃除してください。注:状態辞書がプラグイン用に初期化されると、使用するために含まれるキー/値のペアが3つあります。キーは以下のとおりです(キー名は他で使用されているトークン値と同じです):

VA_DIR : The installation directory of VoiceAttack. VA_APPS : VoiceAttack apps/plugins directory.
VA_DIR:VoiceAttackのインストールディレクトリ。VA_APPS:VoiceAttack apps / pluginsディレクトリ。

VA_SOUNDS : VoiceAttack sounds directory.
VA_SOUNDS:VoiceAttackはディレクトリを鳴らします。

Again, these values can be erased and manipulated however you want.
繰り返しますが、これらの値は必要に応じて消去および操作できます。

```
public static void VA_Init1(dynamic vaProxy)
public static void VA_Init1 (dynamic vaProxy)

{
{

vaProxy.SessionState.Add("new state value", 369); vaProxy.SessionState.Add("second new state value",
"hello");
vaProxy.SessionState.Add ("新しい状態値" 、369); vaProxy.SessionState.Add ("second new state value" 、
"hello" );

String sValue = vaProxy.SessionState["new state value"];
文字列sValue = vaProxy.SessionState [" new state value "];

}
}
```

The SessionState property takes the place of the State parameter that was available for each function in version 3 and before.
SessionStateプロパティは、バージョン3以前の各関数で使用可能だったStateパラメーターの代わりになります。

NOTE: This property should only be used with plugins, as the values are not maintained
注：値は維持されないため、このプロパティはプラグインでのみ使用する必要があります

between calls using inline functions.
インライン関数を使用した呼び出し間。

SetSmallInt(string VariableName, short? Value) – this method allows you to set a VoiceAttack short integer variable indicated by VariableName to a value specified by Value. Short integers were referred to as, 'Conditions' in earlier versions of VoiceAttack. Note that the Value can be null which will clear out the named variable.
SetSmallInt (string VariableName、short？Value) –このメソッドにより、VariableNameで示されるVoiceAttack短整数変数をValueで指定された値に設定できます。VoiceAttackの以前のバージョンでは、短い整数は「条件」と呼ばれていました。Valueがnullになり、名前付き変数がクリアされることに注意してください。

```
public static void VA_Invoke1(dynamic vaProxy)
public static void VA_Invoke1(dynamic vaProxy)

{
{

vaProxy.SetSmallInt("mySmallInt", 55);
vaProxy.SetSmallInt(" mySmallInt"、55);

}
}
```

Small integers are public and can be accessed from any plugin or any command within the VoiceAttack user interface. That means that anybody can view or modify these values.
小さな整数はパブリックであり、VoiceAttackユーザーインターフェイス内の任意のプラグインまたはコマンドからアクセスできます。つまり、誰でもこれらの値を表示または変更できるということです。

You can access the values in conditions using the {SMALL:variableName} ({COND:conditionName} remains for backward compatibility) token in various places in VoiceAttack.
VoiceAttackのさまざまな場所で{SMALL：variableName }({COND：conditionName }は下位互換性のために残ります)トークンを使用して、条件の値にアクセスできます。

SetSmallInt takes partial place of the SmallIntegerValues parameter (dictionary of (String, short)) in version 3 and before.
SetSmallIntは、バージョン3以前のSmallIntegerValuesパラメーター((String、short)のディクショナリー)の部分的な場所を取ります。

GetSmallInt(string VariableName) - this function will return a short integer value if the variable indicated by VariableName exists, or null if the variable does not exist. Note that you will always want to check to see if the value is null:
GetSmallInt(string VariableName)-この関数は、VariableNameで示される変数が存在する場合は短整数値を返し、変数 が存在しない場合はnullを返します。値がnullかどうかを常に確認する必要があることに注意してください。

```
public static void VA_Invoke1(dynamic vaProxy)
public static void VA_Invoke1(dynamic vaProxy)

{
{

short? myShort = vaProxy.GetSmallInt("mySmallInt");
ショート？myShort = vaProxy.GetSmallInt(" mySmallInt");
```

```
if (myShort.HasValue)
if(myShort.HasValue)


{
{


//do some stuff
//何かをする


}
}


}
}
```

GetSmallInt takes partial place of the SmallIntegerValues parameter (dictionary of (String, short)) in version 3 and before.
GetSmallIntは、バージョン3以前のSmallIntegerValuesパラメーター（(String、short)のディクショナリー）の部分的な場所を取ります。

The functionality you see in GetSmallInt and SetSmallInt is repeated for the following functions with their corresponding data types. Note that each replaces their corresponding dictionary parameter in version 3 and before:
GetSmallIntおよびSetSmallIntに表示される機能は、対応するデータ型を持つ次の関数に対して繰り返されます。バージョン3以前では、それぞれが対応する辞書パラメーターを置き換えることに注意してください。

SetInt(string VariableName, int? Value) – Set a nullable integer value
SetInt(string VariableName、int？Value)– NULL入力可能な整数値を設定します

GetInt(string VariableName) – Returns a nullable integer value
GetInt(string VariableName)– NULL入力可能な整数値を返します

SetText(string VariableName, string Value) – Set a string value
SetText(string VariableName、string Value)–文字列値を設定します

GetText(string VariableName) – Returns a string value
GetText(string VariableName)–文字列値を返します

SetDecimal(string VariableName, decimal? Value) – Set a nullable decimal value
SetDecimal（string VariableName、decimal？Value）– NULL入力可能な10進値を設定します

GetDecimal(string VariableName) – Returns a nullable decimal value
GetDecimal（string VariableName）– null許容の10進数値を返します

SetBoolean(string VariableName, Boolean? Value) – Set a nullable Boolean value
SetBoolean（string VariableName、Boolean？Value）– null入力可能なブール値を設定します

GetBoolean(string VariableName) – Returns a nullable Boolean value
GetBoolean（string VariableName）– null入力可能なブール値を返します

SetDate(string VariableName, DateTime? Value) – Set a nullable DateTime value
SetDate（ string VariableName、DateTime？Value）– null許容のDateTime値を設定します

GetDate(string VariableName) – Returns a nullable DateTime value
GetDate（string VariableName）– null許容のDateTime値を返します

ProfileNames() – Returns a string array of all profile names.
ProfileNames ()–すべてのプロファイル名の文字配列を返します。

WriteToLog(String Value, String Color) – This is a simple way to get some information to the VoiceAttack log. This could be for your own debugging reasons or it could be information for the user. The text indicated in Value will be written, with a status icon of the color you choose. The choices are: “red”, “blue”, “green”, “yellow”, “orange”, “purple”, “blank”, “black”, “gray”, “pink”.
WriteToLog（文字列値、文字列色）–これは、VoiceAttackログに情報を取得する簡単な方法です。これは、独自のデバッグ上の理由か、ユーザーの情報である可能性があります。[値]に示されているテキストが、選択した色のステータスアイコンとともに書き込まれます。選択肢は、「赤」、「青」、「緑」、「黄」、「オレンジ」、「紫」、「空白」、「黒」、「灰色」、「ピンク」です。

vaProxy.WriteToLog(“What is love?”, ″red″);
vaProxy.WriteToLog(“ What is love？”、″red″);

ClearLog() – This simply clears the VoiceAttack log on the main screen.
ClearLog（）–これは、メイン画面のVoiceAttackログをクリアするだけです。

ProxyVersion – This will return a System.Version object to indicate the version of the proxy interface. This will help you determine whether or not the installed interface is compatible with your plugin.
ProxyVersion –これは、プロキシインターフェイスのバージョンを示すSystem.Versionオブジェクトを返します。これは、インストールされているインターフェイスがプラグインと互換性があるかどうかを判断するのに役立ちます。

System.Version v = vaProxy.ProxyVersion;
System.Version v = vaProxy.ProxyVersion;

VAVersion – This will return a System.Version object to indicate the version of VoiceAttack currently in use. This will help you determine whether or not the user can use your plugin.
VAVersion –これは、現在使用中のVoiceAttackのバージョンを示すSystem.Versionオブジェクトを返します。これは、ユーザーがプラグインを使用できるかどうかを判断するのに役立ちます。

System.Version v = vaProxy.VAVersion;
System.Version v = vaProxy.VAVersion;

PluginPath() – This will return a string that contains the full path of the executing plugin. This function is not applicable when using the proxy object within inline functions (see, 'Execute an Inline Function' section earlier in this document). Note: This is a function and not a property for some reason o_O.
PluginPath ()–実行中のプラグインのフルパスを含む文字列を返します。この関数は、インライン関数内でプロキシオブジェクトを使用する場合には適用されません(このドキュメントの「インライン関数を実行する」セクションを参照)。注:これは関数であり、何らかの理由でプロパティo_Oではありません。

String s = vaProxy.PluginPath();
文字列s = vaProxy.PluginPath();

Stopped – This will return true if the user has clicked the, 'stop all commands' button on the main screen or a, 'Stop all commands' action has been issued. This property was created mainly for use within inline functions (see, 'Execute an Inline Function' section earlier in this document), but works just as well within plugins. Note – the use of this property this can be replaced with the, 'CommandsStopped' event (later in this section).
停止–ユーザーがメイン画面の「すべてのコマンドを停止」ボタンをクリックした場合、または「すべてのコマンドを停止」アクションが発行された場合、trueを返します。このプロパティは主にインライン関数内で使用するために作成されました(このドキュメントの「インライン関数の実行」セクションを参照)が、プラグイン内でも同様に機能します。注–このプロパティの使用は、「CommandsStopped」イベント(このセクションで後述)に置き換えることができます。

Boolean b = vaProxy.Stopped;
ブールb = vaProxy.Stopped;

InstallDir – This is the VoiceAttack installation directory as a string.
InstallDir– これは、文字列としてのVoiceAttackインストールディレクトリです。

String s = vaProxy.InstallDir;
文字列s = vaProxy.InstallDir;

SoundsDir – By default, this is the folder named, 'Sounds' under the VoiceAttack root directory. This is a place where you can store your VoiceAttack sound packs. If you do not
SoundsDir– デフォルトでは、これはVoiceAttackルートディレクトリの下の「Sounds」という名前のフォルダーです。これは、VoiceAttackサウンドパックを保存できる場所です。もししないなら

want to use the 'Sounds' folder in the VoiceAttack installation directory, you can change this to be whatever folder you want it to be on the VoiceAttack Options page. Note that no check is made to see if this folder exists. This property returns a string.
VoiceAttackインストールディレクトリの「Sounds」フォルダを使用する場合は、VoiceAttackオプションページで目的のフォルダに変更できます。このフォルダが存在するかどうかを確認するチェックは行われないことに注意してください。このプロパティは文字列を返します。

AppsDir – This is the folder named, 'Apps' under the VoiceAttack root directory. This is the place where you can store apps that you use with VoiceAttack (.exe files) and VoiceAttack plugins (.dll files). Just like the, 'Sounds' folder, you can change the location of the VoiceAttack Apps folder in the VoiceAttack Options page. Note that no check is made to see if this folder exists. This property returns a string.
AppsDir– これは、VoiceAttackルートディレクトリの下にある「Apps」という名前のフォルダーです。これは、VoiceAttack（.exeファイル）およびVoiceAttackプラグイン（.dllファイル）で使用するアプリを保存できる場所です。「サウンド」フォルダーと同様に、VoiceAttackオプションページでVoiceAttackアプリフォルダーの場所を変更できます。このフォルダが存在するかどうかを確認するチェックは行われないことに注意してください。このプロパティは文字列を返します。

AssembliesDir – This is the VoiceAttack Shared¥Assemblies folder that is (should be) located in the VoiceAttack installation folder. Note that no check is made to see if the folder exists. This property returns a string.
AssembliesDir– これは、VoiceAttackインストールフォルダーにある（必要な）VoiceAttack Shared ¥ Assembliesフォルダーです。フォルダーが存在するかどうかを確認するチェックは行われないことに注意してください。このプロパティは文字列を返します。

ResetStopFlag() – This function will reset the flag used to check if the, 'Stop all commands' button has been pressed, or a, 'Stop all commands' action has been issued. This function was created mainly for use within inline functions (see, 'Inline Functions' section later in this document), but works just as well within plugins. Note – the use of this function this can be replaced with the, 'CommandsStopped' event (later in this section).
ResetStopFlag()–この関数は、「すべてのコマンドを停止」ボタンが押されたか、「すべてのコマンドを停止」アクションが発行されたかを確認するために使用されるフラグをリセットします。この関数は主にインライン関数内で使用するために作成されました（このドキュメントの「インライン関数」セクションを参照）が、プラグイン内でも同様に機能します。注–この関数の使用は、「CommandsStopped」イベント（このセクションで後述）に置き換えることができます。

if (vaProxy.Stopped)
if（vaProxy.Stopped）

{
{

vaProxy.WriteToLog(“VoiceAttack commands have stopped!”, “Orange”);
vaProxy.WriteToLog(“ VoiceAttackコマンドが停止しました！”、“オレンジ”);

vaProxy.ResetStopFlag(); //reset the flag here
vaProxy.ResetStopFlag(); //ここでフラグをリセットします

}
}

IsRelease – This Boolean property will be true if the current version of VoiceAttack is a full release and false if it is not.
IsRelease– このブール型プロパティは、VoiceAttackの現在のバージョンが完全なリリースである場合はtrue、そうでない場合はfalseです。

IsTrial – This Boolean property will be true if the current version of VoiceAttack is the trial version and false if it is not.
IsTrial – VoiceAttack の現在のバージョンが試用版である場合、このブール型プロパティはtrue、そうでない場合はfalseです。

PluginsEnabled – This Boolean property will be true if plugins are enabled on the Options screen, and false if plugins are not enabled (the intended use for this is within inline functions).
PluginsEnabled– このBooleanプロパティは、オプション画面でプラグインが有効になっている場合はtrueになり、プラグインが有効になっていない場合はfalseになります（これの目的はインライン関数内です）。

NestedTokensEnabled – This Boolean property will be true if the, ‘Use nested tokens’ option is enabled on the Options screen and false if not. This is useful when using the Utility.ParseTokens() function.
NestedTokensEnabled– [オプション]画面で[ネストされたトークンを使用する]オプションが有効になっている場合、このブールプロパティはtrueになり、そうでない場合はfalseになります。これは、Utility.ParseTokens()関数を使用するときに役立ちます。

AutoProfileSwitchingEnabled – This Boolean property will be true if the, ‘Enable Automatic Profile Switching’ option is enabled on the Options screen and false if not.
AutoProfileSwitchingEnabled– [オプション]画面で[自動プロファイル切り替えを有効にする]オプションが有効になっている場合、このブールプロパティはtrueになり、そうでない場合はfalseになります。

MainWindowHandle – This returns VoiceAttack’s main window handle as an IntPtr.
MainWindowHandleは–これはのIntPtrとしてVoiceAttackのメインウィンドウハンドルを返します。

IntPtr hWnd = vaProxy.MainWindowHandle;
IntPtr hWnd = vaProxy.MainWindowHandle;

Close() – This function will close VoiceAttack. It is the same as if you had clicked on the main screen's close button (top-right).
Close()–この関数はVoiceAttackを閉じます。メイン画面の閉じるボタン（右上）をクリックした場合と同じです。

vaProxy.Utility Attributes
vaProxy.Utilityの属性

Below are the attributes that belong to the Utility object of vaProxy. It is a collection of functions that are available in VoiceAttack and are exposed so that you may get use out of them. For example, to access the ParseTokens method of the Utility object, you just call it this way: vaProxy.Utility.ParseTokens("some value").
以下は、vaProxyのUtilityオブジェクトに属する属性です。これは、VoiceAttackで使用可能な機能のコレクションであり、それらを使用できるように公開されています。たとえば、UtilityオブジェクトのParseTokensメソッドにアクセスするには、vaProxy.Utility.ParseTokens(" some value")のように呼び出すだけです。

ExtractPhrases(string Phrases, bool TrimSpaces (optional), bool Lowercase (optiona)) – This utility function returns a string array containing the extracted phrases indicated by Phrases. Phrases can be a string containing single, multipart and dynamic phrases that will be broken up into a string array of single phrases the same way that VoiceAttack extracts them. For example, 'hello;hi;howdy' will result in an array with three elements: hello, hi and howdy. 'good[morning;day;night;gravy]' will result in an array with four elements: good morning, good day, good night, good gravy. The optional parameter, 'TrimSpaces' removes any leading and trailing spaces for each element (default is false). Note: Use of dynamic phrases with this function will result in spaces being trimmed regardless of this parameter.
ExtractPhrases（文字列フレーズ、bool TrimSpaces（オプション）、bool Lowercase（オプション））–このユーティリティ関数は、Phrasesで示される抽出されたフレーズを含む文字列配列を返します。フレーズVoiceAttackが抽出するのと同じ方法で、単一フレーズ、マルチパートフレーズ、および動的フレーズを含むストリングを単一フレーズのストリング配列に分割できます。たとえば、「hello; hi; howdy」は、hello、hi、howdyの3つの要素を持つ配列になります。'good [morning; day; night; gravy]'は、おはよう、おはよう、おやすみ、グレービーの4つの要素を持つ配列になります。オプションのパラメーター 'TrimSpaces'は、各要素の先頭と末尾のスペースを削除します（デフォルトはfalseです）。注：この機能で動的フレーズを使用すると、このパラメーターに関係なくスペースが削除されます。

The optional parameter, 'Lowercase' sets every element to lowercase (default is false).
オプションのパラメーター 'Lowercase'は、すべての要素を小文字に設定します（デフォルトはfalse）。

String[] myArray = vaProxy.Utility.ExtractPhrases("'good[morning;day;night;gravy]");
String [] myArray = vaProxy.Utility.ExtractPhrases(" 'good [morning; day; night; gravy]");

ParseTokens(String Value) – This is just a shortcut to getting values from any of the available tokens.
ParseTokens（文字列値）–これは、利用可能なトークンのいずれかから値を取得するための単なるショートカットです。

String s = vaProxy.Utility.ParseTokens("{ACTIVEWINDOWTITLE}");
文字列s = vaProxy.Utility.ParseTokens(" {ACTIVEWINDOWTITLE}");

CapturedAudio(int type) - This returns a System.IO.MemoryStream (wave stream) that contains audio captured from VoiceAttack's input. The integer parameter is the type of audio that you want to retrieve:
CapturedAudio（int型）-これは、VoiceAttackの入力からキャプチャされたオーディオを含むSystem.IO.MemoryStream（ウェーブストリーム）を返します。整数パラメーターは、取得するオーディオのタイプです。

1. - The last recognized audio (that is, audio that was recognized as a spoken command phrase).
1. -最後に認識された音声（つまり、音声コマンドフレーズとして認識された音声）。

2. - The previous recognized audio. 2 - The latest unrecognized audio.
2. -以前に認識された音声。2 -最新の認識されない音声。

1. - The latest captured recognized or unrecognized audio.
1. -最新のキャプチャされた認識済みまたは認識されていないオーディオ。

2. - The previous captured recognized or unrecognized audio.
2. -以前にキャプチャした認識済みまたは未認識の音声。

3. - Dictation audio (dictation audio captured while dictation mode is turned on).
3. -ディクテーションオーディオ（ディクテーションモードがオンになっている間にキャプチャされたディクテーションオーディオ）。

This example shows how to play back the latest recognized audio (note the 0 parameter):
次の例は、認識された最新のオーディオを再生する方法を示しています（0パラメーターに注意してください）。

```
using (System.IO.MemoryStream ms = vaProxy.Utility.CapturedAudio(0))
使用（System.IO。MemoryStreamをするMS = vaProxy.Utility.CapturedAudio（0））

{
{

if (ms == null) return;
if （ms == null ）return ;

using (System.Media.SoundPlayer sp = new System.Media.SoundPlayer(ms))
使用して（System.Media。SoundPlayerのSP = 新しいSystem.Media。SoundPlayerの（MS））

{
{
```

```
sp.Play();
sp.Play();


}
}


}
}
```

Note that you will have to dispose of the memory stream yourself. See also, 'Captured Audio' feature earlier in this document.
メモリストリームは自分で破棄する必要があることに注意してください。このドキュメントで前述した「キャプチャされたオーディオ」機能も参照してください。

ResetSpeechEngine() - This resets the current speech engine (without resetting the entire profile).
ResetSpeechEngine()-これにより、現在の音声エンジンがリセットされます（プロファイル全体はリセットされません）。

```
vaProxy.Utility.ResetSpeechEngine();
vaProxy.Utility.ResetSpeechEngine();
```

GetSpeechRecordingDeviceMute() - This Boolean function return true if the device that the speech engine is currently using is muted and false if it is not.
GetSpeechRecordingDeviceMute()-このブール関数は、音声エンジンが現在使用しているデバイスがミュートされている場合はtrueを返し、そうでない場合はfalseを返します。

SetSpeechRecordingDeviceMute(Boolean Mute) - This sets the mute state of the recording device that the speech engine is currently using. Passing a true value mutes the device. Passing a false value unmutes the device.
SetSpeechRecordingDeviceMute(Boolean Mute)-音声エンジンが現在使用している録音デバイスのミュート状態を設定します。真の値を渡すと、デバイスがミュートされます。false値を渡すと、デバイスのミュートが解除されます。

ActiveWindowTitle() - This string function returns the active window's title text.
ActiveWindowTitle()-この文字列関数は、アクティブウィンドウのタイトルテキストを返します。

ActiveWindowProcessName() - This string function returns the active window's process name (the one you see in Task Manager).
ActiveWindowProcessName()-この文字列関数は、アクティブウィンドウのプロセス名（タスクマネージャーに表示される名前）を返します。

ActiveWindowProcessID() – Returns the active window's process id (the one you see in Task Manager details) as an integer.
ActiveWindowProcessID()–アクティブウィンドウのプロセスID(タスクマネージャーの詳細に表示されるもの)を整数として返します。

ActiveWindowPath() – This string function returns the path of the active window's executable.
ActiveWindowPath()–この文字列関数は、アクティブウィンドウの実行可能ファイルのパスを返します。

ActiveWindowWidth() – Returns the active window's width as an integer. Helps with resizing/moving.
ActiveWindowWidth()–アクティブウィンドウの幅を整数として返します。サイズ変更/移動を支援します。

ActiveWindowHeight() – Returns the active window's height as an integer. Helps with resizing/moving.
ActiveWindowHeight()–アクティブウィンドウの高さを整数として返します。サイズ変更/移動を支援します。

ActiveWindowLeft() – Returns the active window's left (X coordinate) as an integer. Helps with resizing/moving.
ActiveWindowLeft()–アクティブウィンドウの左(X座標)を整数として返します。サイズ変更/移動を支援します。

ActiveWindowTop() – Returns the active window's top (Y coordinate) as an integer. Helps with resizing/moving.
ActiveWindowTop()–アクティブウィンドウの上部(Y座標)を整数として返します。サイズ変更/移動を支援します。

ActiveWindowRight() – Returns the active window's right (left + width) as an integer. Helps with resizing/moving.
ActiveWindowRight()–アクティブウィンドウの右(左+幅)を整数として返します。サイズ変更/移動を支援します。

ActiveWindowBottom() – Returns the active window's bottom (top + height) as an integer. Helps with resizing/moving.
ActiveWindowBottom()–アクティブウィンドウの下部(上部+高さ)を整数として返します。サイズ変更/移動を支援します。

ProcessExists(String ProcessName) – This Boolean function returns true if a process with the name specified in textVariable exists. Returns false if not. Note that this can take wildcards (*). For instance, if ProcessName contains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
ProcessExists(String ProcessName)– textVariableで指定された名前のプロセスが存在する場合、このブール関数はtrueを返します。そうでない場合はfalseを返します。これにはワイルドカード(*)を使用できることに注意してください。たとえば、ProcessNameに「* notepad *」(引用符なし)が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

ProcessCount(String ProcessName) – Returns an integer value of 1 or more if processes with the name specified in ProcessName exist. Returns 0 if there are none. Note that this can take wildcards (*). For instance, if ProcessName contains '*notepad*' (without quotes), the search will look for processes that have a name that contains 'notepad'.
ProcessCount（String ProcessName）– ProcessNameで指定された名前のプロセスが存在する場合、1以上の整数値を返します。存在しない場合は0を返します。これにはワイルドカード（*）を使用できることに注意してください。たとえば、ProcessNameに「＊ notepad ＊」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

ProcessForeground(String ProcessName) – This Boolean function returns true if a process with a main window with the title specified in ProcessName is the foreground window. Returns false if not. Note that this can take wildcards (*) for window titles that change. For instance, if ProcessName contains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
ProcessForeground（String ProcessName）– ProcessNameで指定されたタイトルを持つメインウィンドウを持つプロセスがフォアグラウンドウィンドウである場合、このブール関数はtrueを返します。そうでない場合はfalseを返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、ProcessNameに「＊ notepad ＊」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

ProcessMinimized(String ProcessName) – Returns a Boolean value of true if a process with a main window with the title specified in ProcessName is minimized. Returns false if not. Note that this can take wildcards (*) for window titles that change. For instance, if ProcessName contains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
ProcessMinimized（String ProcessName）– ProcessNameで指定されたタイトルのメインウィンドウを持つプロセスが最小化されている場合、trueのブール値を返します。そうでない場合はfalseを返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、ProcessNameに「＊ notepad ＊」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

ProcessMaximized(String ProcessName) – Returns a Boolean value of true if a process with a main window with the title specified in ProcessName is maximized. Returns false if not. Note that this can take wildcards (*) for window titles that change. For instance, if ProcessName contains '*notepad*' (without quotes), the search will look for a process that has a name that contains 'notepad'.
ProcessMaximized（String ProcessName）–ProcessNameで指定されたタイトルのメインウィンドウを持つプロセスが最大化される場合、trueのブール値を返します。そうでない場合はfalseを返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、ProcessNameに「＊ notepad ＊」（引用符なし）が含まれている場合、検索では「notepad」を含む名前のプロセスが検索されます。

WindowExists(String WindowName) – Returns a Boolean value of true if a window with the title specified in WindowName exists. Returns false if not. Note that this can take wildcards (*) for window titles that change. For instance, if WindowName contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
WindowExists（String WindowName） –WindowNameで指定されたタイトルのウィンドウが存在する場合、ブール値trueを返します。そうでない場合はfalseを返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、WindowNameに '＊ notepad ＊'（引用符なし）が含まれている場合、検索では 'notepad' を含むタイトルを持つウィンドウが検索されます。

WindowForeground(String WindowName) – Returns a Boolean value of true if a window with the title specified in WindowName is the foreground window. Returns false if not.
WindowForeground（String WindowName）–WindowNameで指定されたタイトルを持つウィンドウがフォアグラウンドウィンドウである場合、ブール値trueを返します。そうでない場合はfalseを返します。

Note that this can take wildcards (*) for window titles that change. For instance, if WindowName contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、WindowNameに '* notepad *'（引用符なし）が含まれている場合、検索では 'notepad'を含むタイトルを持つウィンドウが検索されます。

WindowMinimized(String WindowName) – Returns a Boolean value of true if a window with the title specified in WindowName is minimized. Returns false if not. Note that this can take wildcards (*) for window titles that change. For instance, if textVariable contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
WindowMinimized（String WindowName）–WindowNameで指定されたタイトルのウィンドウが最小化されている場合、ブール値trueを返します。そうでない場合はfalseを返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、textVariableに「* notepad *」（引用符なし）が含まれている場合、検索では「notepad」を含むタイトルを持つウィンドウが検索されます。

WindowMaximized(String WindowName) – Returns a Boolean value of true if a window with the title specified in WindowName is maximized. Returns false if not. Note that this
WindowMaximized（String WindowName）–WindowNameで指定されたタイトルを持つウィンドウが最大化されている場合、trueのブール値を返します。そうでない場合はfalseを返します。これに注意してください

can take wildcards (*) for window titles that change. For instance, if WindowName contains '*notepad*' (without quotes), the search will look for a window that has a title that contains 'notepad'.
変更されるウィンドウタイトルにはワイルドカード（*）を使用できます。たとえば、WindowNameに '* notepad *'（引用符なし）が含まれている場合、検索では 'notepad'を含むタイトルを持つウィンドウが検索されます。

WindowCount(String WindowName) – Returns an integer value of 1 or more if windows with the title specified in WindowName exist. Returns 0 if there are none. Note that this can take wildcards (*) for window titles that change. For instance, if WindowName contains '*notepad*' (without quotes), the search will look for windows that have a title that contains 'notepad'.
WindowCount（String WindowName）–WindowNameで指定されたタイトルのウィンドウが存在する場合、1以上の整数値を返します。存在しない場合は0を返します。変更するウィンドウタイトルにはワイルドカード（*）を使用できることに注意してください。たとえば、WindowNameに「* notepad *」（引用符なし）が含まれる場合、検索では「notepad」を含むタイトルを持つウィンドウが検索されます。

WindowTitleUnderMouse() – Returns the title for the window that is currently located under the mouse as a string.
WindowTitleUnderMouse（）–現在マウスの下にあるウィンドウのタイトルを文字列として返します。

WindowProcessUnderMouse() – Returns the process name for the window that is currently located under the mouse as a string.
WindowProcessUnderMouse()–現在マウスの下にあるウィンドウのプロセス名を文字列として返します。

CommandTargetForeground() – Returns a Boolean value of true if the current command's target is the foreground window. Returns false if the target is not the foreground window or if the target does not exist.
CommandTargetForeground()–現在のコマンドのターゲットがフォアグラウンドウィンドウの場合、ブール値trueを返します。ターゲットがフォアグラウンドウィンドウでない場合、またはターゲットが存在しない場合はfalseを返します。

CommandTargetMinimized() – Returns a Boolean value of true if the current command's target is minimized. Returns false if the target is not minimized or if the target does not exist.
CommandTargetMinimized()–現在のコマンドのターゲットが最小化されている場合、ブール値trueを返します。ターゲットが最小化されていない場合、またはターゲットが存在しない場合はfalseを返します。

CommandTargetMaximized() – Returns a Boolean value of true if the current command's target is maximized. Returns false if the target is not maximized or if the target does not exist.
CommandTargetMaximized()–現在のコマンドのターゲットが最大化されている場合、ブール値trueを返します。ターゲットが最大化されていない場合、またはターゲットが存在しない場合はfalseを返します。

MousePositionScreenX() – Returns the X coordinate of the mouse position as it relates to the screen as an integer. Zero would be the top-left corner of the screen.
MousePositionScreenX()–画面に関連するマウス位置のX座標を整数として返します。ゼロは、画面の左上隅になります。

MousePositionScreenY() – Returns the Y coordinate of the mouse position as it relates to the screen as an integer. Zero would be the top-left corner of the screen.
MousePositionScreenY()–画面に関連するマウス位置のY座標を整数として返します。ゼロは、画面の左上隅になります。

MousePositionWindowX() – Returns the X coordinate of the mouse position as it relates to the active window as an integer. Zero would be the top-left corner of the window.
MousePositionWindowX()–アクティブウィンドウに関連するマウス位置のX座標を整数として返します。ゼロは、ウィンドウの左上隅になります。

MousePositionWindowY() – Returns the Y coordinate of the mouse position as it relates to the active window as an integer. Zero would be the top-left corner of the window.
MousePositionWindowY()–アクティブウィンドウに関連するマウス位置のY座標を整数として返します。ゼロは、ウィンドウの左上隅になります。

CapsLockOn() – Returns a Boolean value of true if the caps lock key is locked. NumLockOn() – Returns a Boolean value of true if the numlock key is locked. ScrollLockOn() – Returns a Boolean value of true if the scroll lock key is locked.
CapsLockOn()– Caps Lockキーがロックされている場合、trueのブール値を返します。NumLockOn()–numlockキーがロックされている場合、trueのブール値を返します。ScrollLockOn()–スクロールロックキーがロックされている場合、trueのブール値を返します。

vaProxy.Command Attributes
vaProxy.Commandの属性

Below are the attributes that belong to the Command object of vaProxy. These attributes pertain to commands (including the command that has invoked the plugin function or inline function).
以下は、vaProxyのCommandオブジェクトに属する属性です。これらの属性は、コマンド(プラグイン関数またはインライン関数を呼び出したコマンドを含む)に関連しています。

Name() – returns a string that indicates the command that is running. If the command was executed using a spoken phrase, what was spoken will be returned. If the command is executed using any other means (keyboard, mouse, joystick, etc.) the full command name will be returned (works exactly like the '{CMD}' token).
Name()–実行中のコマンドを示す文字列を返します。音声フレーズを使用してコマンドが実行された場合、音声が返されます。コマンドが他の手段(キーボード、マウス、ジョイスティックなど)を使用して実行される場合、完全なコマンド名が返されます('{CMD}'トークンとまったく同じように機能します)。

```
string commandName = vaProxy.Command.Name();
string commandName = vaProxy.Command.Name();
```

InternalID() – returns a nullable Guid (Guid?) that indicates the internal id specified by the profile author.
InternalID()–プロファイル作成者によって指定された内部IDを示すNULL可能GUID(Guid?)を返します。

```
Guid? commandId = vaProxy.Command. InternalID();
ガイド? commandId = vaProxy.Command。InternalID();
```

```
if (commandId.HasValue) //if the internal id was indicated
if(commandId.HasValue)//内部IDが示された場合
```

```
{
{
```

```
//do something with the ID here
//ここでIDを使用して何かをする
```

}
}

Segment(int iSegment) – If your command contains, ‘dynamic command sections’ (see, ‘Dynamic Command Sections’ in the, ‘Command Screen’ documentation above), you can retrieve specific portions of the spoken command, indicated by its numeric position. This will allow you to make more precise decisions based on what was spoken.
セグメント（int iSegment）–コマンドに「動的コマンドセクション」が含まれている場合（上記の「コマンド画面」ドキュメントの「動的コマンドセクション」を参照）、数値位置で示される音声コマンドの特定の部分を取得できます。。これにより、話された内容に基づいてより正確な決定を下すことができます。

For instance, let’s say you have a complex dynamic command that you use to build several kinds of items like this: ‘build [1..10][bulldogs;wolves;strikers][please;]’ Then, you say, ‘build 5 bulldogs’ to execute that command. To find out what was built, you can check Segment(2) (note that the specified, ‘segment’ is zero-based. So, the first segment is 0.
たとえば、次のようないくつかの種類のアイテムを作成するために使用する複雑な動的コマンドがあるとします。
'build[1..10] [bulldogs; wolves; strikers] [please;] 'そのコマンドを実行する5つのブルドッグ。構築されたものを調べるには、Segment（2）をチェックできます（指定された「セグメント」はゼロベースです。したがって、最初のセグメントは0です。

The second is 1, and so on). The returned value will be, ‘bulldogs’. Then, you can find out how many items to build by checking Segment(1). The returned value will be, ‘5’ (which you can convert and use in a loop, for instance). For bonus points, you can check Segment(3) and see if, ‘please’ was spoken and then thank the user for being so polite (or chide them if they didn’t say, ‘please’) ;)
2番目は1などです）。返される値は「bulldogs」です。次に、Segment（1）をチェックすることで、構築するアイテムの数を調べることができます。返される値は「5」です（たとえば、変換してループで使用できます）。ボーナスポイントについては、Segment（3）をチェックし、「please」が話されたかどうかを確認してから、ユーザーがとても礼儀正しくしてくれたことに感謝します（または「please」と言わなかった場合はチードします）;)

Action() – returns a string to indicate the method by which the current command was executed. The possible results are, 'Spoken', 'Keyboard', 'Joystick', 'Mouse', 'Profile', 'External', 'Unrecognized', ‘ProfileUnloadChange’, ‘ProfileUnloadClose’, ‘DictationRecognized’, ‘Plugin’ and 'Other'. The value will be, 'Spoken' if the command was executed by a spoken phrase, 'Keyboard' if the command was executed using a keyboard shortcut, 'Joystick' if executed by a joystick button, 'Mouse' if executed by a mouse button click and 'Profile' if the command was executed on profile load (from the command indicated in the profile options screen). The value will be 'External' if the command is executed from a command line or if you right-click on a command on the profile screen and execute it from there. 'Unrecognized' will be the value if the command was executed using the unrecognized phrase catch-all command in the profile options screen. ‘ProfileUnloadChange’ and ‘ProfileUnloadClose’ will be returned if the command is executed as part of the profile being unloaded, either by changing the profile or by VoiceAttack shutting down. ‘DictationRecognized’ is returned if the command was invoked as a result of a dictation phrase being recognized. A value of, ‘Plugin’ is returned if the
アクション() –現在のコマンドが実行された方法を示す文字列を返します。考えられる結果は、「音声」、「キーボード」、「ジョイスティック」、「マウス」、「プロファイル」、「外部」、「認識できない」、「ProfileUnloadChange」、「ProfileUnloadClose」、「DictationRecognized」、「プラグイン」および「その他」。値は、コマンドが音声フレーズによって実行された場合は「音声」、キーボードショートカットを使用してコマンドが実行された場合は「キーボード」、ジョイスティックボタンによって実行された場合は「ジョイスティック」、マウスボタンによって実行された場合は「マウス」プロファイルのロード時にコマンドが実行された場合は、[プロファイル]をクリックします（プロファイルオプション画面に表示されるコマンドから）。値は「外部」になります コマンドがコマンドラインから実行される場合、またはプロファイル画面でコマンドを右クリックしてそこから実行する場合 プロファイルオプション画面で認識されないフレーズcatch-allコマンドを使用してコマンドが実行された場合、「認識されない」が値になります。プロファイルを変更するかVoiceAttackをシャットダウンすることにより、コマンドがアンロードされるプロファイルの一部として実行される場合、「ProfileUnloadChange」および「ProfileUnloadClose」が返されます。ディクテーションフレーズが認識された結果としてコマンドが呼び出された場合、「DictationRecognized」が返されます。「Plugin」の値は、プロファイルを変更するかVoiceAttackをシャットダウンすることにより、コマンドがアンロードされるプロファイルの一部として実行される場合、「ProfileUnloadChange」および「ProfileUnloadClose」が返されます。ディクテーションフレーズが認識された結果としてコマンドが呼び出された場合、「DictationRecognized」が返されます。「Plugin」の値は、プロファイルを変更するかVoiceAttackをシャットダウンすることにより、コマンドがアンロードされるプロファイルの一部として実行される場合、「ProfileUnloadChange」および「ProfileUnloadClose」が返されます。ディクテーションフレーズが認識された結果としてコマンドが呼び出された場合、

command is executed from a plugin or inline function. 'Other' is reserved.
コマンドはプラグインまたはインライン関数から実行されます。「その他」は予約されています。

Before() – returns a string. When using wildcards in spoken phrases, this is the text that occurs before the wildcard phrase. For example, if using, '*rocket*' as your wildcard phrase and you say, 'i am going for a ride in my rocket ship' Before() will contain, 'i am going for a ride in my' and After() will contain 'ship'.
Before()–文字列を返します。音声フレーズでワイルドカードを使用する場合、これはワイルドカードフレーズの前にあるテキストです。たとえば、ワイルドカードフレーズとして「* rocket *」を使用し、「ロケット船に乗るつもりだ」と言う場合、Before()は「iに乗るつもりです」およびAfter()には「ship」が含まれます。

After() – returns a string. When using wildcards in spoken phrases, this is the text that occurs after the wildcard phrase.
After()–文字列を返します。音声フレーズでワイルドカードを使用する場合、これはワイルドカードフレーズの後に現れるテキストです。

WildcardKey() – returns a string. When using wildcards in spoken phrases, this is the text that is not variable. Using the, ‘*rocket*’ example above, the value returned by this function will be, ‘rocket’.
WildcardKey()–文字列を返します。音声フレーズでワイルドカードを使用する場合、これは可変ではないテキストです。上記の「* rocket *」の例を使用すると、この関数によって返される値は「rocket」になります。

Confidence() – returns an int that indicates the confidence level that the speech engine provides when the speech engine detects speech. The value range for a spoken command is from 0 to 100. This value will always be 0 when a command is not executed as a spoken command (use the, 'Action()' function to check how the command was executed). Note this value is accessible from within the specified unrecognized catch-all command.
Confidence()-音声エンジンが音声を検出したときに音声エンジンが提供する信頼レベルを示すintを返します。音声コマンドの値の範囲は0～100です。コマンドが音声コマンドとして実行されない場合、この値は常に0になります(コマンドの実行方法を確認するには、'Action()'関数を使用します)。この値は、指定された認識されないcatch-allコマンド内からアクセスできることに注意してください。

MinConfidence() – returns and int that indicates the minimum confidence level set by the user as it applies to the executing command. The value range is 0 to 100. This value will be 0 if the minimum confidence level is not set. Note that this function can return a value even if the command is not spoken.
MinConfidence()-実行中のコマンドに適用されるときにユーザーが設定した最小信頼レベルを示すintを返します。値の範囲は0～100です。最小信頼レベルが設定されていない場合、この値は0になります。この関数は、コマンドが話されていなくても値を返すことができることに注意してください。

IsSubcommand() – a Boolean that will be true if the currently-executing command is executing as a subcommand (a subcommand is a command that is executed by another command). The returned value will be false if the command is not a subcommand.
IsSubcommand()-現在実行中のコマンドがサブコマンドとして実行されている場合に真になるブール(サブコマンドは別のコマンドによって実行されるコマンドです)。コマンドがサブコマンドでない場合、戻り値はfalseになります。

WhenISay() – returns a string that is the full value of what is indicated in the, 'When I Say' input box on the command screen.
WhenISay()-コマンド画面の「When I Say」入力ボックスに示されているものの完全な値である文字列を返します。

IsListeningOverride() – returns a Boolean value of true if the executing command was invoked by a listening override keyword (for instance, if you executed the command by saying, 'Computer, Open Door' instead of, 'Open Door'). Otherwise, the result will be false.
IsListeningOverride()-リスニングオーバーライドキーワードによって実行中のコマンドが呼び出された場合(たとえば、'Open Door'ではなく 'Computer、Open Door'と言ってコマンドを実行した場合)、ブール値trueを返します。そうでない場合、結果は偽になります。

IsComposite() – returns a Boolean value of true if the executing command is composite (where there is a prefix and a suffix). Otherwise, it will be false.
IsComposite()-実行中のコマンドがコンポジット(プレフィックスとサフィックスがある場合)の場合、ブール値trueを返します。そうでなければ、それは偽になります。

PrefixPart() – returns a string. If an executing command is composite (where there is a prefix and a suffix), this function will return the prefix portion of the command. If called within a non-composite command, this will result in a blank string.
PrefixPart()-文字列を返します。実行中のコマンドが複合である場合（プレフィックスとサフィックスがある場合）、この関数はコマンドのプレフィックス部分を返します。非複合コマンド内で呼び出された場合、これは空の文字列になります。

SuffixPart() – returns a string. If an executing command is composite (where there is a prefix and a suffix), this function will return the suffix portion of the command. If called within a non-composite command, this will result in a blank string.
SuffixPart()-文字列を返します。実行中のコマンドがコンポジット（プレフィックスとサフィックスがある場合）の場合、この関数はコマンドのサフィックス部分を返します。非複合コマンド内で呼び出された場合、これは空の文字列になります。

CompositeGroup() – returns a string. If an executing command is composite (where there
CompositeGroup()-文字列を返します。実行中のコマンドが複合の場合（ある場所）

is a prefix and a suffix), this function will return the group value if it is being used.
は接頭辞と接尾辞です）、この関数は使用されている場合はグループ値を返します。

Category() – returns a string that indicates the category of the executing command. If this token is used on a composite command (a command using a prefix and suffix), the result will be the category of the prefix and the category of the suffix separated by a space. For the individual parts of a composite category, see, 'PrefixCategory()' and 'SuffixCategory()' below.
Category()-実行中のコマンドのカテゴリを示す文字列を返します。このトークンが複合コマンド（プレフィックスとサフィックスを使用するコマンド）で使用される場合、結果は、スペースで区切られたプレフィックスのカテゴリとサフィックスのカテゴリになります。複合カテゴリの個々の部分については、以下の「PrefixCategory()」および「SuffixCategory()」を参照してください。

PrefixCategory() – returns a string that indicates the prefix category of the executing command (if the executing command is a composite command).
PrefixCategory()-実行中のコマンドのプレフィックスカテゴリを示す文字列を返します（実行中のコマンドが複合コマンドの場合）。

SuffixCategory() – returns a string that indicates the suffix category of the executing command (if the executing command is a composite command).
SuffixCategory()-実行中のコマンドのサフィックスカテゴリを示す文字列を返します（実行中のコマンドが複合コマンドの場合）。

SetSessionEnabled(string CommandName, Boolean Enabled) – This function temporarily sets the enabled state of a command (indicated by name) during the current session (that is, while VoiceAttack is running – the setting is not saved). If a command is able to be executed (by voice, keyboard shortcut, mouse button, etc.), setting Enabled to false will temporarily disable the command from executing. Setting Enabled to true again will re– enable the command. Note: You will still be able to execute the command from right– clicking on the, 'Execute' menu item from the command list.
SetSessionEnabled(string CommandName、Boolean Enabled)–この関数は、現在のセッション中(つまり、VoiceAttackの実行中–設定は保存されない)にコマンドの有効状態(名前で表示)を一時的に設定します。コマンドを(音声、キーボードショートカット、マウスボタンなどによって)実行できる場合、Enabledをfalse に設定すると、コマンドの実行が一時的に無効になります。Enabledを再度trueに設定すると、コマンドが再度有効になります。注:コマンドリストの[実行]メニュー項目を右クリックしても、コマンドを実行できます。

SetSessionEnabled(Guid InternalID, Boolean Enabled) – This function works exactly like the one above except that the command to be affected is located by its InternalID.
SetSessionEnabled(Guid InternalID、Boolean Enabled)–この関数は、影響を受けるコマンドがそのInternalIDによって特定されることを除いて、上記の関数とまったく同じように機能します。

GetSessionEnabled(string CommandName) – This Boolean function returns the enabled state of a command indicated by CommandName. This function returns true if the indicated command is enabled during the current session and false if the command is temporarily disabled (see, 'SetSessionEnabled' above). The command is located by its name specified in CommandName.
GetSessionEnabled(文字列CommandName)–このブール関数は、CommandNameで示されるコマンドの有効状態を返します。この関数は、指定されたコマンドが現在のセッション中に有効になっている場合はtrueを返し、コマンドが一時的に無効になっている場合はfalseを返します(上記の「SetSessionEnabled」を参照)。コマンドは、CommandNameで指定された名前で検索されます。

GetSessionEnabled(Guid InternalID) – This Boolean function works exactly like, 'GetSessionEnabled' above, except the command is located by its InternalID.
GetSessionEnabled(Guid InternalID)–このブール関数は、コマンドがそのInternalIDによって検索されることを除いて、上記の 'GetSessionEnabled'とまったく同じように機能します。

SetSessionEnabledByCategory(string CategoryName, Boolean Enabled) – This function temporarily sets the enabled state of a set of commands (indicated by category name) during the current session (that is, while VoiceAttack is running – the setting is not saved). If a command is able to be executed (by voice, keyboard shortcut, mouse button, etc.), setting Enabled to false will temporarily disable the all commands of a given category from executing. Setting Enabled to true again will re–enable the commands. Important: If you edit the category of a command affected by this function, you will need to re–execute this function in order for the changes to be effective.
SetSessionEnabledByCategory(string CategoryName、Boolean Enabled)–この関数は、現在のセッション中(つまり、VoiceAttackの実行中–設定は保存されません)に、一連のコマンド(カテゴリ名で表示)の有効状態を一時的に設定します。コマンドを(音声、キーボードショートカット、マウスボタンなどによって)実行できる場合、Enabledをfalse に設定すると、特定のカテゴリのすべてのコマンドの実行が一時的に無効になります。Enabled を再度trueに設定すると、コマンドが再び有効になります。重要:この機能の影響を受けるコマンドのカテゴリーを編集する場合、変更を有効にするには、この機能を再実行する必要があります。

Note: You will still be able to execute the affected commands from right–clicking on the, 'Execute' menu item from the command list.
注:コマンドリストの[実行]メニュー項目を右クリックすると、影響を受けるコマンドを実行できます。

GetSessionEnabledByCategory(string CategoryName) – This Boolean function returns true if the indicated category is enabled and false if the category has been set to be temporarily disabled during the current session (see, ‘SetSessionEnabledByCategory’ above).
GetSessionEnabledByCategory（string CategoryName）–このブール関数は、示されたカテゴリが有効な場合はtrueを返し、現在のセッション中にカテゴリが一時的に無効に設定されている場合はfalseを返します（上記の「SetSessionEnabledByCategory」を参照）。

Execute (string CommandName, optional Boolean WaitForReturn, optional Boolean AsSubcommand) – This method will execute a VoiceAttack command in the active profile by the name indicated in CommandName. The flow of execution will continue immediately if you pass in false (the default) for WaitForReturn. If you want the flow of execution to wait until the command completes, pass in true. Pass a true value as the AsSubcommand parameter to execute the command in the context of a subcommand to the calling command. This will allow the executed command to gain certain attributes of the calling command, such as command–shared variables and also allows the command to execute if the calling command is called synchronously. Notes – If the command does not exist, the log will display an alert. If you execute a command using this method within a proxy or plugin initialization method or by clicking the, ‘Test’ button of the Inline Function editor, AsSubcommand will be ignored, since the execution will not be within the context of an executing command.
Execute（文字列CommandName、オプションのブール値WaitForReturn、オプションのブール値AsSubcommand）–このメソッドは、CommandNameで示された名前でアクティブなプロファイルでVoiceAttackコマンドを実行します。WaitForReturnにfalse（デフォルト）を渡すと、実行のフローはすぐに継続します。コマンドが完了するまで実行フローを待機させる場合は、trueを渡します。AsSubcommandとして真の値を渡す呼び出しコマンドのサブコマンドのコンテキストでコマンドを実行するパラメーター。これにより、実行されたコマンドは、コマンド共有変数など、呼び出し元コマンドの特定の属性を取得でき、呼び出し元コマンドが同期的に呼び出された場合にコマンドを実行できます 注–コマンドが存在しない場合、ログにアラートが表示されます。プロキシまたはプラグインの初期化メソッド内でこのメソッドを使用して、またはインライン関数エディターの「テスト」ボタンをクリックしてコマンドを実行すると、実行は実行中のコマンドのコンテキスト内にないため、AsSubcommandは無視されます。

vaProxy.Command.Execute(“fire weapons”); //continues immediately (asynchronous)
vaProxy.Command.Execute (“fire weapons”, true); //waits until the command com– pletes
vaProxy.Command.Execute(“ fire weapons”); //すぐに継続（非同期）vaProxy.Command.Execute(“ fire weapons”、true); //コマンドが完了するまで待機します

vaProxy.Command.Execute (“fire weapons”, true, true); //waits until the command
vaProxy.Command.Execute(″fire weapons″、true、true); //コマンドが実行されるまで待機します

completes and also execute the command as a subcommand
サブコマンドとしてコマンドを完了し、実行します

Execute(Guid InternalID, optional Boolean WaitForReturn, optional Boolean AsSubcommand) – This method works exactly like the Execute method above, except this version locates the command to execute by InternalID.
Execute（Guid InternalID、オプションのブール値WaitForReturn、オプションのブール値AsSubcommand）–このメソッドは、上記のExecuteメソッドとまったく同じように機能しますが、このバージョンは、InternalIDによって実行するコマンドを見つけます。

Exists(string CommandName) – This Boolean function returns true if a command is available to the active profile with the name specified in CommandName.
Exists（string CommandName）– CommandNameで指定された名前を持つアクティブなプロファイルでコマンドが使用可能な場合、このブール関数はtrueを返します。

```
if (vaProxy.CommandExists(“fire weapons”))
if（vaProxy.CommandExists（“ fire weapons”））

{
{

vaProxy.ExecuteCommand(“fire weapons”);
vaProxy.ExecuteCommand（“ fire weapons”）;

}
}
```

Exists(Guid InternalID) – This Boolean function returns true if a command is available to the active profile with InternalID specified in InternalID.
Exists（Guid InternalID）–このBoolean関数は、InternalIDで指定されたInternalIDを持つアクティブなプロファイルに対してコマンドが使用可能な場合、trueを返します。

Active(string CommandName) – This Boolean function returns true if the command indicated by name in CommandName is actively executing (most likely long-running or running in a loop).
アクティブ（文字列CommandName）–このブール関数は、CommandNameの名前で示されるコマンドがアクティブに実行されている場合（ほとんどの場合、長時間実行またはループで実行中）にtrueを返します。

```
if (vaProxy.CommandActive(“fire weapons”) == false) //if command not active
if（vaProxy.CommandActive（“ fire weapons”）== false）//コマンドがアクティブでない場合

{
{

vaProxy.ExecuteCommand(“fire weapons”); //execute the command
vaProxy.ExecuteCommand（“ fire weapons”）; //コマンドを実行します

}
}
```

Active(Guid InternalID) – This Boolean function returns true if the indicated command is actively executing (most likely long-running or running in a loop). The difference between this function and the one above is that this version locates the command via InternalID and not by name.
アクティブ(Guid InternalID)-このブール関数は、指定されたコマンドがアクティブに実行されている場合(ほとんどの場合、長時間実行またはループで実行中)にtrueを返します。この関数と上記の関数の違いは、このバージョンでは、名前ではなくInternalIDを介してコマンドを検索することです。

CategoryExists(string CategoryName) – This Boolean function returns true if any command is available to the active profile that has a category with the name specified in CategoryName.
CategoryExists(string CategoryName)-このブール関数は、CategoryNameで指定された名前のカテゴリを持つアクティブなプロファイルでコマンドが使用可能な場合、trueを返します。

LastUserExec() – returns an int that indicates the number of seconds since the last command executed by the user – spoken phrase, keyboard key press, mouse click or joystick button press. That is, it does not include subcommands, commands executed externally, rightclick execute, etc.
LastUserExec()-ユーザーが最後にコマンドを実行してからの秒数を示すintを返します-音声フレーズ、キーボードのキーの押下、マウスのクリック、またはジョイスティックのボタンの押下。つまり、サブコマンド、外部で実行されるコマンド、右クリック実行などは含まれません。

ExecutionCount() – returns and int that is the count of top-level commands (commands that are not subcommands) that have executed since VoiceAttack had launched.
ExecutionCount()- VoiceAttackが起動してから実行されたトップレベルコマンド(サブコマンドではないコマンド)の数であるintを返します。

LastSpoken() – returns a string which indicates the last-spoken command phrase. Useful from within sub-commands where you need to know what was said to invoke the root command, or if you need to know what was spoken prior to the current command (if the current command was not spoken (executed by keyboard, mouse, joystick, external, etc.).
LastSpoken()-最後に話されたコマンドフレーズを示す文字列を返します。ルートコマンドを呼び出すために言われたことを知る必要があるサブコマンド内から、または現在のコマンドの前に話されたことを知る必要がある場合(現在のコマンドが話されていない場合(キーボード、マウス、ジョイスティックによって実行される場合)、外部など)。

PreviousSpoken() – returns a string that indicates the spoken command phrase prior to the last spoken command phrase.
PreviousSpoken()-最後の音声コマンドフレーズの前の音声コマンドフレーズを示す文字列を返します。

SpokenHistory(int value) – returns a string. This provides access to the history of spoken commands (up to a maximum of 1000 phrases), starting at 0. A value of zero (SpokenHistory(0)) is the same as getting the value of the LastSpoken() function. A value of 1 (SpokenHistory(1)) is the same as getting the value of PreviousSpoken(). A value of 2 would get the spoken phrase that was issued before the previous spoken command and so on. If no history exists for the value, a blank (empty string) is returned.
SpokenHistory（int value）–文字列を返します。これにより、0から始まる音声コマンドの履歴（最大1000フレーズ）にアクセスできます。値0（SpokenHistory（0））は、LastSpoken（）関数の値を取得するのと同じです。値1（SpokenHistory（1））は、PreviousSpoken（）の値を取得するのと同じです。値が2の場合、前の音声コマンドなどの前に発行された音声フレーズが取得されます。値の履歴が存在しない場合、空白（空の文字列）が返されます。

vaProxy.Profile Attributes
vaProxy.Profileの属性

Below are the attributes that belong to the Profile object of vaProxy. These attributes pertain to profile (including the profile that is currently active).
以下は、vaProxyのProfileオブジェクトに属する属性です。これらの属性は、プロファイルに関連しています（現在アクティブなプロファイルを含む）。

Name() – This function returns the name of the active profile.
名前（）–この関数は、アクティブなプロファイルの名前を返します。

string profileName = vaProxy.Profile.Name();
文字列profileName = vaProxy.Profile.Name（）;

InternalID() – This function returns a nullable Guid (Guid?) that indicates the InternalID value of the active profile. The profile's InternalID is an author flag (see, 'Author Flags' section).
InternalID（）–この関数は、アクティブなプロファイルのInternalID値を示すNULL可能GUID（Guid？）を返します。プロファイルのInternalIDは作成者フラグです（「作成者フラグ」セクションを参照）。

Exists(string ProfileName) – This Boolean function returns true if a profile is available with the name specified in ProfileName.
Exists（文字列ProfileName）–このブール関数は、ProfileNameで指定された名前のプロファイルが利用可能な場合にtrueを返します。

Exists(Guid InternalID) – This Boolean function returns true if a profile is available with the InternalID specified in InternalID.
Exists（Guid InternalID）–このブール関数は、InternalIDで指定されたInternalIDでプロファイルが使用可能な場合にtrueを返します。

AuthorTag1(), AuthorTag2(), AuthorTag3() – Returns the value for AuthorTag1, AuthorTag2 and AuthorTag3 (see, 'Author Flags' section).
AuthorTag1()、AuthorTag2()、AuthorTag3()– AuthorTag1、AuthorTag2、およびAuthorTag3の値を返します(「Authorフラグ」セクションを参照)。

AuthorID() – Returns the value for AuthorID (see, 'Author Flags' section). The return type for this function is nullable Guid (Guid?).
AuthorID()– AuthorID の値を返します(「Author Flags」セクションを参照)。この関数の戻り値の型はnull値を許可するGuid(Guid?)です。

ProductID() – Returns the value for ProductID (see, 'Author Flags' section). The return type for this function is nullable Guid (Guid?).
ProductID()– ProductID の値を返します(「作成者フラグ」セクションを参照)。この関数の戻り値の型はnull値を許可するGuid(Guid?)です。

PreviousName() – returns a string that indicates the name of the profile that was loaded prior to the currently-loaded profile.
PreviousName()–現在ロードされているプロファイルの前にロードされたプロファイルの名前を示す文字列を返します。

PreviousInternalID() – This function returns a nullable Guid (Guid?) that indicates the InternalID value of the profile that was loaded prior to the currently-loaded profile. A profile's InternalID is an author flag (see, 'Author Flags' section).
PreviousInternalID()–この関数は、現在ロードされているプロファイルの前にロードされたプロファイルのInternalID値を示すNULL可能GUID(Guid?)を返します。プロファイルのInternalIDは作成者フラグです(「作成者フラグ」セクションを参照)。

PreviousAuthorTag1(), PreviousAuthorTag2(), PreviousAuthorTag3() – These return the three different AuthorTags of the profile that was loaded prior to the currently-loaded profile (see, 'VoiceAttack Author Flags' later on in this document). This is an advanced feature that will probably not be used by most.
PreviousAuthorTag1()、PreviousAuthorTag2()、PreviousAuthorTag3() –これらは、現在ロードされているプロファイルの前にロードされたプロファイルの3つの異なるAuthorTagsを返します(このドキュメントで後述する「VoiceAttack Author Flags」を参照)。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

PreviousAuthorID() – returns the value of AuthorID from the profile that was loaded prior to the currently-loaded profile. The return type of this function is nullable Guid (Guid?) (see, 'Author Flags' section).
PreviousAuthorID()–現在ロードされているプロファイルの前にロードされたプロファイルからAuthorIDの値を返します。この関数の戻り値の型はnull値を許可するGUID(Guid?)です(「Author Flags」セクションを参照)。

PreviousProductID() –returns the value of ProductID from the profile that was loaded prior to the currently–loaded profile. The return type of this function is nullable Guid (Guid?) (see, ‘Author Flags’ section).
PreviousProductID()–現在ロードされているプロファイルの前にロードされたプロファイルからProductIDの値を返します。この関数の戻り値の型はnull値を許可するGUID（Guid？）です（「Author Flags」セクションを参照）。

NextName() – returns a string that indicates the name of the profile that has been selected to load after the current profile is unloaded. This function will return as empty if the profile is not unloading or if the profile is unloading due to VA shutting down. Note: If you haven’t been able to tell by now, this function is only useful from within an unload command (see, ‘Execute a command each time this profile is unloaded’ earlier in this document).
NextName()–現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルの名前を示す文字列を返します。この関数は、プロファイルがアンロードされていない場合、またはVAのシャットダウンによりプロファイルがアンロードされている場合、空として返されます。注：今までに伝えることができなかった場合、この関数はアンロードコマンド内でのみ役立ちます（このドキュメントの「このプロファイルがアンロードされるたびにコマンドを実行する」を参照）。

NextInternalID() – This function returns a nullable Guid (Guid?) that indicates the InternalID value of the profile that has been selected to load after the current profile isi unloaded. InternalID is an author flag (see, ‘Author Flags’ section). Note: If you haven’t been able to tell by now, this function is only useful from within an unload command (see, ‘Execute a command each time this profile is unloaded’ earlier in this document).
NextInternalID ()–この関数は、現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルのInternalID値を示すNULL可能 GUID（Guid？）を返します。InternalIDは作成者フラグです（「作成者フラグ」セクションを参照）。注：今までに伝えることができなかった場合、この関数はアンロードコマンド内でのみ役立ちます（このドキュメントの「このプロファイルがアンロードされるたびにコマンドを実行する」を参照）。

NextAuthorTag1(),NextAuthorTag2(), NextAuthorTag3() – These return the three different AuthorTag values of the profile that has been selected to load after the current profile is unloaded. These functions will return as empty if the profile is not unloading or if the profile is unloading due to VA shutting down. Note: Just like NextName(), these functions are only useful from within an unload command (see, ‘Execute a command each time this profile is unloaded’ earlier in this document, as well as the, ‘Author Flags’ section later in this document). This is an advanced feature that will probably not be used by most.
NextAuthorTag1()、NextAuthorTag2()、NextAuthorTag3()–これらは、現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルの3つの異なるAuthorTag値を返します。これらの関数は、プロファイルがアンロードされていない場合、またはVAのシャットダウンによりプロファイルがアンロードされている場合、空として返されます。注：NextName()と同様に、これらの関数はアンロードコマンド内からのみ使用できます（このドキュメントの「このプロファイルがアンロードされるたびにコマンドを実行する」、およびこの後の「Author Flags」セクションを参照してください）資料）。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

NextAuthorID() – returns the value of AuthorID from the profile that has been selected to load after the current profile is unloaded. This function will return as null if the profile is not
NextAuthorID()–現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルからAuthorIDの値を返します。プロファイルがそうでない場合、この関数はnullを返します。

unloading or if the profile is unloading due to VA shutting down. The return type of this function is nullable Guid (Guid?) (see, 'Author Flags' section). Note: If you haven't been able to tell by now, this function is only useful from within an unload command (see, 'Execute a command each time this profile is unloaded' earlier in this document).
アンロード、またはVAのシャットダウンによりプロファイルがアンロードされている場合。この関数の戻り値の型はnull値を許可するGUID（Guid？）です（「Author Flags」セクションを参照）。注：今までに伝えることができなかった場合、この関数はアンロードコマンド内でのみ役立ちます（このドキュメントの「このプロファイルがアンロードされるたびにコマンドを実行する」を参照）。

NextProductID() – returns the value of ProductID from the profile that has been selected to load after the current profile is unloaded. This function will return as null if the profile is not unloading or if the profile is unloading due to VA shutting down. The return type of this function is nullable Guid (Guid?) (see, 'Author Flags' section). Note: If you haven't been able to tell by now, this function is only useful from within an unload command (see, 'Execute a command each time this profile is unloaded' earlier in this document).
NextProductID()–現在のプロファイルがアンロードされた後にロードするために選択されたプロファイルからProductIDの値を返します。この関数は、プロファイルがアンロードされていない場合、またはVAのシャットダウンによりプロファイルがアンロードされている場合、nullを返します。この関数の戻り値の型はnull値を許可するGUID（Guid？）です（「Author Flags」セクションを参照）。注：今までに伝えることができなかった場合、この関数はアンロードコマンド内でのみ役立ちます（このドキュメントの「このプロファイルがアンロードされるたびにコマンドを実行する」を参照）。

History(int value) – returns a string. This provides access to the history of the names of loaded profiles (up to a maximum of 1000 profiles), starting at 0. A value of zero (History(0)) is the same as getting the value of the Name() function. A value of 1 (History(1)) is the same as getting the value of PreviousName(). A value of 2 would get the name of the profile loaded before the previous profile and so on. If no history exists for the value, a blank (empty string) is returned.
History（int value）–文字列を返します。これにより、0から始まるロードされたプロファイルの名前（最大1000プロファイル）の履歴にアクセスできます。値0（History（0））は、Name（）関数の値を取得するのと同じです。値1（History（1））は、PreviousName（）の値を取得することと同じです。値が2の場合、前のプロファイルなどの前にロードされたプロファイルの名前が取得されます。値の履歴が存在しない場合、空白（空の文字列）が返されます。

HistoryInternalID(int value) – returns a nullable Guid?. This provides access to the history of the InternalID values of loaded profiles (up to a maximum of 1000 profiles), starting at 0. A value of zero (HistoryInternalID(0)) is the same as getting the value of the InternalID() function. A value of 1 (HistoryInternalID(1)) is the same as getting the value of PreviousInternalID(). A value of 2 would get the InternalID of the profile loaded before the previous profile and so on. If no history exists for the value, null is returned.
HistoryInternalID（int value）– null 値を許可するGuid？を返します。これにより、0から始まる、ロードされたプロファイル（最大1000プロファイル）のInternalID値の履歴へのアクセスが提供されます。値0（HistoryInternalID（0））は、InternalID（）関数の値を取得するのと同じです。 。値1（HistoryInternalID（1））は、PreviousInternalID（）の値を取得することと同じです。値が2の場合、前のプロファイルなどの前にロードされたプロファイルのInternalIDが取得されます。値の履歴が存在しない場合、nullが返されます。

HistoryAuthorID(int value) – returns a nullable Guid (Guid?). This provides access to the history of the AuthorID values of loaded profiles (up to a maximum of 1000 profiles), starting at 0. A value of zero (HistoryAuthorID(0)) is the same as getting the value of the AuthorID() function. A value of 1 (AuthorID(1)) is the same as getting the value of PreviousAuthorID(). A value of 2 would get the AuthorID of the profile loaded before the previous profile and so on. If no history exists for the value, null is returned.
HistoryAuthorID（int value）– null 値を許可するGUID（Guid？）を返します。これにより、ロードされたプロファイル（最大1000プロファイル）のAuthorID値の履歴へのアクセスが0から始まります。ゼロの値（HistoryAuthorID（0））はAuthorID（）関数の値を取得することと同じです。 。値1（AuthorID（1））は、PreviousAuthorID（）の値を取得することと同じです。値2は、前のプロファイルの前にロードされたプロファイルのAuthorIDを取得します。値の履歴が存在しない場合、nullが返されます。

HistoryAuthorTag1(int value), HistoryAuthorTag2(int value), HistoryAuthorTag3(int value) – This provides access to the history of the three AuthorTag values of loaded profiles (up to a maximum of 1000 profiles), starting at 0. A value of zero (HistoryAuthorTag1(0)) is the same as getting the value of the AuthorTag1() function. A value of 1 (HistoryAuthorTag1(1)) is the same as getting the value of PreviousAuthorTag1(). A value of 2 would get the Author Tags of the profile loaded before the previous profile and so on. If no history exists for the value, a blank (empty string) is returned. (See, 'VoiceAttack Author Flags' later on in this document). This is an advanced feature that will probably not be used by most.
HistoryAuthorTag1（int value）、 HistoryAuthorTag2（int value）、HistoryAuthorTag3（int value）–これにより、0から始まる、ロードされたプロファイルの3つのAuthorTag値の履歴（最大1000プロファイル）へのアクセスが提供されます。値0（HistoryAuthorTag1（0））はAuthorTag1（ ） 関数。値1（HistoryAuthorTag1（1））は、PreviousAuthorTag1（）の値を取得するのと同じです。値が2の場合、前のプロファイルなどの前にプロファイルの作成者タグがロードされます。値の履歴が存在しない場合、空白（空の文字列）が返されます。（このドキュメントで後述する「VoiceAttack Author Flags」を参照してください）。これは高度な機能であり、ほとんどの場合はおそらく使用されません。

SwitchTo(String ProfileName) – This function attempts to switch to the profile indicated by name in ProfileName. Note that switching profiles causes the current profile to cease execution.
SwitchTo（文字列プロファイル名） –この関数は、名前によって示されるプロファイルに切り替えるしようとプロファイル名。プロファイルを切り替えると、現在のプロファイルの実行が停止することに注意してください。

SwitchTo(Guid InternalID) – This function attempts to switch to the profile indicated by InternalID. Note that switching profiles causes the current profile to cease execution.
SwitchTo（Guid InternalID）–この関数は、InternalIDで示されるプロファイルへの切り替えを試みます。プロファイルを切り替えると、現在のプロファイルの実行が停止することに注意してください。

Reset() – This function reloads the current, active profile. Note that reloading the current profile will cause the profile's execution to cease.
Reset（）–この関数は、現在のアクティブなプロファイルをリロードします。現在のプロファイルを再ロードすると、プロファイルの実行が停止することに注意してください。

vaProxy.Queue Attributes
vaProxy.Queueの属性

Below are the attributes that belong to the Queue object of vaProxy. These attributes pertain to getting information about command queues that may be active.
以下は、vaProxyのQueueオブジェクトに属する属性です。これらの属性は、アクティブになっている可能性のあるコマンドキューに関する情報の取得に関連しています。

Count() – returns an int that indicates the number of command execution queues that are currently available.
Count()–現在使用可能なコマンド実行キューの数を示すintを返します。

```
int iQueueCount = vaProxy.Queue.Count();
int iQueueCount = vaProxy.Queue.Count();
```

Status(string Name) – returns a string that is the status of a command execution queue indicated in the Name parameter. The following values will be returned:
Status(string Name)– Nameパラメーターで示されるコマンド実行キューのステータスである文字列を返します。次の値が返されます。

'Not Initialized' – This will be the returned status if the queue does not exist (that is, no commands have been enqueued into a queue by the name given). 'Running' – The queue is currently running (not paused) and there is at least one command in the queue. 'Idle' – The queue is running (not paused) but there are zero items in the queue. 'Paused' – The queue is paused, or flagged to be paused if a command is currently executing within the queue. 'Stopped' – The queue is in a stopped state or has been flagged to stop if a command is currently executing within the queue.
「初期化されていません」–キューが存在しない場合(つまり、指定された名前でコマンドがキューにエンキューされていない場合)、返されたステータスになります。「実行中」–キューは現在実行されており(一時停止されていない)、キューに少なくとも1つのコマンドがあります。「アイドル」–キューは実行されています(一時停止されていません)が、キューにはアイテムがありません。'Paused'–キューは一時停止されているか、コマンドが現在キュー内で実行されている場合に一時停止のフラグが立てられます。'Stopped'–キューは停止状態にあるか、コマンドが現在キュー内で実行されている場合に停止するフラグが立てられています。

```
if (vaProxy.Queue.Status("myQueue") == "Running")
if(vaProxy.Queue.Status(" myQueue")==" Running")

{
{

//do something here
//ここで何かをする

}
}
```

CommandCount(string Name) – returns an int that is the number of commands contained within a command execution queue that is indicated in the Name parameter. If the queue does not exist, the return value will be 0.
CommandCount（string Name）– Nameパラメーターで示されるコマンド実行キュー内に含まれるコマンドの数であるintを返します。キューが存在しない場合、戻り値は0になります。

ActiveCommandName(string Name) – returns a string that is the name of the executing command of a command execution queue (indicated in the Name parameter). If the queue does not exist, or if there is no command currently executing within the queue, an empty value ‘’ will be returned.
ActiveCommandName（string Name）–コマンド実行キューの実行コマンドの名前である文字列を返します（Nameパラメーターで指定）。キューが存在しない場合、またはキュー内で現在実行中のコマンドがない場合、空の値 ''が返されます。

vaProxy.State Attributes
vaProxy.Stateの属性

Below are the attributes that belong to the State object of vaProxy. This set of functions deals with various states of devices, your computer, and VoiceAttack itself. Just like the, ‘Utility’ object earlier in this section, this is a bunch of functions that VoiceAttack uses internally, but is exposed to hopefully make things a bit easier for you for certain stuff.
以下は、vaProxyのStateオブジェクトに属する属性です。この一連の機能は、デバイス、コンピューター、およびVoiceAttack自体のさまざまな状態を処理します。このセクションで前述した「Utility」オブジェクトと同様に、これはVoiceAttackが内部的に使用する一連の関数ですが、特定のものについては少し簡単にするために公開されています。

GetListeningEnabled() – This Boolean function returns true if VoiceAttack’s, ‘listening’ is turned on, false if it is not.
GetListeningEnabled（）–このブール関数は、VoiceAttackの 'listening'がオンになっている場合はtrueを返し、そうでない場合はfalseを返します。

SetListeningEnabled(bool Value) – This function turns VoiceAttack’s, ‘listening’ on and off. Passing a true value turns, ‘listening’ on, while passing a false value turns, ‘listening’ off.
SetListeningEnabled（bool Value）–この関数は、VoiceAttackの「リスニング」のオンとオフを切り替えます。真の値を渡すと「リスニング」がオンになり、偽の値を渡すと「リスニング」がオフになります。

GetShortcutsEnabled() – This Boolean function returns true if VoiceAttack’s keyboard shortcut keys are enabled.
GetShortcutsEnabled（）–このブール関数は、VoiceAttackのキーボードショートカットキーが有効になっている場合にtrueを返します。

SetShortcutsEnabled(bool Value) – This function turns VoiceAttack’s keyboard shortcuts on and off. Passing a true value turns the shortcuts on, while passing a false value turns them off.
SetShortcutsEnabled（bool Value）–この関数は、VoiceAttackのキーボードショートカットのオンとオフを切り替えます。true値を渡すとショートカットがオンになり、false値を渡すとオフになります。

GetJoystickButtonsEnabled() - This Boolean function returns true if VoiceAttack's joystick shortcut buttons are enabled.
GetJoystickButtonsEnabled()-VoiceAttackのジョイスティックショートカットボタンが有効な場合、このブール関数はtrueを返します。

SetJoystickButtonsEnabled(bool Value) - This function turns VoiceAttack's joystick shortcuts on and off. Passing a true value turns the shortcuts on, while passing a false value turns them off.
SetJoystickButtonsEnabled(bool Value)-この関数は、VoiceAttackのジョイスティックショートカットをオンまたはオフにします。true値を渡すとショートカットがオンになり、false値を渡すとオフになります。

GetMouseButtonsEnabled() - This Boolean function returns true if VoiceAttack's mouse shortcut buttons are enabled.
GetMouseButtonsEnabled()-このブール関数は、VoiceAttackのマウスショートカットボタンが有効になっている場合にtrueを返します。

SetMouseButtonsEnabled(bool Value) - This function turns VoiceAttack's mouse shortcuts on and off. Passing a true value turns the shortcuts on, while passing a false value turns them off.
SetMouseButtonsEnabled(bool Value)-この関数は、VoiceAttackのマウスショートカットのオンとオフを切り替えます。true値を渡すとショートカットがオンになり、false値を渡すとオフになります。

KeyDown(String Key) - returns a Boolean value of true if the indicated keyboard key is pressed down. The, 'Key' parameter can be any key you can type in a token: 'A', 'B', 'C', 'ß', 'ö', 'ñ', 'ç', as well as keys you can't type in: ENTER, TAB, LCTRL, ARROWR (see section later in this document titled, 'Key State Token Parameter Values' for the full list). For example, if you want to test to see if the F10 key is down, just call KeyDown("F10"). To test for the letter 'A', just call KeyDown("A").
KeyDown(String Key)-指定されたキーボードキーが押されると、ブール値trueを返します。「Key」パラメーターには、トークンに入力できる任意のキーを指定できます。「A」、「B」、「C」、「ß」、「ö」、「ñ」、「ç」、およびキーENTER、TAB、LCTRL、ARROWRを入力することはできません(完全なリストについては、このドキュメントの「キー状態トークンパラメータ値」というタイトルのセクションを参照してください)。たとえば、F10キーが押されているかどうかをテストするには、KeyDown(" F10")を呼び出します。文字「A」をテストするには、KeyDown(「A」)を呼び出すだけです。

AnyKeyDown() - returns a Boolean value of true if any keyboard key is currently in the pressed down state, and returns false if no keys are pressed down.
AnyKeyDown()-キーボードのキーが現在押されている状態の場合はブール値trueを返し、押されているキーがない場合はfalseを返します。

MouseLeftButtonDown() MouseRightButtonDown() MouseMiddleButtonDown() MouseForwardButtonDown()
MouseLeftButtonDown()MouseRightButtonDown()MouseMiddleButtonDown()MouseForwardButtonDown()

MouseBackButtonDown() – Each of these functions test to see if a mouse button is being pressed. For example, if you want to test for the right mouse button, use the function MouseRightButtonDown().. If the mouse button is pressed down, the returned value will be true, otherwise it will be false.
MouseBackButtonDown()–これらの各関数は、マウスボタンが押されているかどうかをテストします。たとえば、マウスの右ボタンをテストする場合は、MouseRightButtonDown()関数を使用します。マウスボタンが押されると、戻り値はtrueになり、そうでない場合はfalseになります。

AnyMouseButtonDown() – This Boolean function checks to see if any of the five standard mouse buttons are currently pressed. If any mouse button is down, the returned value will be true. If no buttons are pressed, false will be returned.
AnyMouseButtonDown()–このブール関数は、5つの標準マウスボタンのいずれかが現在押されているかどうかを確認します。マウスボタンが押されている場合、返される値はtrueになります。ボタンが押されていない場合、falseが返されます。

CPU(int Core)
CPU(int Core)

CPU() – These will return your cpu usage as an integer. CPU() will return the average for all cores. The value returned will be from 0 to 100. CPU(int Core) will allow you to specify a particular core. For instance, CPU(5) will get the cpu usage for core 5.
CPU()–CPU使用量を整数として返します。CPU()は、すべてのコアの平均を返します。返される値は0～100です。CPU(int Core)を使用すると、特定のコアを指定できます。たとえば、CPU(5)はコア5のCPU使用率を取得します。

RAMTotal() – This will return the total RAM on your system in bytes as an unsigned long (ulong).
RAMTotal()–これは、システム上の合計RAMを符号なしlong(ulong)としてバイト単位で返します。

RAMAvailable() – This will return the available RAM on your system in bytes as an unsigned long (ulong).
RAMAvailable()–これは、システムで利用可能なRAMを、符号なしlong(ulong)としてバイト単位で返します。

FileExists(String Path) – This Boolean function will return true if the file indicated in Path exists, or false if it does not.
FileExists(String Path)–このブール関数は、Pathに示されたファイルが存在する場合はtrueを返し、存在しない場合はfalseを返します。

DirectoryExists(String Path) – This Boolean function will return true if the directory indicated in Path exists, or false if it does not.
DirectoryExists(String Path)–このブール関数は、Pathで指定されたディレクトリが存在する場合はtrueを返し、存在しない場合はfalseを返します。

DirectoryHasFiles(String Path) – This Boolean function will return true if the directory indicated in Path has files in it, or false if it does not. Note that if the directory does not exist, false will also be returned.
DirectoryHasFiles（String Path）–このブール関数は、Pathで指定されたディレクトリにファイルが含まれている場合はtrueを、含まれていない場合はfalseを返します。ディレクトリが存在しない場合、falseも返されることに注意してください。

AudioLevel() – This function indicates the currently reported audio level from the speech engine as an integer. The returned value will be from 0 to 100.
AudioLevel（）–この関数は、音声エンジンから現在報告されている音声レベルを整数として示します。返される値は0～100です。

AudioLastFile() – This will return the path of the last audio file that is played as a string.
AudioLastFile（）–これは、文字列として再生される最後のオーディオファイルのパスを返します。

AudioCount() – This returns the number of all currently-playing audio files as an integer. If legacy audio mode is on this value will always be zero.
AudioCount（）–現在再生中のすべてのオーディオファイルの数を整数として返します。レガシーオーディオモードがオンの場合、この値は常にゼロになります。

AudioCount(String Path) – This returns the number of currently-playing instances of an audio file with a given file path as an integer.
AudioCount（String Path）–これは、整数として指定されたファイルパスを持つオーディオファイルの現在再生中のインスタンスの数を返します。

Notes: Since sounds run asynchronously in VoiceAttack, there is a slight chance that if you use this function IMMEDIATELY after executing a 'Play Sound' action the file may not yet have had a chance to queue or load up and will not be included in the count. This is technically correct, but may not be a proper count depending on what you are trying to accomplish.
注：音がVoiceAttackで非同期に実行されるため、わずかなチャンスがあることを、あなたは、この機能を使用する場合は、すぐに実行した後、ファイルがまだキューまたはロードアップする機会を持っていない可能性があり、中には含まれませんアクション「サウンド再生」をカウント。これは技術的には正しいですが、達成しようとしている内容によっては適切なカウントではない場合があります。

If legacy audio mode is on this function will always return zero.
レガシーオーディオモードがオンの場合、この関数は常にゼロを返します。

AudioPos(String Path) – This returns the position of currently-playing audio file with a given file path, expressed in seconds as an integer value.
AudioPos（String Path）–これは、整数値として秒単位で表される、指定されたファイルパスを持つ現在再生中のオーディオファイルの位置を返します。

Notes: Since sounds run asynchronously in VoiceAttack, there is a slight chance that if you use this token IMMEDIATELY after executing a ‘Play Sound’ action the file may not yet have had a chance to queue or load up and will return a position of 0. This is technically correct, but may not be a proper value depending on what you are trying to accomplish.
注：音がVoiceAttackで非同期に実行されるため、わずかなチャンスがあることを、あなたはこのトークンを使用する場合は、すぐに実行した後、ファイルがまだキューまたはロードし、0の位置を返しますする機会を持っていないかもしれないアクション「サウンド再生」を。これは技術的には正しいですが、達成しようとしている内容によっては適切な値ではない場合があります。

Since you can run multiple instances of a sound file at once, if there is more than one instance currently playing, the rendered value will be zero, (assumptions cannot be made
サウンドファイルの複数のインスタンスを一度に実行できるため、現在再生中のインスタンスが複数ある場合、レンダリングされた値はゼロになります（仮定はできません

on which instance to be chosen). To help with this, check the AudioCount() function outlined above prior to using the AudioPos() function.
どのインスタンスを選択するか）。これを支援するには、AudioPos（）関数を使用する前に、上記のAudioCount（）関数を確認してください。

If no instances of the indicated file are currently playing, zero will be returned. If legacy audio mode is on this function will always return zero.
指定されたファイルのインスタンスが現在再生されていない場合、ゼロが返されます。レガシーオーディオモードがオンの場合、この関数は常にゼロを返します。

DefaultPlayback() – This returns the device name of the default multimedia audio playback device as indicated by Windows as a string. Note that there is a very minor memory leak when accessing the multimedia device property store, so this will be reflected
DefaultPlayback（）–これは、Windowsが文字列として示すデフォルトのマルチメディアオーディオ再生デバイスのデバイス名を返します。マルチメディアデバイスプロパティストアにアクセスするときに非常に小さなメモリリークがあるため、これが反映されることに注意してください。

in VoiceAttack (using this function sparingly will not present a problem… running it over and over in a loop will chew up memory… the search for a better way continues).
VoiceAttackで（この関数を控えめに使用しても問題はありません。ループ内で繰り返し実行するとメモリが消費されます...より良い方法の検索が続行されます）。

DefaultPlaybackComms() – This works just like DefaultPlayback() (above), except the default communications playback device name will be returned.
DefaultPlaybackComms（）–これはDefaultPlayback（）（上記）と同様に機能しますが、デフォルトの通信再生デバイス名が返されます。

DefaultRecording() – This works just like DefaultPlayback() (above), except the default multimedia recording device name will be returned.
DefaultRecording（）–これはDefaultPlayback（）（上記）と同様に機能しますが、デフォルトのマルチメディア記録デバイス名が返されます。

DefaultRecordingComms() – This works just like DefaultPlayback() (above), except the default communications recording device name will be returned.
DefaultRecordingComms()–これはDefaultPlayback()（上記）と同様に機能しますが、デフォルトの通信記録デバイス名が返されます。

SysDir() – This returns the path of the system directory (e.g. 'C:¥Windows¥System32').
SysDir()–システムディレクトリのパスを返します（例： 'C：¥ Windows ¥ System32'）。

WinDir() – This returns the path of the Windows directory (e.g. 'C:¥Windows').
WinDir()–これは、Windowsディレクトリのパスを返します（例： 'C：¥ Windows'）。

EnvironmentVariable(String Value) – This renders the Windows environment variable specified in, 'Value' as a string. For example, EnvironmentVariable("programfiles") would (usually) return 'C:¥Program Files'.
EnvironmentVariable（String Value）–'Value'で指定されたWindows環境変数を文字列としてレンダリングします。たとえば、EnvironmentVariable（" programfiles"）は（通常） 'C：¥ Program Files'を返します。

Culture() – This returns the default user locale of the system as a string.
Culture()–これは、システムのデフォルトのユーザーロケールを文字列として返します。

UICulture() – This returns the default user interface language as a string.
UICulture()–これは、デフォルトのユーザーインターフェイス言語を文字列として返します。

SysVol() – This returns the system volume (default playback device) as an integer value from 0 to 100.
SysVol()–システムボリューム（デフォルトの再生デバイス）を0～100の整数値として返します。

SysMute() – If the system volume (default playback device) is muted, this function returns true. If not, the value returned is false.
SysMute()–システムボリューム（デフォルトの再生デバイス）がミュートされている場合、この関数はtrueを返します。そうでない場合、返される値はfalseです。

MicVol() – This returns the microphone volume (default recording device) as an integer value from 0 to 100.
MicVol()–マイクの音量（デフォルトの録音デバイス）を0～100の整数値として返します。

MicMute() – If the microphone volume (default recording device) is muted, this function returns true. If not, the value is returned is false.
MicMute()–マイクの音量(デフォルトの録音デバイス)がミュートされている場合、この関数はtrueを返します。そうでない場合、返される値はfalseです。

AppVol(String Context) – This function returns an integer value between 0 and 100 based on the indicated app volume as it relates to the System Volume Mixer. If the volume
AppVol(String Context) –この関数は、System Volume Mixerに関連する指定されたアプリボリュームに基づいて、0～100の整数値を返します。ボリュームが

cannot be accessed (app closed or no audio is playing for instance), a value of –1 will be returned. The Context parameter for this function should be the window title, process name or window class name of the target application (see "Set Audio Level" section of the, "Other Stuff" screen for more information on targeting the application).
アクセスできない(アプリを閉じる、またはオーディオが再生されていないなど)場合、–1の値が返されます。この関数のContextパラメーターは、ターゲットアプリケーションのウィンドウタイトル、プロセス名、またはウィンドウクラス名である必要があります(アプリケーションのターゲット設定の詳細については、「その他の設定」画面の「オーディオレベルの設定」セクションを参照してください)。

AppMute(String Context) – This works exactly the same as the AppVol() function above, except this returns a value of true if the indicated application is muted as it relates to the System Volume Mixer, and false if the application is not muted. False is returned if the information cannot be accessed.
AppMute(String Context)– これは、上記のAppVol()関数とまったく同じように機能しますが、指定されたアプリケーションがSystem Volume Mixerに関連してミュートされている場合はtrue、アプリケーションがミュートされていない場合はfalseを返します。情報にアクセスできない場合は、Falseが返されます。

SpeechDeviceMute() – This function tests if the recording device that the speech engine is currently using is muted. If the device is muted, a Boolean value of true will be returned.
SpeechDeviceMute()–この関数は、音声エンジンが現在使用している録音デバイスがミュートされているかどうかをテストします。デバイスがミュートされている場合、ブール値trueが返されます。

Otherwise, the returned value will be false.
それ以外の場合、戻り値はfalseになります。

SpeechDeviceVol() – This returns the volume of the recording device that the speech engine is currently using as an integer value from 0 to 100.
SpeechDeviceVol()–これは、スピーチエンジンが現在使用している録音デバイスのボリュームを0～100の整数値として返します。

SpeechActive() – This function tests to see if the speech engine is detecting speech. If speech is currently detected, the returned value will be true. If, not, the returned value will be false.
SpeechActive()–この関数は、音声エンジンが音声を検出しているかどうかをテストします。音声が現在検出されている場合、戻り値はtrueになります。そうでない場合、戻り値はfalseになります。

State.Joystick – The Joystick object within State holds a collection of methods that are a direct port of all the token functions outlined in the token function section of this help document. Since the audience for these methods will most likely be extremely small, and for brevity (and to hopefully save a tree), the full documentation on these methods has been omitted. Please refer to the section titled, 'Joystick State Token Reference' – the methods are pretty much a 1 to 1 match. If you have questions or need clarification on this object, please come by the VoiceAttack Discord server or the VoiceAttack user forum. If it turns out that these methods are in high demand, proper documentation will be added here.
State.Joystick – State内のJoystickオブジェクトは、このヘルプドキュメントのトークン関数セクションで概説されているすべてのトークン関数の直接ポートであるメソッドのコレクションを保持します。これらのメソッドの対象者は非常に少なく、簡潔にするため(そしてツリーを保存するため)、これらのメソッドの完全なドキュメントは省略されています。「Joystick State Token Reference」というタイトルのセクションを参照してください–メソッドはほぼ1対1の一致です。このオブジェクトについて質問がある場合、または説明が必要な場合は、VoiceAttack Discordサーバーまたは VoiceAttackユーザーフォーラムにアクセスしてください。これらの方法の需要が高いことが判明した場合は、適切なドキュメントをここに追加します。

vaProxy Events
vaProxyイベント

ProfileChanging(Guid? FromInternalID, Guid? ToInternalID, String FromName, String ToName) – this event is raised when a profile is about to be changed. As a helper, the InternalID and name of both the current profile and the profile that is about to be loaded are included as parameters. Note that the usage for inline functions and plugins varies slightly.
ProfileChanging(Guid？FromInternalID、Guid？ToInternalID、String FromName、String ToName) –このイベントは、プロファイルが変更されようとしているときに発生します。ヘルパーとして、現在のプロファイルとロードされようとしているプロファイルの両方のInternalIDと名前がパラメーターとして含まれています。インライン関数とプラグインの使用法はわずかに異なることに注意してください。

Example for inline function usage:
インライン関数の使用例:

```
public void main()
public void main()

{
{

VA.ProfileChanging += MyProfileChangingAction;
VA.ProfileChanging + = MyProfileChangingAction;

}
}
```

public void MyProfileChangingAction(Guid? currentID, Guid? loadingID,
public void MyProfileChangingAction（Guid？currentID、Guid？loadingID、

String currentName, String loadingName)
文字列currentName、文字列loadingName）

{
{

VA.WriteToLog("Profile changing to " + loadingName, "orange");
VA.WriteToLog（"プロファイル変更" + loadingName 、 "orange" ）;

}
}

Example for plugin usage:
プラグインの使用例：

private static dynamic _proxy; //note - keeping a reference to the proxy object public static void VA_Init1(dynamic vaProxy)
private static dynamic _proxy; //注意-プロキシオブジェクトへの参照を保持するpublic static void VA_Init1（dynamic vaProxy）

{
{

_proxy = vaProxy;
_proxy = vaProxy;

_proxy.ProfileChanging += new Action<Guid?, Guid?, String, String>(MyPro fileChangingAction);
_proxy.ProfileChanging + = new Action <Guid ?, Guid ?, String、String>（MyPro fileChangingAction）;

}
}

public static void MyProfileChangingAction(Guid? FromInternalID, Guid? ToInternalID,
public static void MyProfileChangingAction（Guid？FromInternalID、Guid？ToInternalID、

String FromName, String ToName)
文字列FromName、文字列ToName）

```
{
{


_proxy.WriteToLog("Profile changing to " + ToName, "orange");
_proxy.WriteToLog("プロファイル変更" + ToName、"orange" );


}
}
```

ProfileChanged(Guid? FromInternalID, Guid? ToInternalID, String FromName, String ToName) – this event is raised when a profile has changed. As a helper, the InternalID and name of both the previous profile and the current profile are included as parameters. Note that the usage for inline functions and plugins varies slightly.
ProfileChanged(Guid？FromInternalID、Guid？ToInternalID、String FromName、String ToName) –このイベントは、プロファイルが変更されたときに発生します。ヘルパーとして、以前のプロファイルと現在のプロファイルの両方のInternalIDと名前がパラメーターとして含まれています。インライン関数とプラグインの使用法はわずかに異なることに注意してください。

Example for inline function usage:
インライン関数の使用例：

```
public void main()
public void main()


{
{


VA.ProfileChanged += MyProfileChangedAction;
VA.ProfileChanged + = MyProfileChangedAction;


}
}
```

```
public void MyProfileChangedAction(Guid? previousID, Guid? currentID
public void MyProfileChangedAction(Guid？previousID、Guid？currentID


String previosName, String currentName)
文字列previosName、文字列currentName)
```

```
{
{


VA.WriteToLog("Profile changed to " + currentName, "orange");
VA.WriteToLog("プロファイルが" + currentName 、 "orange"に変更されました);


}
}
```

Example for plugin usage:
プラグインの使用例：

```
private static dynamic _proxy; //note - keeping a reference to the proxy object
private static dynamic _proxy; //注意-プロキシオブジェクトへの参照を保持する



public static void VA_Init1(dynamic vaProxy)
public static void VA_Init1（dynamic vaProxy）


{
{


_proxy = vaProxy;
_proxy = vaProxy;


_proxy.ProfileChanged += new Action<Guid?, Guid?, String, String>(MyPro fileChangedAction);
_proxy.ProfileChanged + = new Action <Guid ?, Guid ?, String、String>（MyPro fileChangedAction）;


}
}
```

```
public static void MyProfileChangedAction(Guid? FromInternalID, Guid? ToInternalID,
public static void MyProfileChangedAction（Guid？FromInternalID、Guid？ToInternalID、


String FromName, String ToName)
文字列FromName、文字列ToName）
```

{
{

_proxy.WriteToLog("Profile changed to " + ToName, "orange");
_proxy.WriteToLog("プロファイルが" + ToName、"orange"に変更されました);

}
}

CommandsStopped – This event is raised when a, 'Stop All Commands' action is executed or the user presses the, 'Stop Commands' button on the main screen. Use this to receive notification that this event has occurred so you can clean up and exit your inline function (this can replace the Stopped property and ResetStopFlag() function indicated earlier). Note that if this event is used with the, 'Wait for the inline function to complete' option of an inline function, a ten-second grace period is provided in order to allow the inline function to clean up and exit properly before the thread is terminated. Although available and usable within a plugin, this is intended for use within inline functions (due to its nature, VoiceAttack removes this event when commands are stopped). For plugins, use, 'VA_StopCommand' (recommended).
CommandsStopped –このイベントは、「すべてのコマンドを停止」アクションが実行されるか、ユーザーがメイン画面の「コマンドを停止」ボタンを押すと発生します。これを使用して、このイベントが発生したという通知を受け取り、インライン関数をクリーンアップして終了します(これにより、前述のStoppedプロパティとResetStopFlag()関数を置き換えることができます)。このイベントがインライン関数の「インライン関数の完了を待つ」オプションで使用される場合、スレッドが実行される前にインライン関数がクリーンアップして適切に終了できるように、10秒間の猶予期間が提供されることに注意してください。終了しました。プラグイン内で使用可能で使用可能ですが、これはインライン関数内で使用することを目的としています(その性質により、VoiceAttackはコマンドが停止するとこのイベントを削除します)。プラグインの場合、「VA_StopCommand」を使用します(推奨)。

Example for inline function usage:
インライン関数の使用例:

public void main()
public void main()

{
{

VA.CommandsStopped += MyCommandsStoppedAction;
VA.CommandsStopped + = MyCommandsStoppedAction;

}
}

public void MyCommandsStoppedAction()
public void MyCommandsStoppedAction()

```
{
{


VA.WriteToLog("CommandsStopped Handled", "orange");
VA.WriteToLog("CommandsStopped Handled" 、"orange" );


}
}
```

ApplicationFocusChanged(System.Diagnostics.Process Process, String TopmostWindowTitle) – this event is raised when application focus is changed in Windows. For instance, if you switch from Notepad to Wordpad as your focused application, this event will be invoked. The event receives two parameters – the Process that has been focused, and the Window title of that process (as a convenience). This works in conjunction with the Automatic Profile Switching feature within VoiceAttack, so, 'Automatic Profile Switching' must be turned on from the Options screen in order for this event to fire. Use the, 'AutoProfileSwitchingEnabled' property to test whether or not automatic profile switching is turned on. Note: The polling frequency for application changes is set to 250ms, and the event is fired asynchronously from within VoiceAttack. Note that the usage for inline functions and plugins varies slightly.
ApplicationFocusChanged(System.Diagnostics.Process Process、String TopmostWindowTitle) –このイベントは、Windowsでアプリケーションフォーカスが変更されたときに発生します。たとえば、フォーカスしたアプリケーションとしてメモ帳からワードパッドに切り替えると、このイベントが呼び出されます。イベントは2つのパラメーターを受け取ります–フォーカスされたプロセスと、そのプロセスのウィンドウタイトル(便宜上)。これは、VoiceAttack内の自動プロファイル切り替え機能と連携して動作するため、このイベントを起動するには、オプション画面から「自動プロファイル切り替え」をオンにする必要があります。'AutoProfileSwitchingEnabled'プロパティを使用して、自動プロファイル切り替えがオンになっているかどうかをテストします。注:アプリケーション変更のポーリング頻度は250ミリ秒に設定され、イベントはVoiceAttack内から非同期で発生します。インライン関数とプラグインの使用法はわずかに異なることに注意してください。

Example for inline function usage:
インライン関数の使用例:

```
public void main()
public void main()


{
{


VA.ApplicationFocusChanged += MyApplicationFocusChangedAction;
VA.ApplicationFocusChanged + = MyApplicationFocusChangedAction;


}
}
```

public void MyApplicationFocusChangedAction(Process pFocus, String title)
public void MyApplicationFocusChangedAction(Process pFocus、String title)

{
{

VA.WriteToLog(title + " – " + pFocus.ProcessName , "orange");
VA.WriteToLog(title + "–" + pFocus.ProcessName 、"orange");

}
}

Example for plugin usage:
プラグインの使用例:

private static dynamic _proxy; //note – keeping a reference to the proxy object public static void VA_Init1(dynamic vaProxy)
private static dynamic _proxy; //注意–プロキシオブジェクトへの参照を保持するpublic static void VA_Init1(dynamic vaProxy)

{
{

_proxy = vaProxy;
_proxy = vaProxy;

_proxy.ApplicationFocusChanged += new Action<System.Diagnostics.Process, String>(MyApplicationFocusChangedAction);
_proxy.ApplicationFocusChanged + = new Action <System.Diagnostics.Process、String>(MyApplicationFocusChangedAction);

}
}

public static void MyApplicationFocusChangedAction(System.Diagnostics.Process pFocus, String WindowTitle)
public static void MyApplicationFocusChangedAction(System.Diagnostics.Process pFocus、String WindowTitle)

{
{

_proxy.WriteToLog(WindowTitle + " – " + pFocus.ProcessName, "orange");
_proxy.WriteToLog(WindowTitle + "–" + pFocus.ProcessName、"orange");

}
}

Remember… if you get stuck, drop by the VoiceAttack user forums and go to the plugin boards. Help is always around ;)
覚えておいてください...行き詰まった場合は、VoiceAttackユーザーフォーラムにアクセスして、プラグインボードにアクセスしてください。ヘルプは常にあります;)

Notes on testing your plugin
プラグインのテストに関する注意

Something I have done to make my plugin test development a little easier was to set up Visual Studio to output the built .dll straight into a subdirectory of the VoiceAttack Apps directory. I thought that was worth mentioning even though you probably already know to do this.
プラグインのテスト開発を少し簡単にするために、ビルドした.dllをVoiceAttack Appsディレクトリのサブディレクトリに直接出力するようにVisual Studioをセットアップすることでした。おそらくこれを行うことを既に知っているとしても、言及する価値があると思いました。

You can output your values using the, 'WriteToLog' function with the vaProxy object: vaProxy.WriteToLog("some value", "orange"); The output is token-parsed so you can use any or all of the tokens in there ({SMALL:}, {INT:}, {BOOL:}, {DEC:}, various {DATE:} and
vaProxyオブジェクトで「WriteToLog」関数を使用して値を出力できます。vaProxy.WriteToLog(" some value"、" orange"); 出力はトークン解析されるため、そこにあるトークンの一部またはすべてを使用できます({SMALL:}、{INT:}、{BOOL:}、{DEC:}、さまざまな{DATE:}および

{TXT:}).
{TXT:})。

Now that the dynamic vaProxy is in place and generic parameters are not being passed, there will no longer be a test harness project included with the samples.
動的vaProxyが配置され、汎用パラメーターが渡されていないため、サンプルに含まれるテストハーネスプロジェクトはなくなります。

Notes on referenced assemblies
参照アセンブリに関する注意

A lot of times your plugin will want to reference assemblies (.dlls) that are not part of the
多くの場合、プラグインは、の一部ではないアセンブリ(.dll)を参照します。

.Net framework or are not installed in the Global Assembly Cache (GAC). Your plugin will only work if VoiceAttack can actually locate and use your referenced assemblies. In order for VoiceAttack to locate your referenced assemblies, the assemblies must be located in the GAC (of course), in the same folder as your plugin, in the VoiceAttack installation directory (where the VoiceAttack.exe resides), or in the Shared¥Assemblies folder located in the VoiceAttack installation directory.
.Net frameworkまたはGlobal Assembly Cache（GAC）にインストールされていません。プラグインは、VoiceAttackが参照アセンブリを実際に見つけて使用できる場合にのみ機能します。VoiceAttackが参照アセンブリを見つけるには、GAC（もちろん）、プラグインと同じフォルダー、VoiceAttackインストールディレクトリ（VoiceAttack.exeが存在する）、またはShared ¥にアセンブリを配置する必要があります。 VoiceAttackインストールディレクトリにあるAssembliesフォルダー。

Version 3 and below plugin parameter notes (this is old stuff but still left here as a reference. Note that there have been some updates to this interface to allow access to vaProxy if you still require this interface).
バージョン3以下のプラグインパラメーターのメモ（これは古いものですが、参考としてここに残されています。このインターフェイスが必要な場合、vaProxyへのアクセスを許可するために、このインターフェイスにいくつかの更新があります）

State parameter
状態パラメーター

The state parameter that is passed in to VA_Invoke1, VA_Exit1 and VA_Init1 is for your own private use within this plugin. No other plugin has access to the values. It serves as a kind of session, so that you can easily maintain information between calls without having to do any kind of persistence. The dictionary is (String, Object) so you can name your values whatever you want, as well as store any kind of value type. Whatever you do to this dictionary will be reflected in VoiceAttack upon return (VA_Invoke1 and VA_Init1 only). So, if you empty this dictionary, VoiceAttack's copy of this dictionary will be emptied. Null values will not be cleaned out. Note: When the state dictionary is initialized for a plugin, there are three key/value pairs that are included for use. The keys are below (the key names are the same as the token values used elsewhere):
VA_Invoke1、VA_Exit1、およびVA_Init1に渡される状態パラメーターは、このプラグイン内で独自に使用するためのものです。他のプラグインは値にアクセスできません。これは一種のセッションとして機能するため、コール間で情報を維持するために、いかなる種類の永続性も必要ありません。ディクショナリは（String、Object）であるため、任意の値に名前を付けたり、任意の種類の値タイプを保存したりできます。このディクショナリに対して行うことは、返されるとVoiceAttackに反映されます（VA_Invoke1およびVA_Init1のみ）。したがって、この辞書を空にすると、この辞書のVoiceAttackのコピーは空になります。null値は消去されません。注：状態辞書がプラグイン用に初期化されると、使用するために含まれるキー/値のペアが3つあります。

VA_DIR : The installation directory of VoiceAttack. VA_APPS : VoiceAttack apps/plugins directory.
VA_DIR：VoiceAttackのインストールディレクトリ。VA_APPS：VoiceAttack apps / pluginsディレクトリ。

VA_SOUNDS : VoiceAttack sounds directory.
VA_SOUNDS：VoiceAttackはディレクトリを鳴らします。

VA_SOUNDS : To allow plugin version v3 and prior to allow access to the new vaProxy object, the VA_PROXY key was added. You can access the VA_PROXY object like so: dynamic vaProxy = state["VA_PROXY"];
VA_SOUNDS：プラグインバージョンv3を許可し、新しいvaProxyオブジェクトへのアクセスを許可する前に、VA_PROXYキーが追加されました。あなたはそのようVA_PROXYオブジェクトにアクセスすることができますダイナミック vaProxy =状態[「VA_PROXY」];

See notes above on how to use vaProxy. Note also that VoiceAttack v1.6 or greater will be needed to be able to use this feature.
vaProxyの使用方法については、上記のメモを参照してください。また、この機能を使用するにはVoiceAttack v1.6以上が必要になることに注意してください。

Again, these values can be erased and manipulated however you want.
繰り返しますが、これらの値は必要に応じて消去および操作できます。

SmallIntegerValues (formerly, 'Conditions') parameter
SmallIntegerValues（以前の「Conditions」）パラメーター

The small integers parameter is a dictionary of (String, nullable Int16 (or short)). These are the same small integers (conditions) you find when using them within the VoiceAttack interface.
small integersパラメータは、(String、nullable Int16（またはshort）)の辞書です。これらは、VoiceAttackインターフェース内で使用するときに見つかる同じ小さな整数（条件）です。

To get values from VoiceAttack into the plugin in VA_Invoke1, simply indicate what small integers to pass in the, 'Small Integer Variables' box in the, 'Execute an External Plugin Function' screen. For example, if you want to pass in, 'myValue1' and 'myValue5', just put 'myValue1;myValue5' (no quotes) in the box. VoiceAttack will copy the values into a new list and pass them in to the plugin for you to use. Inside the plugin's function (either VA_Invoke1 or VA_Init1), you can read 'myValue1' or 'myValue5', alter their values (set to null to remove) and/or add more values to the small integers dictionary. Any changes will be reflected in VoiceAttack upon return. To indicate to VoiceAttack that you want to remove a small integer, simply set its value to null (smallIntegerValues[“myConditionName”] = null
VoiceAttackからVA_Invoke1のプラグインに値を取得するには、「外部プラグイン関数の実行」画面の「小さい整数変数」ボックスに渡す小さな整数を指定します。たとえば、「myValue1」および「myValue5」を渡す場合は、ボックスに「myValue1; myValue5」（引用符なし）を入力します。VoiceAttackは値を新しいリストにコピーし、使用するプラグインに渡します。プラグインの関数（VA_Invoke1またはVA_Init1のいずれか）内で、「myValue1」または「myValue5」を読み取り、それらの値を変更（削除するにはnullに設定）および/または小さな整数辞書に値を追加できます。変更は、戻ったときにVoiceAttackに反映されます。小さな整数を削除することをVoiceAttackに示すには、

). On a programming side-note... Since these are dictionaries, it is necessary to check to see if an item exists before doing anything to it... otherwise things come to a quick halt.
）。プログラミングに関する注意事項...これらは辞書であるため、何かを行う前に項目が存在するかどうかを確認する必要があります...そうでなければ、物事はすぐに停止します。

Small integers are public and can be accessed from any plugin or any command within the VoiceAttack user interface. That means that anybody can view or modify these values.
小さな整数はパブリックであり、VoiceAttackユーザーインターフェイス内の任意のプラグインまたはコマンドからアクセスできます。つまり、誰でもこれらの値を表示または変更できるということです。

You can access the values in conditions using the {SMALL:variableName}
{SMALL：variableName } を使用して条件の値にアクセスできます

({COND:conditionName} remains for backward compatibility) token in various places in VoiceAttack.
（{COND：conditionName }は下位互換性のために残されています）VoiceAttackのさまざまな場所のトークン。

TextValues parameter
TextValuesパラメーター

The textValues parameter is a dictionary of (String, String). It behaves pretty much exactly like the smallIntegerValues parameter, except the token to access a text value is
textValuesパラメーターは（String、String）の辞書です。これは、smallIntegerValuesパラメーターとほとんど同じように動作しますが、テキスト値にアクセスするトークンが

{TXT:textVariableName}, and you pass in values to your plugin via the Text Variables box in the command action.
{TXT：textVariableName }、コマンドアクションの[テキスト変数]ボックスを介してプラグインに値を渡します。

(and now for some copy/paste/replace… four times :))
（そして今、いくつかのコピー/貼り付け/置換のために... 4回:））

IntValues parameter
IntValuesパラメーター

The intValues parameter is a dictionary of (String, int). It behaves pretty much exactly like the smallIntegerValues parameter, except the token to access a text value is
intValuesパラメーターは、（String、int）の辞書です。これは、smallIntegerValuesパラメーターとほとんど同じように動作しますが、テキスト値にアクセスするトークンが

{INT:variableName}, and you pass in values to your plugin via the Integer Variables box in the command action.
{INT：variableName }、コマンドアクションの[整数変数]ボックスを介してプラグインに値を渡します。

DecimalValues parameter
DecimalValuesパラメーター

The decimalValues parameter is a dictionary of (String, decimal). It behaves pretty much exactly like the smallIntegerValues parameter, except the token to access a text value is
decimalValuesパラメータは（String、decimal）の辞書です。これは、smallIntegerValuesパラメーターとほとんど同じように動作しますが、テキスト値にアクセスするトークンが

{DEC:variableName}, and you pass in values to your plugin via the Decimal Variables box in the command action.
{DEC：variableName }、コマンドアクションの[10進変数]ボックスを介してプラグインに値を渡します。

BooleanValues parameter
BooleanValuesパラメーター

The booleanValues parameter is a dictionary of (String, Boolean). It behaves pretty much exactly like the smallIntegerValues parameter, except the token to access a text value is
booleanValuesパラメーターは（String、Boolean）の辞書です。これは、smallIntegerValuesパラメーターとほとんど同じように動作しますが、テキスト値にアクセスするトークンが

{BOOL:variableName}, and you pass in values to your plugin via the Boolean Variables box in the command action.
{BOOL：variableName }、コマンドアクションの[ブール変数]ボックスを介してプラグインに値を渡します。

DateTimeValues parameter
DateTimeValuesパラメーター

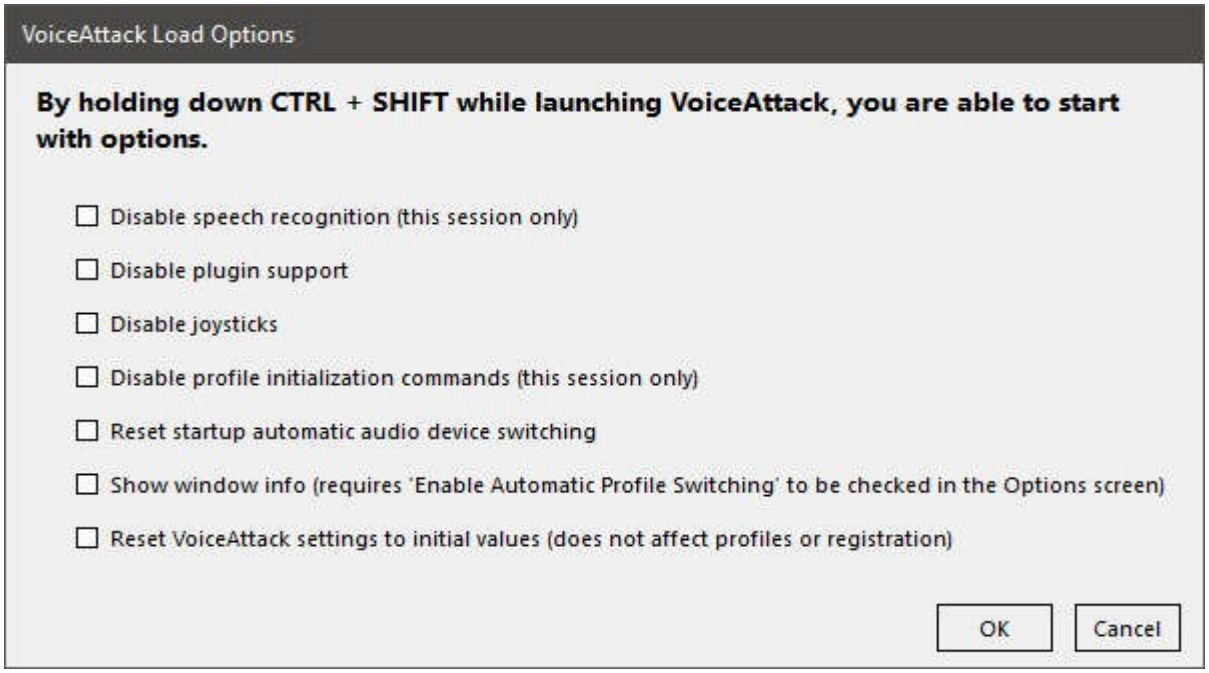
The dateTimeValues parameter is a dictionary of (String, DateTime). It behaves pretty much exactly like the smallIntegerValues parameter, except the token to access a text value is {DATE:variableName}, and you pass in values to your plugin via the Date/Time Variables box in the command action.
dateTimeValuesパラメーターは（String、DateTime）の辞書です。テキスト値にアクセスするトークンが{DATE：variableName }であり、コマンドアクションの[日付/時刻変数]ボックスを介してプラグインに値を渡すことを除いて、smallIntegerValuesパラメーターとほとんど同じように動作します。

Context parameter (VA_Invoke1 only)
コンテキストパラメータ（VA_Invoke1のみ）

This parameter is just a string that can be used to get a value into the plugin. This was added so that you don't have to go through the legwork of setting a condition or textValue just to get something simple passed. Convert the string value inside the plugin to whatever type you need to use. You will probably use this parameter the most.
このパラメーターは、プラグインに値を取得するために使用できる単なる文字列です。これは、単純なものを渡すためだけに条件またはtextValueを設定するという手間をかける必要がないように追加されました。プラグイン内の文字列値を、使用する必要のあるタイプに変換します。おそらく、このパラメーターを最もよく使用します。

VoiceAttack Load Options Screen
VoiceAttackロードオプション画面

Sometimes you may need to change a particular setting before VoiceAttack loads. If you hold down Control + Shift before and during VoiceAttack's launch, you will get the following screen:
VoiceAttackがロードされる前に特定の設定を変更する必要がある場合があります。VoiceAttackの起動前および起動中にCtrl + Shiftを押し続けると、次の画面が表示されます。



From here, you can disable the speech engine's recognition facilities. This is available in case your speech engine is not working and you're not able to get into VoiceAttack for some reason. Note this option is for the current session only and the next time you run VoiceAttack, the speech engine will be enabled (that is, unless you select the, 'Disable Speech Recognition' on the System/Advanced tab of the Options screen).
ここから、音声エンジンの認識機能を無効にできます。これは、音声エンジンが機能しておらず、何らかの理由でVoiceAttackにアクセスできない場合に利用できます。このオプションは現在のセッション専用であり、VoiceAttackを次に実行するときに音声エンジンが有効になります（つまり、[オプション]画面の[システム/詳細設定]タブで[音声認識を無効にする]を選択しない限り）。

You can disable plugin support (in case you get a rogue plugin that you don't want launched on startup). Setting this will disable plugin support until you enable it again in the Options screen.
プラグインのサポートを無効にすることができます（起動時に起動したくない不正なプラグインを取得した場合）。これを設定すると、オプション画面で再度有効にするまでプラグインのサポートが無効になります。

You can also disable joystick support. This will disable joystick support until you enable it again.
ジョイスティックのサポートを無効にすることもできます。これにより、再び有効にするまでジョイスティックのサポートが無効になります。

Disabling profile initialization will prevent any of the commands you have chosen to execute on profile change from actually executing. Note that this only occurs on the current running instance of VoiceAttack. If you run VoiceAttack again without this setting, the profile initialization commands will be enabled again.
プロファイルの初期化を無効にすると、プロファイルの変更時に実行することを選択したコマンドが実際に実行されなくなります。これはVoiceAttackの現在実行中のインスタンスでのみ発生することに注意してください。この設定なしでVoiceAttackを再度実行すると、プロファイル初期化コマンドが再び有効になります。

'Reset startup automatic audio device switching' will reset the options that allow for playback and recording devices to be automatically set on startup. This will help if VoiceAttack is experiencing device trouble accessing these devices.
「起動時の自動オーディオデバイス切り替えのリセット」は、起動時に再生デバイスと録音デバイスを自動的に設定できるオプションをリセットします。これは、VoiceAttackがこれらのデバイスへのアクセスに問題を抱えている場合に役立ちます。

Selecting, 'Show window info' will display the window title and process name from selected foreground windows. This is so you can easily see what VoiceAttack sees when deciding how to choose your target applications for automatic profile switching. Note that this only occurs on the current running instance of VoiceAttack.
[ウィンドウ情報を表示]を選択すると、選択したフォアグラウンドウィンドウのウィンドウタイトルとプロセス名が表示されます。これにより、自動プロファイル切り替え用のターゲットアプリケーションの選択方法を決定する際に、VoiceAttackが表示する内容を簡単に確認できます。これはVoiceAttackの現在実行中のインスタンスでのみ発生することに注意してください。

Also Note: This option will only work if the, 'Enable Automatic Profile Switching' option is turned on in the main Options screen.
注：このオプションは、メインのオプション画面で[プロファイルの自動切り替えを有効にする]オプションがオンになっている場合にのみ 機能します。

Checking the, 'Reset VoiceAttack settings to initial values' box will wipe out your VoiceAttack user settings (just like clicking the, 'Reset Defaults' button on the Options screen). Note that your profiles and registration information are preserved.
[VoiceAttackの設定を初期値にリセットする]ボックスをオンにすると、VoiceAttackのユーザー設定が消去されます（[オプション]画面の[デフォルトにリセット]ボタンをクリックするのと同じです）。プロファイルと登録情報は保持されることに注意してください。

Once you have made your selections, just click, 'OK' and VoiceAttack will continue to load.
選択したら、[OK]をクリックするだけで、VoiceAttackの読み込みが続行されます。

NOTE: This screen will automatically appear if VoiceAttack is not shut down properly (probably due to a crash). You can choose to not automatically show this screen from the dialog that pops up prior to the Load Options screen being displayed.
注：VoiceAttackが適切にシャットダウンされない場合（おそらくクラッシュが原因）、この画面は自動的に表示されます。[ロードオプション]画面が表示される前にポップアップするダイアログから、この画面を自動的に表示しないように選択できます。

Advanced Variable Control (Scope)
高度な変数制御（スコープ）

This section is to provide a little bit of guidance on variable, 'scope' in VoiceAttack. Initially, all variables in VoiceAttack were globally-scoped. That means that when a variable is set, it is available to be read from and written to from any command in any profile. For the most part, this system works well and will continue to be used. The only gotcha is that global variables must be named very uniquely in order to not interfere with each other. For example:
このセクションでは、VoiceAttackの変数「スコープ」に関するガイダンスを少し提供します。当初、VoiceAttackのすべての変数はグローバルスコープでした。つまり、変数が設定されると、どのプロファイルのどのコマンドからでも読み取りおよび書き込みが可能になります。ほとんどの場合、このシステムはうまく機能し、引き続き使用されます。唯一の落とし穴は、互いに干渉しないようにグローバル変数に非常に一意の名前を付ける必要があることです。例えば：

You have two profiles: Profile A and Profile B. Profile A has a command that creates a text variable, 'myTextVariable'. All of Profile A's commands can see and modify, 'myTextVariable'. Also, when you switch to Profile B, all of Profile B's commands can see and modify the same, 'myTextVariable' variable. In most situations this is alright, but let's say your friend created Profile B and your friend also uses a text variable called, 'myTextVariable'. If you switch between Profile A and Profile B, you can see where modifying the same variable can be a problem. Profile A and Profile B would need to be very aware of each other's variables (and that's not much fun).
プロファイルAとプロファイルBの2つのプロファイルがあります。プロファイルAには、テキスト変数「myTextVariable」を作成するコマンドがあります。プロファイルAのコマンドはすべて、「myTextVariable」を表示および変更できます。また、プロファイルBに切り替えると、プロファイルBのすべてのコマンドで同じ「myTextVariable」変数を表示および変更できます。ほとんどの場合、これで問題ありませんが、友人がプロファイルBを作成し、友人も「myTextVariable」というテキスト変数を使用するとします。プロファイルAとプロファイルBを切り替えると、同じ変数の変更が問題になる可能性がある場所を確認できます。プロファイルAとプロファイルBは、互いの変数をよく知っている必要があります（それはあまり面白くないです）。

In v1.6.2, some additional control was introduced to make variables a little more private and more contained. Variables can now also be scoped to stay within the profile (profile-scoped) as well as within individual commands (command-scoped).
v1.6.2では、変数をもう少しプライベートにし、より多く含めるために、いくつかの追加の制御が導入されました。変数は、個々のコマンド（コマンドスコープ）だけでなく、プロファイル（プロファイルスコープ）内にとどまるようにスコープすることもできるようになりました。

In order to not triple up the user interfaces for variables, variable names will now have a prefix associated with them. Profile-scoped variables will be prefixed with the, '>' (greater-than) character, and command-level variables will be prefixed with, '~' (tilde) character. Global variables will remain un-prefixed. So, 'myTextVariable' will be globally-scoped, '>myTextVariable' will be profile-scoped and '~myTextVariable' will be scoped at the command level.
変数のユーザーインターフェイスを3倍にしないために、変数名には接頭辞が関連付けられています。プロファイルスコープの変数には、接頭辞として「>」（より大きい）文字が付けられ、コマンドレベルの変数には接頭辞として「～」（チルダ）文字が付けられます。グローバル変数は接頭辞なしのままになります。だから、「myTextVariableは」グローバルスコープとなり、「> myTextVariableは、」プロファイルのスコープになりますと「～ myTextVariableは、」コマンド・レベルでスコープされます。

Using the previous example for profile-scoped variables, if Profile A and Profile B both happen to have a variable called, '>myTextVariable' (note the, '>' character), each profile will have its own copy of '>myTextVariable'. So, if, '>myTextVariable' in Profile A is set to 'hello', '>myTextVariable' in Profile B can be set independently to, 'world'.
プロファイルスコープ変数の前の例を使用して、プロファイルAとプロファイルBの両方に「> myTextVariable」(「>」文字に注意)という変数がある場合、各プロファイルには「> myTextVariable」の独自のコピーがあります。したがって、プロファイルAの「> myTextVariable」が「hello」に設定されている場合、プロファイルBの「> myTextVariable」は独立して「world」に設定できます。

Command-scoped variables work the same way, except that a command-scoped variable is only available to the command instance in which it is executing. It is not available to any other executing commands that may be running, even if those commands are additional instances of the same command. So, if Command A is executing, Command B cannot read or write any of Command A's command-scoped variables. Additionally, if Command A is run several times in parallel (asynchronously), each running instance of Command A will have its own set of command-scoped variables.
コマンドスコープ変数は同じように機能しますが、コマンドスコープ変数は、それが実行されているコマンドインスタンスでのみ使用できる点が異なります。これらのコマンドが同じコマンドの追加インスタンスであっても、実行中の他の実行コマンドでは使用できません。そのため、コマンドAが実行中の場合、コマンドBはコマンドAのコマンドスコープ変数を読み書きできません。さらに、コマンドAが並列(非同期)で複数回実行される場合、コマンドAの実行中の各インスタンスには、コマンドスコープの変数の独自のセットがあります。

More advanced stuff:
より高度なもの:

If you are still with me down here, there are two more additional prefixes that you can use. First, when a profile is switched, the profile-scoped variables are lost. If you want to retain a profile-scoped variable between profile changes, prefix your variables with two '>' characters. For example, '>myTextVariable' (single, '>') will be lost on profile switch. '>>myTextVariable' (note the double '>') will be retained and will be available when you switch back to that profile.
まだここにいる場合は、さらに2つのプレフィックスを使用できます。まず、プロファイルが切り替えられると、プロファイルスコープの変数は失われます。プロファイルの変更間でプロファイルスコープの変数を保持する場合は、変数の前に2つの「>」文字を付けます。たとえば、「> myTextVariable」(単一、「>」)はプロファイルの切り替え時に失われます。「>> myTextVariable」(二重の「>」に注意)は保持され、そのプロファイルに切り替えると使用可能になります。

The last prefix (and my favorite) is the command-shared scope prefix. Command-scoped
最後のプレフィックス(および私のお気に入り)は、コマンド共有スコーププレフィックスです。コマンドスコープ

variable names that are prefixed with two '~' characters are considered command-shared. That means that the variable will be, 'shared' among commands that are executing in the same execution chain. Basically, the values are passed from one command to any subcommands and then back when control is released from the subcommands. Clear as mud?
2つの「~」文字が前に付いた変数名は、コマンド共有と見なされます。つまり、変数は、同じ実行チェーンで実行されているコマンド間で「共有」されます。基本的に、値は1つのコマンドから任意のサブコマンドに渡され、サブコマンドから制御が解放されると元に戻ります。泥だらけ?

Some examples... First, WITHOUT command-shared variables:
いくつかの例...まず、コマンド共有変数なし:

Command A sets text variable ~myText to 'hi'. Command A then executes Command B as a subcommand. Command B reads ~myText and it is, 'Not Set' (null). That is because
コマンドAは、テキスト変数～myTextを「hi」に設定します。次に、コマンドAはサブコマンドとしてコマンドBを実行します。コマンドBは～myTextを読み取り、「Not Set」（null）です。それは

~myText is command-scoped (single, '~'). It is only available to Command A.
～myTextはコマンドスコープです（単一、'～'）。コマンドAでのみ使用可能です。

WITH command-shared variables:
WITHコマンド共有変数：

Command A sets a shared text variable, '~~myTextVariable' (two, '~') value to 'hello'. Command A then executes subcommand Command B. Since ~~myTextVariable is shared, Command B can then read and write ~~myTextVariable. Command B reads
コマンドAは、共有テキスト変数「~~ myTextVariable」（2つ、「～」）値を「hello」に設定します。次に、コマンドAはサブコマンドCommand Bを実行します。~~ myTextVariableは共有されているため、コマンドBは~~ myTextVariableを読み書きできます。コマンドBの読み取り

~~myTextVariable as, 'hello' and then decides to set ~~myTextVariable to, 'greetings'. When Command B completes its execution and control returns to Command A, Command A will read ~~myTextVariable updated as 'greetings'.
~~ myTextVariableとして 'hello'を設定し、~~ myTextVariableを 'greetings'に設定することにしました。コマンドBが実行を完了し、制御がコマンドAに戻ると、コマンドAは～greetingsとして更新されたmyTextVariableを読み取ります。

Another example: Command A executes Command B. Command B executes Command C. Command C initiates the creation of ~~myTextVariable and sets its value to 'welcome'. When Command C completes, it releases control to Command B. Command B completes and releases control to Command A. Command A reads ~~myTextVariable as, 'welcome'. The idea here is that even though the variable was created two levels down from Command A, it is still accessible to Command A (see note below).
別の例：コマンドAはコマンドBを実行します。コマンドBはコマンドCを実行します。コマンドCは~~ myTextVariableの作成を開始し、その値を「welcome」に設定します。コマンドCが完了すると、コマンドBに制御が解放されます。コマンドBが完了し、コマンドAに制御が解放されます。コマンドAは、~~ myTextVariableを「welcome」として読み取ります。ここでの考え方は、変数がコマンドAから2レベル下に作成されたとしても、コマンドAには引き続きアクセスできるということです（以下の注を参照）。

NOTE: All command-shared variables are passed down to subcommands. If you want updated values to be passed back up to the calling command, the, 'Wait until this command completes before continuing' option MUST be selected for the subcommand. Scoped variables are also accessible via plugins and also work when you save variables to the profile (scope is restored when the variable is retrieved).
注：すべてのコマンド共有変数はサブコマンドに渡されます。更新された値を呼び出しコマンドに戻すには、サブコマンドに対して「続行する前にこのコマンドが完了するまで待機する」オプションを選択する必要があります。スコープ変数はプラグインからもアクセスでき、変数をプロファイルに保存するときにも機能します（変数が取得されるとスコープが復元されます）。

What makes this my favorite is that this is kind of an unorthodox way to be able to pass private information between commands without having to use global variables or establish parameters or return values. Please make sure to visit the VoiceAttack user forums to talk more about this if you are stumped ;)
これが私のお気に入りであるのは、これがグローバル変数を使用したり、パラメーターや戻り値を確立したりすることなく、コマンド間でプライベート情報を渡すことができる一種の非正統的な方法であるということです。困惑している場合は、VoiceAttackユーザーフォーラムにアクセスして、これについて詳しく説明してください。

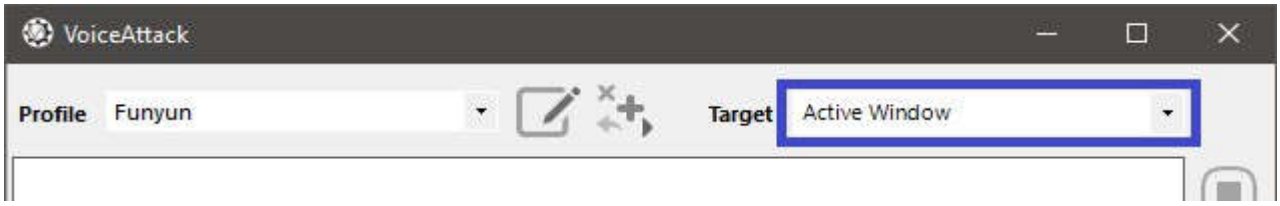
Application Focus (Process Target) Guide
アプリケーションフォーカス（プロセスターゲット）ガイド

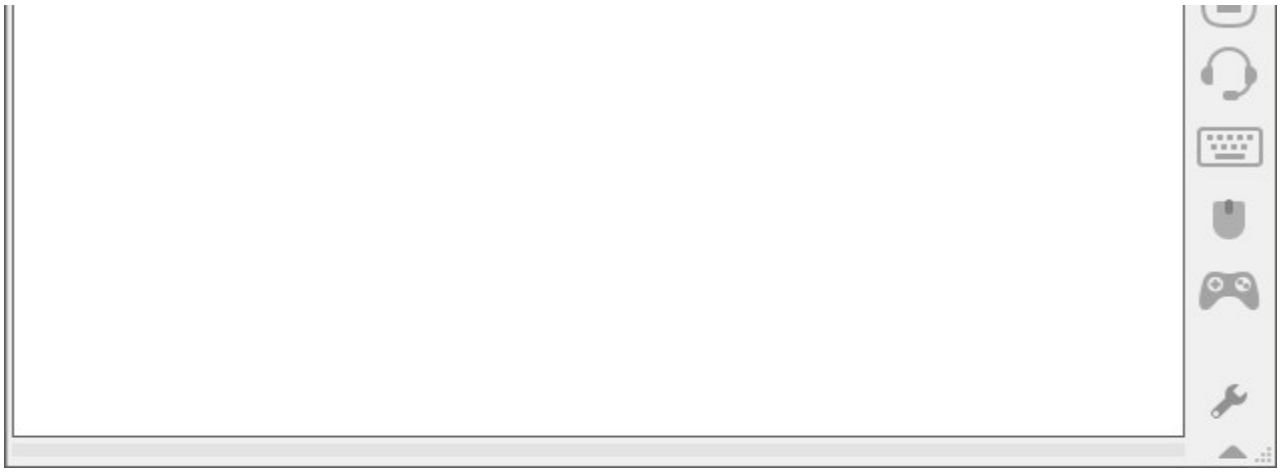
Note: I thought it might be good to consolidate things a bit to get a better picture of what is going on to illustrate what is now available in regards to process targets. In previous versions of VoiceAttack, you had less control over process targets or needed to know a LOT about VoiceAttack in order to change them (like creating sub-commands... bleh).
注：プロセスターゲットに関して現在利用可能なものを説明するために、進行中の状況をよりよく把握するために、少し物事を統合するのが良いと思いました。VoiceAttackの以前のバージョンでは、プロセスターゲットを制御することができなかったか、VoiceAttackを変更するために（サブコマンドの作成...など）、LOTを知る必要がありました。

Hope I don't confuse things even more...
物事をもっと混乱させないことを願っています...

Something you'll notice in Windows is that you need to have a window focused in order for it to receive input. With VoiceAttack, the two forms of input are by keyboard and by mouse. In order to type into, say, a comment box in a form or click an, 'OK' button with the mouse, the containing window needs to be at the top of all other windows (foreground) as well as selected (active). If the intended window is not in the foreground and active, nothing will happen. The input that was supposed to go to that window will be occurring in some other window (the window that is actually selected and in the foreground). In VoiceAttack, the active window in the foreground is called the, 'process target' (or just, 'target').
Windowsで気付くのは、入力を受け取るためにウィンドウをフォーカスする必要があるということです。VoiceAttackでは、キーボードとマウスによる2つの入力形式があります。たとえば、フォームのコメントボックスに入力するか、マウスで[OK]ボタンをクリックするには、含まれているウィンドウが他のすべてのウィンドウ（フォアグラウンド）の一番上にあり、選択済み（アクティブ）である必要があります。目的のウィンドウがフォアグラウンドでアクティブでない場合、何も起こりません。そのウィンドウに移動するはずの入力は、他のウィンドウ（実際に選択され、フォアグラウンドにあるウィンドウ）で発生します。VoiceAttackでは、フォアグラウンドのアクティブウィンドウは「プロセスターゲット」（または単に「ターゲット」）と呼ばれます。

With all this focusing that needs to be done, VoiceAttack supplies a few ways to help out. The first and easiest way is to just use what is called, 'Active Window' as the process target. You can do this by selecting, 'Active Window' in the, 'Target' box on the main screen:
集中する必要があるこのすべてを、VoiceAttackは支援するいくつかの方法を提供します。最初の最も簡単な方法は、「アクティブウィンドウ」と呼ばれるものをプロセスターゲットとして使用することです。これを行うには、メイン画面の[ターゲット]ボックスで[アクティブウィンドウ]を選択します。

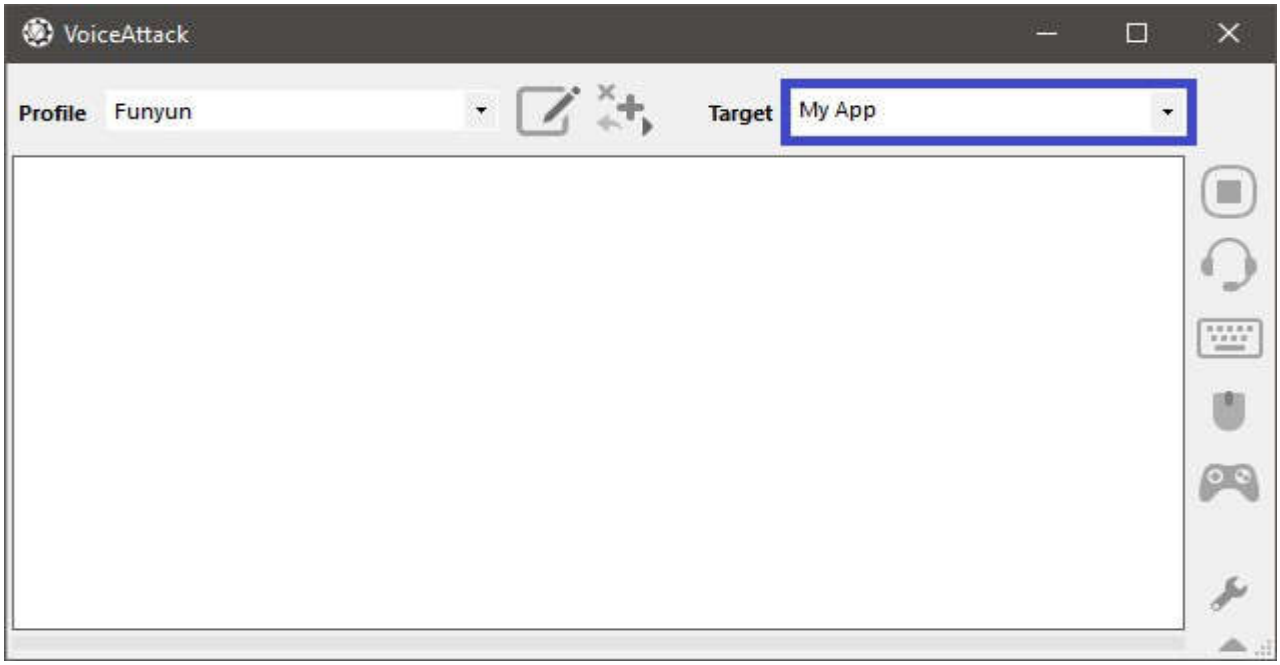




When, 'Active Window' is selected as the process target, whatever application window that is in the foreground and active at the beginning of the executed command will remain VoiceAttack's target for the duration of the command. So, if Notepad is active in the foreground and a command is executed, Notepad will become the target and any key presses or mouse clicks will be sent to it until the command finishes. If you execute the same command and Wordpad is in the foreground and active, all the same actions will occur against Wordpad. It's a very simple system and it is what is used by most (it's also
プロセスターゲットとして「アクティブウィンドウ」が選択されている場合、実行中のコマンドの先頭でフォアグラウンドにあり、アクティブなアプリケーションウィンドウは、コマンドの実行中はVoiceAttackのターゲットのままになります。したがって、メモ帳がフォアグラウンドでアクティブになっていてコマンドが実行されると、メモ帳がターゲットになり、コマンドが終了するまでキーが押されたりマウスがクリックされたりします。同じコマンドを実行し、ワードパッドがフォアグラウンドでアクティブな場合、ワードパッドに対して同じアクションがすべて発生します。これは非常にシンプルなシステムであり、ほとんどの人が使用しています（また、

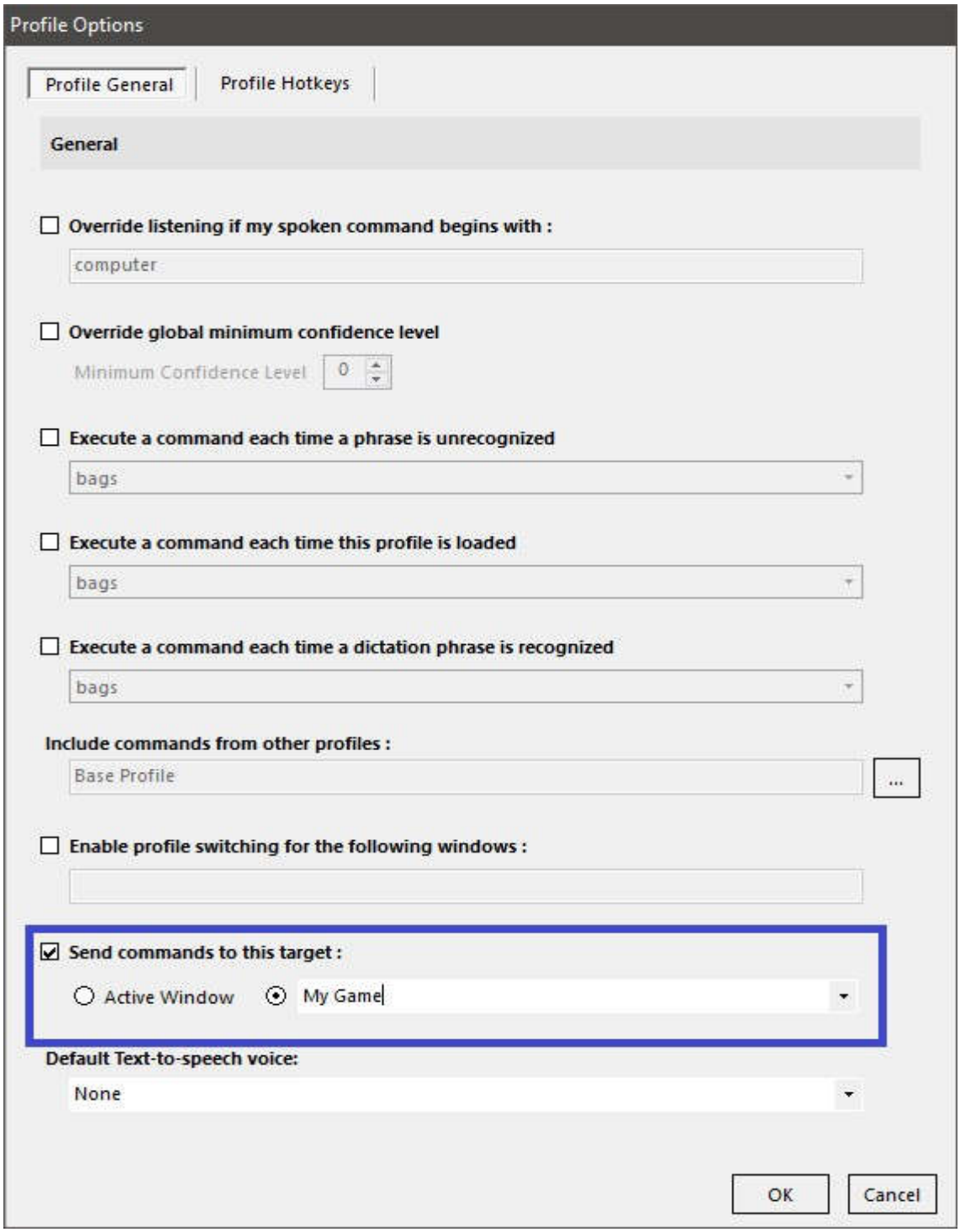
the default setting).
デフォルト設定）。

The second way of targeting a window is by selecting the target by name. Targeting by name will make VoiceAttack seek out the window with the specified name and make it the target (that is, bring it to the foreground and activate it) before sending any input to it. So, if you have an application with a window titled, 'My App', you can have VoiceAttack target the, 'My App' window by simply putting the term, 'My App' in the 'Target to' box:
ウィンドウをターゲットにする2番目の方法は、名前でターゲットを選択することです。名前でターゲットを設定すると、VoiceAttackは指定された名前のウィンドウを探し出し、入力を送信する前にターゲットにします（つまり、フォアグラウンドに移動してアクティブにします）。したがって、「My App」というタイトルのウィンドウを持つアプリケーションがある場合、「Target to」ボックスに「My App」という用語を入力するだけで、VoiceAttackが「My App」ウィンドウをターゲットにすることができます。



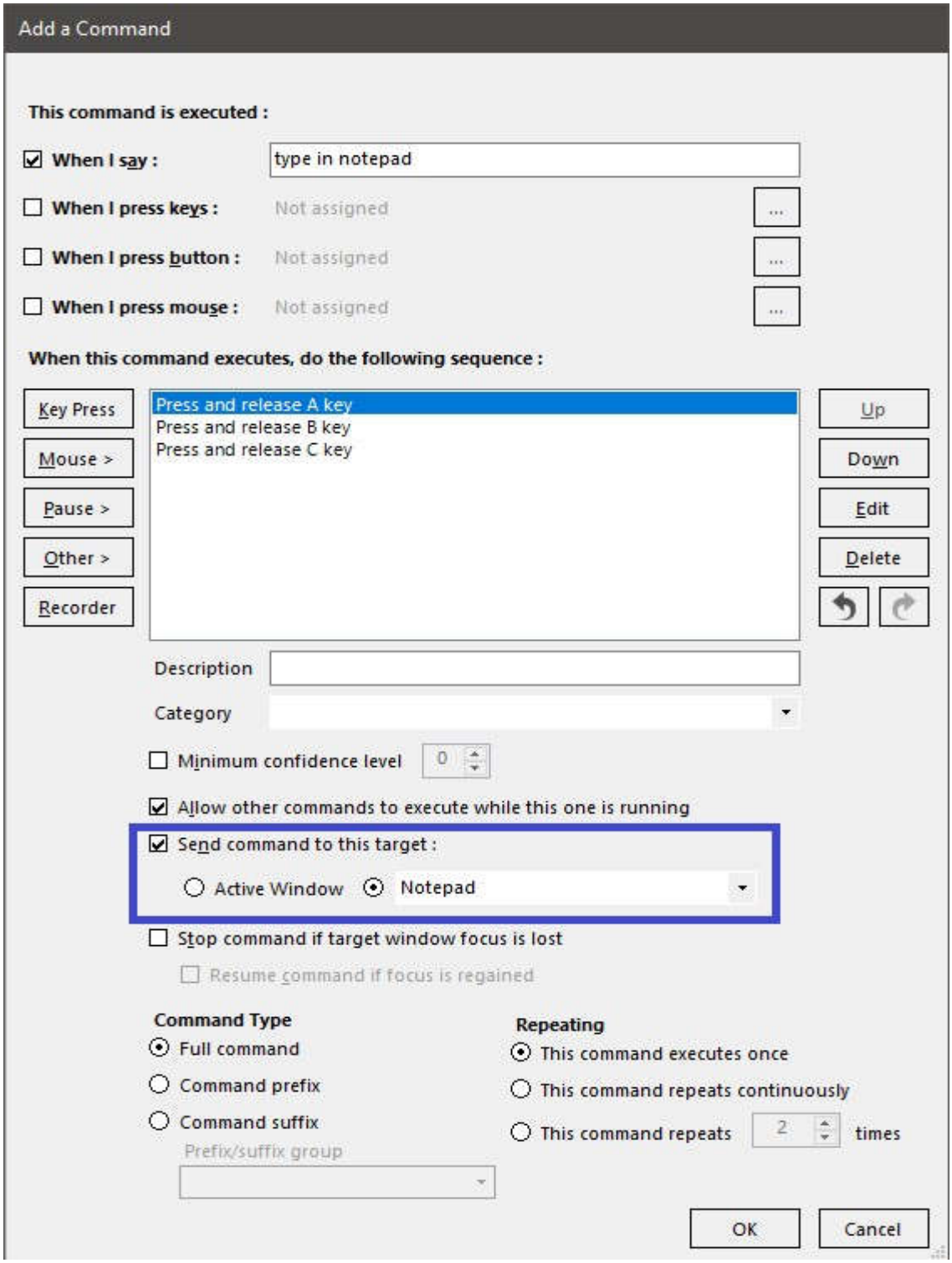
Note that you can freely type in the, 'Target' box, as well as select applications from the list. You'll also be able to select from some recently used items that you have added.
[ターゲット]ボックスに自由に入力でき、リストからアプリケーションを選択できることに注意してください。また、追加した最近使用したアイテムから選択することもできます。

To keep targeting flexible, either of these methods can be selected from the main screen as shown (global-level), from the profile options screen of any profile (profile-level) or from a command within a profile (command-level). The target is selected in a cascading manner: the command-level target will override anything specified at the profile level and a profile- level target will override anything at the global level. Maybe you want, 'Active Window' to be the preferred targeting method at the global level, but you want all the commands in a certain profile to be directed at a game. For that type of scenario, simply leave 'Active Window' selected on the Main screen and select the game's window title, 'My Game' from the 'Send commands to this target' drop-down on the Profile Options screen:
ターゲット設定を柔軟に保つために、これらの方法のいずれかを、示されているメイン画面（グローバルレベル）、任意のプロファイルのプロファイルオプション画面（プロファイルレベル）、またはプロファイル内のコマンド（コマンドレベル）から選択できます。ターゲットはカスケード方式で選択されます。コマンドレベルのターゲットは、プロファイルレベルで指定されたすべてのものをオーバーライドし、プロファイルレベルのターゲットは、グローバルレベルですべてのものをオーバーライドします。「アクティブウィンドウ」をグローバルレベルの優先ターゲティング方法にしたいが、特定のプロファイルのすべてのコマンドをゲームに向けたい場合があります。このタイプのシナリオでは、メイン画面で「アクティブウィンドウ」を選択したまま、プロファイルオプション画面の「このターゲットにコマンドを送信」ドロップダウンからゲームのウィンドウタイトル「マイゲーム」を選択します。



All profiles will continue to use the, 'Active Window' as the target EXCEPT the profile where you had specified, 'My Game'.
すべてのプロファイルは、指定したプロファイル「マイゲーム」を除き、「アクティブウィンドウ」をターゲットとして使用し続けます。

Let's say in that same profile you want to be able to also type some characters into Notepad. Since, 'My Game' is the process target for the entire profile, you will need to be able to specify a process target just for the command. You can do this by putting, 'Notepad' in the, 'Send command to this target' box on the Command Screen:
同じプロファイルで、いくつかの文字をメモ帳にも入力できるようにしたいとします。「マイゲーム」はプロファイル全体のプロセスターゲットなので、コマンドだけでプロセスターゲットを指定できる必要があります。これを行うには、コマンド画面の「このターゲットにコマンドを送信」ボックスに「メモ帳」を配置します。


Add a Command
This command is executed :
When I say :
type in notepad
When I press keys :
Not assigned
When I press button :
Not assigned
When I press mouse :
Not assigned
When this command executes, do the following sequence :
Key Press
Mouse >
Pause >
Other >
Recorder
Press and release A key
Press and release B key
Press and release C key
Up
Down
Edit
Delete
Description
Category
Minimum confidence level
0
Allow other commands to execute while this one is running
Send command to this target :
Active Window
Notepad
Stop command if target window focus is lost
Resume command if focus is regained
Command Type
Full command
Command prefix
Command suffix
Prefix/suffix group
Repeating
This command executes once
This command repeats continuously
This command repeats
2
times
OK
Cancel

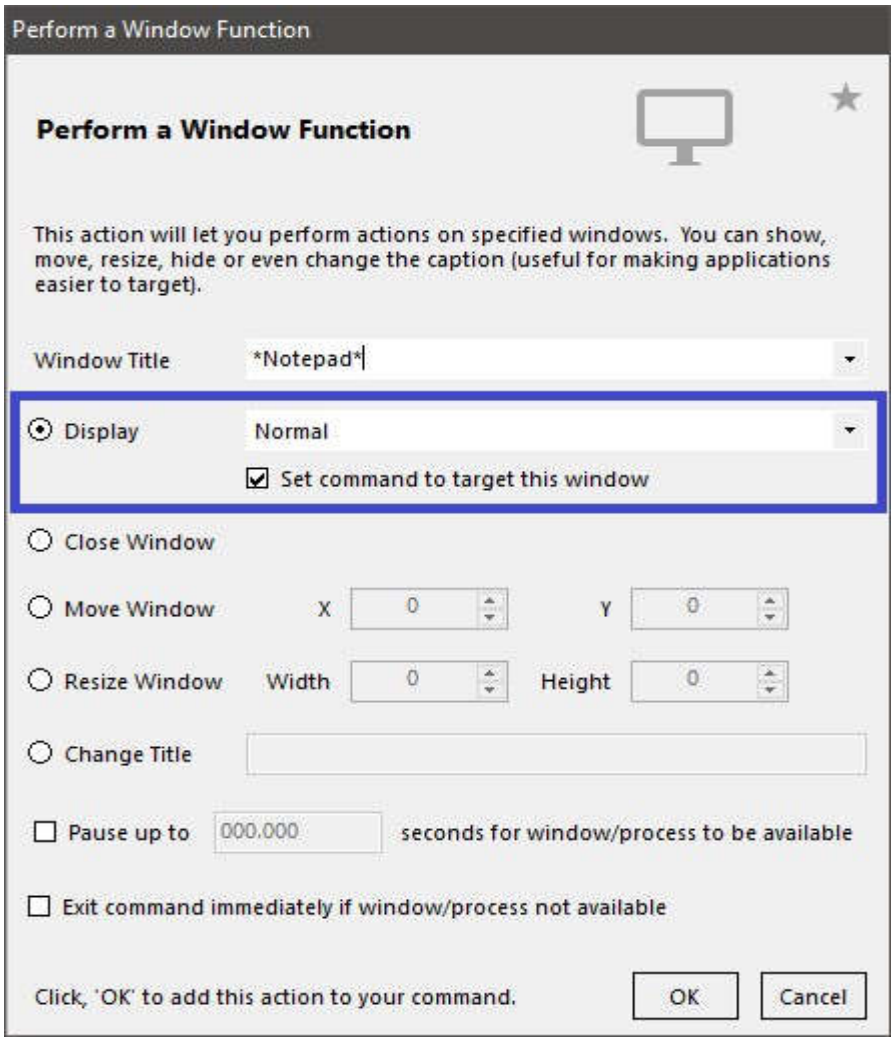
Note: If your window title is the type that changes, you can even specify wildcards in the process target box to make finding the window a little easier by only using a portion of the title (search for, 'wildcard' in this document for more information). You can even change the window's title completely if you need more control (see, 'Perform a Window Function' action for more info). Also, if you do not have the window title handy, you can indicate the process name as it is displayed in Windows Task Manager. The wildcards apply here as well. If the window title and/or process name do not suffice, you can supply the window class name. This is a super-advanced feature and will require you to know the class name for the window. You will need a third-party tool such as Spy++ to get this information.
注：ウィンドウタイトルが変更されるタイプである場合、タイトルの一部のみを使用することでウィンドウを見つけやすくするために、プロセスターゲットボックスでワイルドカードを指定することもできます（このドキュメントで「ワイルドカード」を検索して、情報）。さらに制御が必要な場合は、ウィンドウのタイトルを完全に変更することもできます（詳細については、「ウィンドウ関数の実行」アクションを参照してください）。また、ウィンドウタイトルが手元にない場合は、Windowsタスクマネージャに表示されるプロセス名を指定できます。ここにもワイルドカードが適用されます。ウィンドウのタイトルやプロセス名では不十分な場合は、ウィンドウクラス名を指定できます。これは非常に高度な機能であり、ウィンドウのクラス名を知る必要があります。この情報を取得するには、Spy ++などのサードパーティツールが必要です。

Wildcards do apply for the window class name if you need them. The window title is searched for first, then the process name and then the window class name.
必要な場合、ウィンドウクラス名にはワイルドカードが適用されます。ウィンドウタイトルが最初に検索され、次にプロセス名、次にウィンドウクラス名が検索されます。

Advanced targeting
高度なターゲティング

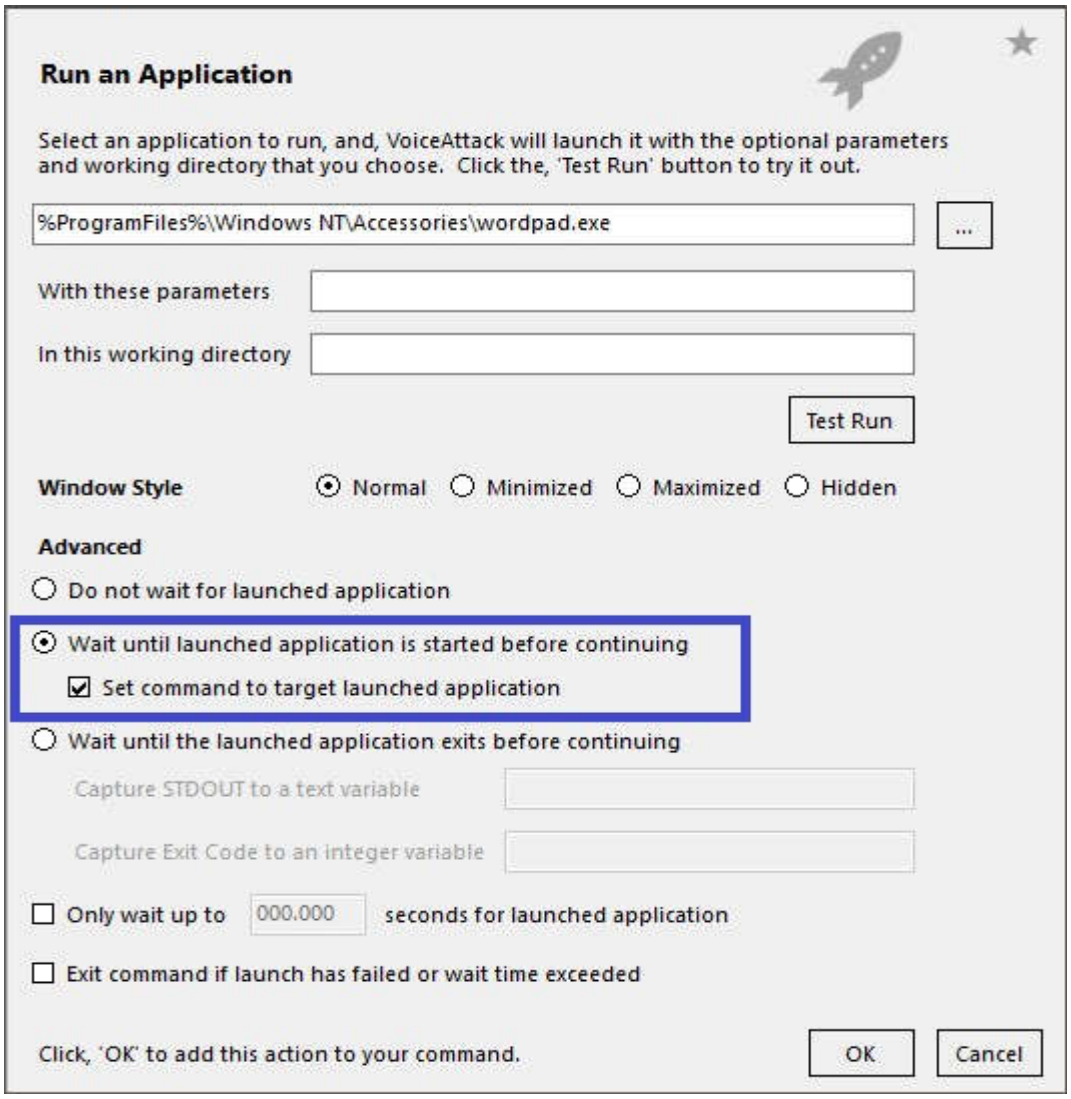
For the most part, the methods above will fill the needs of almost all command macros. The only drawback is that the targeting is too rigid for some situations. For example, when targeted by, 'Active Window', a window remains the target throughout the duration of the command, EVEN IF you manually click another window and make it the foreground window (VoiceAttack will change the process target window BACK to the foreground, selected window as it continues through the command). This is so that activities can occur with other windows, but any input commands (such as pressing keys or typing a message) will continue to be sent to the indicated process target. Similarly, selecting a target by name will only seek out the named window for the entire command. Sometimes there is the need to change the process target even in the middle of a command while it executes. You may need to have a specified target at the start of a command just to get a handle on it, and then redirect input to another application. There may also be the need to launch an application and immediately start sending input to it even if you don't know what it's window title will be. The good news is that VoiceAttack has some features built in to help you handle these situations.
ほとんどの場合、上記の方法はほとんどすべてのコマンドマクロのニーズを満たします。唯一の欠点は、状況によってはターゲット設定が厳しすぎることです。たとえば、「アクティブウィンドウ」をターゲットにすると、別のウィンドウを手動でクリックしてフォアグラウンドウィンドウにした場合でも、コマンドの実行中はウィンドウがターゲットのままになります（VoiceAttackはプロセスターゲットウィンドウをフォアグラウンドに戻しますが、コマンドを続行するときに選択されたウィンドウ）。これにより、他のウィンドウでアクティビティを実行できますが、入力コマンド（キーを押す、メッセージを入力するなど）は、引き続き指定されたプロセスターゲットに送信されます。同様に、名前でターゲットを選択すると、コマンド全体の名前付きウィンドウのみが検索されます。実行中のコマンドの途中であっても、プロセスターゲットを変更する必要がある場合があります。コマンドの開始時に指定されたターゲットを使用して、そのハンドルを取得し、入力を別のアプリケーションにリダイレクトする必要がある場合があります。また、ウィンドウタイトルが何であるかわからない場合でも、アプリケーションを起動してすぐに入力を送信する必要がある場合があります。幸いなことに、VoiceAttackには、こうした状況に対処するのに役立ついくつかの機能が組み込まれています。

The new process target may need to be launched, or the new target may already be running. If the new target is already running, you can change to this target by using the, 'Display' feature in the, 'Perform a Window Function' action and selecting the, 'Set command to target this window' option:
新しいプロセスターゲットを起動する必要があるか、新しいターゲットが既に実行されている可能性があります。新しいターゲットが既に実行されている場合、「ウィンドウ機能の実行」アクションで「表示」機能を使用し、「このウィンドウをターゲットにするコマンドを設定」オプションを選択して、このターゲットに変更できます。



What will happen when this action is run is if the target window is found, it will then be set as the process target for the remaining duration of the command. Note that you can set how long to wait for the new target to be active, as well as exit the command completely if the wait time is exceeded (or if the window just simply can't be found).
このアクションが実行されると、ターゲットウィンドウが見つかった場合、コマンドの残りの期間のプロセスターゲットとして設定されます。新しいターゲットがアクティブになるまでの待機時間を設定できるほか、待機時間を超えた場合（またはウィンドウが単に見つからない場合）にコマンドを完全に終了できることに注意してください。

If the new target needs to be launched, just use the 'Run an Application' action and select the, 'Wait until launched application is started before continuing' feature with the, 'Set command to target launched application' option selected:
新しいターゲットを起動する必要がある場合は、「アプリケーションの実行」アクションを使用して、「起動したアプリケーションが起動するまで待機してから続行する」機能を選択し、「起動したアプリケーションをターゲットに設定」オプションを選択します：



What will happen is when the application is launched, VoiceAttack will wait until the app is ready to accept input and then set that application as the new process target for the remaining duration of the command. Note that you can also specify how long to wait and to exit if launching the app exceeds the indicated time (or if the launched app fails to launch).
アプリケーションが起動されると、VoiceAttackはアプリが入力を受け入れる準備ができるまで待機し、コマンドの残りの期間、そのアプリケーションを新しいプロセスターゲットとして設定します。アプリの起動が指定の時間を超えた場合（または起動したアプリの起動に失敗した場合）に待機する時間と終了する時間も指定できることに注意してください。

The last way to set a process target happens automatically whenever VoiceAttack uses the mouse to, 'click' on a window (including the desktop). Whenever a mouse button down event occurs (mouse down, click, double-click), the window that is located under the mouse will become the process target (if it is not already) for the remaining duration of the command. This is to simulate what happens when a hardware mouse is clicked. To turn
プロセスターゲットを設定する最後の方法は、VoiceAttackがマウス（デスクトップを含む）を「クリック」するためにマウスを使用するたびに自動的に行われます。マウスボタンダウンイベント（マウスダウン、クリック、ダブルクリック）が発生すると、コマンドの残りの期間、マウスの下にあるウィンドウがプロセスターゲットになります（まだない場合）。これは、ハードウェアマウスがクリックされたときに何が起こるかをシミュレートするためです。回す

this functionality off and go back to what was in place before, simply select the, 'Bypass Mouse Targeting' option on the options screen.
この機能をオフにして前の設定に戻り、オプション画面で[マウスターゲティングのバイパス]オプションを選択します。

Targeting helpers
ヘルパーのターゲティング

Commands that are executing are aware of their process target. There are couple of options to note on the command screen: 'Stop command if target window focus is lost' and, 'Resume command if focus is regained'. 'Stop command if target window focus is lost' will stop the command completely if the process target's focus is lost. If you select, 'Resume command if focus is regained', the command will be paused until the process target is active again.
実行中のコマンドは、プロセスターゲットを認識しています。コマンド画面には、「ターゲットウィンドウのフォーカスが失われた場合はコマンドを停止する」、「フォーカスが戻った場合はコマンドを再開する」という注意事項があります。「ターゲットウィンドウのフォーカスが失われた場合にコマンドを停止」は、プロセスターゲットのフォーカスが失われた場合にコマンドを完全に停止します。「フォーカスが回復したらコマンドを再開する」を選択すると、コマンドはプロセスターゲットが再びアクティブになるまで一時停止します。

A related feature that has the potential to change the process target is the automatic profile switching feature. We won't go into any detail here about that, but it's worth mentioning.
プロセスターゲットを変更する可能性のある関連機能は、自動プロファイル切り替え機能です。ここでは詳細については説明しませんが、言及する価値はあります。

Command Execution Queues Overview
コマンド実行キューの概要

Command execution queues (or just, 'queues') execute commands in the order that you add (enqueue) them, making sure that each command is fully executed before moving on to the next command. This will help you execute commands one after the next without having to do all the monitoring yourself. This will also add just a little bit more flexibility in the way that commands can be handled when executed. Think of queues like you would checkout lanes at the grocery store. If a customer (command) gets in line (enqueued), the customer must wait until the person in front of them is finished before checking out (executing). Each subse- quent customer must wait until the customer in front of them is finished before being checked out. The lane can fill up and have any number of customers, but, each customer must wait their turn in the order that they showed up.
コマンド実行キュー（または単に「キュー」）は、追加（エンキュー）する順序でコマンドを実行し、次のコマンドに進む前に各コマンドが完全に実行されることを確認します。これにより、すべての監視を自分で行う必要なく、次から次へとコマンドを実行できます。これにより、実行時にコマンドを処理する方法に柔軟性が少し追加されます。食料品店のレーンをチェックアウトするようなキューを考えてください。顧客（コマンド）が並んでいる（エンキューされている）場合、顧客はチェックアウト（実行）する前に彼らの前の人が終了するまで待たなければなりません。後続の各顧客は、自分の前の顧客が終了するまで待ってからチェックアウトする必要があります。レーンはいっぱいになり、何人でも顧客を抱えることができますが、

Let's say you are firing various weapons from your ship, but those operations must occur one after the next (never at the same time for various reasons). Let's say you'd like to first fire your missiles, followed by the rail gun, then machine guns. Normally, if you issued com- mands for all three of these things in rapid succession, they would all execute at pretty much the same time. If each command is brief, that is usually not much of a problem, but if the commands take a certain amount of time or depend on one another to execute in order, then, that can be an issue. If you enqueue each of these operations - first enqueue the command to fire the missiles, then enqueue the command to fire the rail gun and finally enqueue the command to fire the machine gun - each operation will wait for the last to finish before execut- ing, no matter how long each takes to process.
船からさまざまな武器を発射しているとしましょう。ただし、これらの操作は次々に実行する必要があります（さまざまな理由で同時に実行することはできません）。最初にミサイルを発射し、次にレールガン、次にマシンガンを発射するとします。通常、これら3つのすべてのコマンドをすばやく連続して発行した場合、それらはほぼ同時に実行されます。各コマンドが短い場合、通常はそれほど問題ではありませんが、コマンドの実行に一定の時間がかかるか、相互に依存して順番に実行する場合は、問題になる可能性があります。これらの各操作をキューに入れる場合-最初にコマンドをキューに入れてミサイルを発射し、次にコマンドをキューに入れてレールガンを発射し、最後にコマンドをキューに入れてマシンガンを発射します-各操作は最後の完了を待ってから実行します、

So, what if you don't want your queued shield commands to wait for your weapon commands to finish? Queues can be named, so that you can have as many queues as you would like to use for different purposes, even at the same time. So, if you have a particular order that you would like to fire weapons (like above), you can create a queue named, 'weapons' and add each weapon command to that queue. You can then create a queue named, 'shields' and then add shield-related commands to that queue. Each queue will execute its commands in- dependently and in the order the commands were added. Note that once a queue is created, it will stay resident until VoiceAttack is closed.
それで、キューに入れられたシールドコマンドが武器コマンドの終了を待てないようにするにはどうすればいいですか？キューには名前を付けることができるため、同時にでも、さまざまな目的に使用するキューをいくつでも持つことができます。したがって、武器を発射する特定の順序がある場合（上記のように）、「weapons」という名前のキューを作成し、そのキューに各武器コマンドを追加できます。次に、「shields」という名前のキューを作成し、そのキューにシールド関連のコマンドを追加できます。各キューは、コマンドを独立して、コマンドが追加された順に実行します。キューが作成されると、VoiceAttackが閉じられるまでキューは常駐することに注意してください。

Other neat things that queues can do is that queues can be preloaded with commands to be executed once the queue is filled how you like it, or, the queue's commands can start execu- tion immediately. Queues can also be paused so that once a command is completed, the next command's execution will be postponed until you would like the queue to continue.
キューができる他のすばらしいことは、キューが好きなように満たされると実行されるコマンドをキューにプリロードできること、またはキューのコマンドがすぐに実行を開始できることです。また、キューを一時停止して、コマンドが完了すると、次のコマンドの実行がキューを続行するまで延期されるようにすることもできます。

Queues can also remain running even after they are empty so that any time a new command is added, that command will immediately execute. There's a good bit you can do, and it may seem a little confusing, so, let's walk through creating a queue so you can see how it works.
また、キューは空になった後でも実行を継続できるため、新しいコマンドが追加されるたびに、そのコマンドはすぐに実行されます。できることは少しありますが、少しわかりにくいかもしれません。そのため、キューを作成して、どのように機能するかを見てみましょう。

Creating a Command Execution Queue
コマンド実行キューの作成

You create a command execution queue just by simply enqueueing a command. When you enqueue a command, it's pretty much the same as the old, 'Execute Another Command' ac- tion that you've been using for some time. The difference is that when you enqueue a com- mand, instead of the command immediately executing, the command is handed off to a queue that manages that command and any subsequent command that is enqueued. To enqueue a
コマンドをキューに登録するだけで、コマンド実行キューを作成できます。コマンドをキューに入れるとき、それはあなたがしばらく使用していた古い「別のコマンドを実行する」アクションとほとんど同じです。違いは、コマンドをすぐに実行するのではなく、コマンドをすぐに実行するとき、コマンドはそのコマンドとその後にキューに入れられるコマンドを管理するキューに渡されることです。エンキューするには

command, simply add an, 'Enqueue a Command' action by clicking, Other > VoiceAttack Ac- tion > Command Queues > Enqueue a Command from the Command screen. From this screen, you'll first want to indicate the name of the queue in which you would like to add the command. Simply type in a name, or, if you've already set up a queue elsewhere, you can select it from this list. The reason that you would want to name your queue is so that you can have more than one active queue. So, if you have a particular order that you would like to fire weapons, you can create a queue named, 'weapons' and add each weapon command to that queue. You can then create a queue named, 'shields' and then add shield-related commands to that queue. Each queue will execute its commands independently and in the order the commands were added.
コマンドを追加するには、[その他]> [VoiceAttackアクション]> [コマンドキュー]> [コマンドをキューに登録]をクリックして、[コマンドをキューに追加]アクションを追加します。この画面から、最初にコマンドを追加するキューの名前を指定します。名前を入力するか、他の場所に既にキューを設定している場合は、このリストから選択できます。キューに名前を付けたい理由は、複数のアクティブなキューを持つことができるようにするためです。そのため、武器を発射したい特定の順序がある場合、「武器」という名前のキューを作成し、各武器コマンドをそのキューに追加できます。次に、「shields」という名前のキューを作成し、そのキューにシールド関連のコマンドを追加できます。

Next, you will want to indicate whether or not the command is to start executing the moment it is added. If you want the command to start executing immediately, select the, 'Start queue when this command is added' option. If this option is not selected, the command is added to the queue, but queue is not started until you explicitly start it (or, enqueue a subsequent com- mand with this option selected). Your queue can also be started explicitly by executing a, 'Queue Start' action (more about that down below). Once started, your queue will keep wait- ing for you to add more commands to it. If no commands are in the queue, the next added command will run immediately. If a command is still running in the queue, the latest enqueued command will have to wait until the executing command is finished as well as any other waiting command.
次に、コマンドが追加された瞬間に実行を開始するかどうかを示します。コマンドの実行をすぐに開始する場合は、[このコマンドが追加されたときにキューを開始する]オプションを選択します。このオプションが選択されていない場合、コマンドはキューに追加されますが、明示的に開始するまで（または、このオプションを選択して後続のコマンドをキューに入れるまで）キューは開始されません。また、「Queue Start」アクションを実行して、キューを明示的に開始することもできます（これについては以下で詳しく説明します）。開始すると、キューはコマンドを追加するのを待ち続けます。キューにコマンドがない場合、追加された次のコマンドはすぐに実行されます。コマンドがまだキューで実行されている場合、最新のエンキューされたコマンドは、実行中のコマンドが終了するまで待機する必要があります。他の待機コマンドも同様です。

The Queue Actions
キューアクション

In most cases, your command execution queue will be started up and then can just be forgot- ten. Sometimes you may want to have a little bit of extra control over your queues. There are a few actions that you can perform on a command execution queue, such as starting, stop- ping and pausing. Note that you can indicate a specific queue or affect all queue instances.
ほとんどの場合、コマンド実行キューは起動された後、忘れられます。場合によっては、キューを少し制御したいことがあります。開始、停止、一時停止など、コマンド実行キューで実行できるアクションがいくつかあります。特定のキューを指定したり、すべてのキューインスタンスに影響を与えることができることに注意してください。

The actions are outlined below (you can find more detailed descriptions of each in the, ‘Com- mand Queues - Queue Action’ section earlier in this document):
アクションの概要は次のとおりです（各ドキュメントの詳細については、このドキュメントの「コマンドキュー-キューアクション」セクションを参照してください）。

Start - This will make your queue start executing commands in the order that they were added.
開始-これにより、キューはコマンドが追加された順序で実行を開始します。

Pause - Pauses the queue once the currently-executing command has completed.
一時停止-現在実行中のコマンドが完了すると、キューを一時停止します。

Unpause - Unpauses the queue (starts command execution).
一時停止解除-キューの一時停止を解除します（コマンドの実行を開始します）。

Toggle pause/unpause
一時停止/一時停止の切り替え

Stop - Stops and clears the queue, also stopping any command that may be executing within the queue.
停止-キューを停止してクリアし、キュー内で実行中のコマンドを停止します。

Stop, but allow current command to complete - This will do everything the Stop action will do, except the current command will be allowed to complete.
停止しますが、現在のコマンドの完了を許可します-これは、現在のコマンドの完了を許可することを除き、停止アクションが実行するすべてを実行します。

Supporting Tokens
サポートトークン

You will be able to access various details about your command execution queues by using tokens. Information such as queue status (whether or not a queue exists, if it is running or paused), how many commands a queue has in it and whether or not a command is executing. See the section labeled, ‘Token reference’ earlier in this document for more info on these items (Note that the queue tokens all start with ‘{QUEUE’).
トークンを使用して、コマンド実行キューに関するさまざまな詳細にアクセスできます。キューの状態（キューが存在するかどうか、実行中または一時停止中の場合）、キューに含まれるコマンドの数、コマンドが実行中かどうかなどの情報。これらの項目の詳細については、このドキュメントの「トークンリファレンス」のセクションを参照してください（キュートークンはすべて「{QUEUE」で始まることに注意してください）。

VoiceAttack Profile Package Reference
VoiceAttackプロファイルパッケージリファレンス

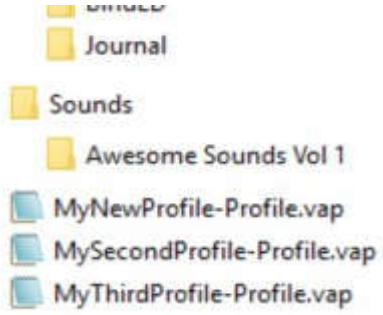
You can import a collection of files into VoiceAttack using a VoiceAttack package file. A package file can contain any number of exported VoiceAttack profiles (.vap), sounds or plugin/apps. A VoiceAttack package file is a .zip containing all the necessary files to make up a profile package, and has a '.vax' extension.
VoiceAttackパッケージファイルを使用して、ファイルのコレクションをVoiceAttackにインポートできます。パッケージファイルには、エクスポートされたVoiceAttackプロファイル（.vap）、サウンド、またはプラグイン/アプリをいくつでも含めることができます。VoiceAttackパッケージファイルは、プロファイルパッケージを構成するために必要なすべてのファイルを含む.zipであり、 'があります。vax '拡張。

To import a VoiceAttack profile package, simply click on, 'Import Profile' from the Main screen. Select a package file (.vax) and the import will start. The files are copied into the folders you have specified for Sounds and Apps (on the Options screen), and the included profile files (.vap) are imported as profiles.
VoiceAttackプロファイルパッケージをインポートするには、メイン画面から[プロファイルのインポート]をクリックするだけです。パッケージファイル（.vax）を選択すると、インポートが開始されます。ファイルは、（オプション画面で）サウンドとアプリ用に指定したフォルダーにコピーされ、含まれているプロファイルファイル（.vap）がプロファイルとしてインポートされます。

To create a VoiceAttack package to share, you simply zip together the necessary exported profiles (.vap) and the folders (sounds and apps) that you would like to include. The profiles MUST be in the root of the .zip (not inside a folder inside the .zip). Any sounds must be in a folder named, 'Sounds' in the .zip (the, 'Sounds' folder MUST be in the root of the .zip (not inside a folder inside the .zip)). The file structure *inside* the, 'Sounds' folder is what will be copied to the folder designated as the Sounds folder for VoiceAttack (what is set on the Options screen). Note that any file with the same path and filename will be overwritten (useful for large updates).
共有するVoiceAttackパッケージを作成するには、必要なエクスポートされたプロファイル（.vap）と、含めるフォルダー（サウンドとアプリ）を一緒に圧縮します。プロファイルは.zipのルートにある必要があります（.zip内のフォルダー内ではありません）。すべてのサウンドは、.zipの「Sounds」という名前のフォルダーになければなりません（「Sounds」フォルダーは.zipのルートにある必要があります（.zip内のフォルダー内ではありません））。'Sounds'フォルダーの*内部*のファイル構造は、VoiceAttackのSoundsフォルダーとして指定されたフォルダー（オプション画面で設定されているもの）にコピーされます。同じパスとファイル名を持つファイルは上書きされることに注意してください（大規模な更新に役立ちます）。

In exactly the same fashion as the, 'Sounds' folder, any plugins/apps that are in the package must be in a folder named, 'Apps' in the .zip (the, 'Apps' folder MUST be in the root of the .zip (not inside a folder inside the .zip)). The file structure *inside* the, 'Apps' folder is what will be copied to the folder designated as the Apps folder for VoiceAttack (what is set on the Options screen). Note that any file with the same path and filename will be overwritten (again, useful for updates). Note that since plugins are .dll files, the plugin feature must be disabled (and VoiceAttack restarted) prior to importing the, 'Apps' folder. Note that if you do NOT have any plugins/apps, do not include an, 'Apps' folder (there is a check for the presence of an, 'Apps' folder in the package).
「サウンド」フォルダーとまったく同じ方法で、パッケージ内のすべてのプラグイン/アプリは、.zip内の「アプリ」という名前のフォルダー内になければなりません（「アプリ」フォルダーは、 .zip（.zip内のフォルダー内ではない））。「内部」の「Apps」フォルダーのファイル構造は、VoiceAttackのAppsフォルダーとして指定されたフォルダー（オプション画面で設定されているもの）にコピーされます。同じパスとファイル名を持つファイルは上書きされることに注意してください（これも更新に役立ちます）。プラグインは.dllファイルであるため、「Apps」フォルダーをインポートする前に、プラグイン機能を無効にする（およびVoiceAttackを再起動する）必要があることに注意してください。プラグイン/アプリがない場合は、「アプリ」フォルダーを含めないでください（「アプリ」の存在のチェックがあります）

Here is an image of a sample package that contains profiles, sounds and plugins:
プロファイル、サウンド、プラグインを含むサンプルパッケージの画像を次に示します。



Note again that everything is in the root of the zip (and not in a folder in the .zip lol).
繰り返しますが、すべてがzipのルートにあることに注意してください(.zip lolのフォルダーにはありません)。

Two additional, 'hidden' features of the package import are two flags that are included with exported profiles (version v1.6.5+). You'll have to edit the .vap files with a text editor to get to them, though. The first flag is, 'IO' (for import order). You'll see this flag somewhere near the bottom of the file as '<IO>0</IO>'. The profiles included will be ordered by this flag. So, if Profile, 'A' has an IO flag of '0', and Profile, 'B' has an IO flag set to '1', Profile A will be
パッケージインポートの2つの追加の「隠された」機能は、エクスポートされたプロファイル(バージョンv1.6.5 +)に含まれる2つのフラグです。ただし、.vapファイルをテキストエディターで編集してファイルにアクセスする必要があります。最初のフラグは 'IO'(インポート順)です。このフラグは、ファイルの下部近くに「<IO> 0 </ IO>」として表示されます。含まれるプロファイルは、このフラグによって順序付けられます。したがって、プロファイル「A」のIOフラグが「0」であり、プロファイル「B」のIOフラグが「1」に設定されている場合、プロファイルAは

imported first, followed by Profile B. Is order important? Probably not at the moment, but if necessary at some point, it's there. The second flag is, 'IS' (for import select). If this flag is set to 1 (or non-zero), VoiceAttack will switch to this profile when it is imported (just like when you import a single profile (.vap)). If this value remains zero, VoiceAttack will not switch away from the currently-selected profile when the package is finished importing.
最初にインポートされ、次にプロファイルBがインポートされます。順序は重要ですか?おそらく現時点ではないかもしれませんが、必要な場合はいつかそこにあります。2番目のフラグは 'IS'(インポート選択用)です。このフラグが1(またはゼロ以外)に設定されている場合、VoiceAttackはインポートされるときにこのプロファイルに切り替わります(単一のプロファイル(.vap)をインポートする場合と同様)。この値がゼロのままである場合、VoiceAttackは、パッケージのインポートが終了しても、現在選択されているプロファイルから切り替えません。

The last thing to do is to make sure that your zipped up file is renamed with the extension of '.vax' so that VoiceAttack will pick it up. That's all there is to it! Since this is a relative advanced topic, you'll probably want to come visit everybody over at the VoiceAttack User Forums for more info: http://www.voiceattack.com/forum
最後に行うことは、VoiceAttackがそれを取得できるように、圧縮したファイルの拡張子を「.vax」に変更することです。それだけです!これは比較的高度なトピックであるため、詳細についてはVoiceAttackユーザーフォーラムにアクセスしてください。http://www.voiceattack.com/forum

Troubleshooting Guide
トラブルシューティングガイド

This section is going to be updated frequently, so, check back. Also, if this guide does not help you, please visit the VoiceAttack forums at http://www.voiceattack.com/forum
このセクションは頻繁に更新されるため、もう一度確認してください。また、このガイドが役に立たない場合は、http://www.voiceattack.com/forumのVoiceAttackフォーラムにアクセスしてください。

(edit – this guide is only covering some basic items which may help you in your particular situation. The following thread on the forum is going to help you the most with tough speech engine stuff (fairly new, fairly complete):
(編集–このガイドは、あなたの特定の状況であなたを助けるかもしれないいくつかの基本的な項目のみをカバーします。フォーラムの次のスレッドはあなたに最も助けになります(かなり新しい、かなり完全な):

http://voiceattack.com/SMF/index.php?topic=1635.0 note that there are links in that thread to other threads that will help as well.).
http://voiceattack.com/SMF/index.php?topic=1635.0 そのスレッドには、他のスレッドへのリンクもあります。)。

Almost every single issue with VoiceAttack will have to do with VoiceAttack not hearing or understanding your commands.
VoiceAttackのほとんどすべての問題は、VoiceAttackがコマンドを聞いたり理解したりしないことに関係しています。

Here are some things I've found while using VoiceAttack over the last year or so...
ここ1年ほどでVoiceAttackを使用しているときに見つけたことがいくつかあります...

VoiceAttack not listening at all (Level bar on the main screen is not moving):
VoiceAttackがまったく聞こえない(メイン画面のレベルバーが動いていない):

1. Make sure your microphone is plugged in all the way (this gets me just about every time).
1. マイクが完全に差し込まれていることを確認します(これにより、ほぼ毎回接続できます)。

2. Make sure your microphone switch is on. This is sometimes located on the cord of your headset.
2. マイクスイッチがオンになっていることを確認します。これは、ヘッドセットのコードにある場合があります。

3. Make sure the volume of your microphone is adequate. Also, make sure that it is not muted in Windows.
3. マイクの音量が適切であることを確認します。また、Windowsでミュートされていないことを確認してください。

Additional places to look:
その他の見どころ:

If you are using Microsoft Windows Vista/7/8/10, go into Control Panel & run the 'Sound' app.
Microsoft Windows Vista / 7/8/10を使用している場合は、コントロールパネルに移動して「サウンド」アプリを実行します。

Select the recording tab. Select your input device (will have a green checkmark next to it).
記録タブを選択します。入力デバイスを選択します(横に緑色のチェックマークが付きます)。

Go to the levels tab. Make sure the volume is adequate (mine is up all the way). If there is a balance button, click it. Make sure that the microphone channels are at adequate levels (mine are both up all the way).
レベルタブに移動します。ボリュームが十分であることを確認します(私のボリュームがいっぱいになっています)。バランスボタンがある場合は、クリックします。マイクのチャンネルが適切なレベルにあることを確認してください(私の両方が完全にアップしています)。

In XP, go to the 'Sounds and Audio Device Properties' app. Click the 'Audio' tab & make sure the proper device is selected in the drop down list. Click on the 'Volume' button to make adjustments to the volume.
XPでは、「サウンドとオーディオデバイスのプロパティ」アプリに移動します。[オーディオ]タブをクリックし、ドロップダウンリストで適切なデバイスが選択されていることを確認します。「音量」ボタンをクリックして、音量を調整します。

Even more:
さらに:

If you are using Microsoft Windows Vista / 7 / 8 / 10, go into Control Panel and run the 'Speech Recognition Options' app.
Microsoft Windows Vista / 7/8/10を使用している場合は、コントロールパネルに移動して、「音声認識オプション」アプリを実行します。

Select the 'Set up microphone' link. Follow the instructions indicated to set up your
[マイクの設定]リンクを選択します。指示に従って指示に従ってセットアップしてください

microphone.
マイクロフォン。

Make sure to check out the, 'Setting up Microphone Input' later in this document.
このドキュメントで後述する「マイク入力のセットアップ」を必ず確認してください。

VoiceAttack is not understanding commands (Level bar is moving, but, VoiceAttack is not recognizing what I say):
VoiceAttackはコマンドを理解していません(レベルバーは動いていますが、VoiceAttackは私が言ったことを認識していません)。

1. Make sure your microphone levels are adequate as outlined above.
1. 上記で概説したように、マイクのレベルが適切であることを確認します。

2. Make sure that the speech recognition engine is trained to your voice. This sounds a bit tedious, but, it will make a world of difference in how well VoiceAttack can understand you & will add levels of fun to your VoiceAttack experience. Seriously... I can't stress this enough. Go into Control Panel and run the 'Speech Recognition Options' app. Click on the 'Train your computer to better understand you' link. I did it three times. I don't know if doing it that many times was overkill, but, it certainly made things work really well.
2. 音声認識エンジンが自分の声に合わせてトレーニングされていることを確認します。これは少し退屈に聞こえますが、VoiceAttackがあなたをどの程度理解し、VoiceAttack体験に楽しさを追加するかという点で世界に違いをもたらします。真剣に...私はこれを十分に強調することはできません。コントロールパネルに移動し、「音声認識オプション」アプリを実行します。「コンピューターをよりよく理解してトレーニングする」リンクをクリックします。私はそれを3回しました。何度もそれをやるのがやり過ぎだったかどうかはわかりませんが、それは確かに物事を本当にうまく機能させました。

3. Check to see if VoiceAttack is 'Listening'. It's a big button on the main screen on the right side. If it says, 'Not Listening', you need to click it :)
3. VoiceAttackが「リスニング」かどうかを確認します。右側のメイン画面にある大きなボタンです。「Not Listening」と表示されている場合は、クリックする必要があります:)

4. Make sure VoiceAttack is not already running a long or huge macro. You will see indication of this in the recognition log on the main screen (big, white list looking thing). Click the 'Stop Commands' button to halt your macro if you need to.
4. VoiceAttackが既に長いマクロまたは巨大なマクロを実行していないことを確認します。メイン画面の認識ログにこれの表示が表示されます（大きな、ホワイトリストに見えるもの）。必要に応じて、[コマンドの停止]ボタンをクリックしてマクロを停止します。

5. Are you using the right profile? Sometimes I'll switch to a different profile and forget that I did. Switch back over to the right profile :) (Note : this is for non-trial version only. There is only one profile in the trial version.)
5. 適切なプロファイルを使用していますか？時々、私は別のプロファイルに切り替えて、自分がやったことを忘れます。正しいプロファイルに切り替えます:)（注：これは試用版以外のバージョンのみです。試用版にはプロファイルが1つしかありません。）

6. Is your environment fairly quiet? Sometimes your chatty or noisy house mates will confuse VoiceAttack.
6. 環境はかなり静かですか？おしゃべりや騒がしい家の仲間がVoiceAttackを混乱させることがあります。

7. Take a look at the recognition log on the main screen. What does VoiceAttack 'think' you are saying? You might have to make adjustments to your command (given that you have trained up the speech recognition engine as outlined above). Just a note... we have thick accents down here in Texas. Sometimes what I am saying is not recognizable even by other humans :)
7. メイン画面の認識ログを見てください。VoiceAttackはあなたが何を言っていると思いますか？コマンドの調整が必要になる場合があります（上記で概説したように音声認識エンジンをトレーニングした場合）。ちょっと注意してください...ここテキサスでは、太いアクセントがあります。時々私が言っていることは、他の人間によってさえ認識できない:)

8. Out on a limb... make sure you didn't change to a speech engine that you did not train up (you'll find this on the options page). Try switching to 'System Default'.
8. 手足に乗って...訓練していない音声エンジンに変更していないことを確認します（これはオプションページにあります）。「システムのデフォルト」に切り替えてみてください。

9. Even more out on a limb... Make sure you are not running any kind of voice-altering software (you know the kinds that make you sound like an alien or an orc or whatever). Depending on your software, these effects may be passed straight into VoiceAttack, which, in turn, will probably not recognize what is being said.
9. さらに手足に...音声変更ソフトウェアを実行していないことを確認します（エイリアンやオークなどに聞こえる種類を知っています）。お使いのソフトウェアによっては、これらの効果がVoiceAttackに直接渡される場合がありますが、VoiceAttackは発言内容を認識しない可能性があります。

VoiceAttack is recognizing what I am saying (commands are showing up in the log), but, nothing is happening in my game.
VoiceAttackは私が言っていることを認識しています（コマンドはログに表示されます）が、ゲームでは何も起こりません。

Occasionally, you may have to adjust your commands in VoiceAttack to work with certain games. Here are some things you will want to try:
特定のゲームで動作するように、VoiceAttackのコマンドを調整する必要がある場合があります。試してみたいことがいくつかあります：

The most common issue is that VoiceAttack is sending key presses too quickly to your game.
最も一般的な問題は、VoiceAttackがあなたのゲームに早すぎるキーの押下を送信していることです。

If a game is polling for input at a specific time, and, VoiceAttack's key press does not coincide with that polling, the game will simply not catch it. To fix this, you need to increase the amount of time a key is held down before releasing (see, 'Key Press Screen' – 'Hold down for X seconds' option). A good place to start is 0.10 seconds. Try increasing and decreasing this number to see what works best.
ゲームが特定の時間に入力をポーリングしていて、VoiceAttackのキープレスがそのポーリングと一致しない場合、ゲームは単にそれをキャッチしません。これを修正するには、キーを押してから放すまでの時間を長くする必要があります(「キープレス画面」-「X秒ホールド」オプションを参照)。開始するのに適した場所は0.10秒です。この数値を増減してみて、最適な結果を確認してください。

Another common, frustrating issue is that some games or apps will require VoiceAttack to be run in Administrator mode to allow interaction. To run VoiceAttack in Administrator mode, do the following:
もう1つの一般的なイライラする問題は、一部のゲームまたはアプリでは、対話を許可するために管理者モードでVoiceAttackを実行する必要があることです。VoiceAttackを管理者モードで実行するには、次の手順を実行します。

Locate the file called, 'VoiceAttack.exe' (usually in C:¥Program Files (x86)¥VoiceAttack folder). Right-click on the file and select, 'Properties'. Go to the, 'Compatibility' tab and then check the, 'Run as Administrator' box. Click, 'OK' and then run VoiceAttack.
「VoiceAttack.exe」と呼ばれるファイルを見つけます(通常は C:¥ Program Files(x86)¥ VoiceAttackフォルダーにあります)。ファイルを右クリックして、「プロパティ」を選択します。「互換性」タブに移動し、「管理者として実行」ボックスをオンにします。「OK」をクリックしてから、VoiceAttackを実行します。

One last thing to check is to make sure VoiceAttack is sending input to the right place. Verify that your commands are being sent to the proper process, or, to the Active Window (see, 'Voice Attack's Main Screen' – 'Target' option).
最後に確認することは、VoiceAttackが正しい場所に入力を送信していることを確認することです。コマンドが適切なプロセスまたはアクティブウィンドウに送信されていることを確認します(「音声攻撃のメイン画面」-「ターゲット」オプションを参照)。

As always, if none of these methods work, please check out the VoiceAttack user forums :)
いつものように、これらの方法のいずれも機能しない場合は、VoiceAttackユーザーフォーラムをチェックしてください:)

Setting up Microphone Input
マイク入力のセットアップ

Way down here, gleaned from the VoiceAttack User Forums, is a short guide to hopefully help you set up your microphone input for use with VoiceAttack and Windows speech recognition. It just goes over the basic steps that we sometimes miss when setting things up.
ここから先は、VoiceAttackユーザーフォーラムから収集した、VoiceAttackおよびWindows音声認識で使用するマイク入力のセットアップに役立つ短いガイドです。設定の際に見逃してしまうことがある基本的な手順について説明しています。

Audio input and output in Windows is somewhat of a maze, but makes sense after you learn your way around. As with everything, you set something once and then forget how it's done. This is here for both you and me :)
Windowsでのオーディオ入出力はやや迷路ですが、方法を学んだ後は理にかなっています。すべてと同様に、何かを一度設定してから、それがどのように行われたかを忘れます。これはあなたと私の両方のためにここにあります:)

Firstly, VoiceAttack uses Windows' built-in speech recognition to do its thing. Out of the box, the speech engine uses the default recording device to receive input. The recording devices can be accessed by right-clicking on the, 'Speakers' icon in the system tray and selecting, 'Recording Devices', or, by right clicking on the VoiceAttack icon in the task bar and selecting, 'Recording Devices'. Right-clicking on the desired microphone/headset and selecting, 'Set as default device' will do just that. You will know your device is selected as default by the green check mark appearing over the device's icon. This works for most, since we've usually only got one device on our machines that we primarily use.
まず、VoiceAttackはWindowsに組み込まれている音声認識を使用して処理を行います。デフォルトでは、音声エンジンはデフォルトの録音デバイスを使用して入力を受け取ります。録音デバイスにアクセスするには、システムトレイの「スピーカー」アイコンを右クリックして「録音デバイス」を選択するか、タスクバーのVoiceAttackアイコンを右クリックして「録音デバイス」を選択します。目的のマイク/ヘッドセットを右クリックして[デフォルトのデバイスとして設定]を選択すると、それが実行されます。デバイスのアイコンの上に緑色のチェックマークが表示され、デバイスがデフォルトとして選択されていることがわかります。これはほとんどの場合に機能します。通常、マシンに使用するデバイスは主に1つだけであるためです。

For those that have multiple microphones/headsets/webcams, the default device is sometimes not what you want selected. To change the recording device that the speech engine uses, just open up Control Panel and choose the Speech Recognition applet, or, right- click on the VoiceAttack icon in the task bar and select, 'Speech Control Panel'. Click on the, 'Advanced speech options' link and then click on the button at the bottom labeled, 'Advanced...'. This opens up the, 'Audio Input Settings' dialog. There are two choices in this dialog: 'Use preferred audio input device' and 'Use this audio input device'. The first choice allows you to continue to use the default input device (whatever that device might be).
複数のマイク/ヘッドセット/ウェブカメラを搭載しているデバイスでは、デフォルトのデバイスが選択したものと異なる場合があります。音声エンジンが使用する録音デバイスを変更するには、コントロールパネルを開いて音声認識アプレットを選択するか、タスクバーのVoiceAttackアイコンを右クリックして[音声コントロールパネル]を選択します。[詳細な音声オプション]リンクをクリックし、下部にある[詳細...]というラベルのボタンをクリックします。これにより、「オーディオ入力設定」ダイアログが開きます。このダイアログには、「優先するオーディオ入力デバイスを使用する」と「このオーディオ入力デバイスを使用する」の2つの選択肢があります。最初の選択肢では、デフォルトの入力デバイス（そのデバイスが何であれ）を引き続き使用できます。

Choosing, 'Use this audio device' lets you choose a specific device from the dropdown (default recording device is ignored).
[このオーディオデバイスを使用する]を選択すると、ドロップダウンから特定のデバイスを選択できます（デフォルトの録音デバイスは無視されます）。

VoiceAttack's Data Storage
VoiceAttackのデータストレージ

VoiceAttack stores its data in a single file named, 'VoiceAttack.dat'. Each user on your PC will have their own VoiceAttack.dat file that will be stored (usually) in "C:¥Users¥YOUR_USER_NAME¥AppData¥Roaming¥VoiceAttack". If you want to back up your VoiceAttack data in one big chunk, the VoiceAttack.dat file is the one you will want to save.
VoiceAttackは、データを「VoiceAttack.dat」という名前の単一のファイルに保存します。PCの各ユーザーには、「通常：」「C：¥ Users ¥ YOUR_USER_NAME ¥ AppData ¥ Roaming ¥ VoiceAttack」に保存されるVoiceAttack.datファイルがあります。VoiceAttackデータを1つの大きなチャンクにバックアップする場合は、VoiceAttack.datファイルを保存する必要があります。

Accessing this folder for the first time may be tricky, as Windows may have these folders hidden. What you will want to do (if you haven't already done so) is to have Windows show hidden files and folders: https://support.microsoft.com/en-us/help/14201/windows-show- hidden-files Another (probably quicker) way to browse this folder is by clicking on the link provided on the Options screen, under the, 'System/Advanced' tab labeled, 'Click here to browse VoiceAttack's data folder'.
Windowsでこれらのフォルダーが非表示になっている場合があるため、このフォルダーに初めてアクセスするのは難しい場合があります。(まだ行っていない場合)行うことは、Windowsで非表示のファイルとフォルダーを表示することです:https://support.microsoft.com/en-us/help/14201/windows-show- hidden-ファイルこのフォルダーを参照する別の(おそらくより速い)方法は、「システム/詳細設定」タブの下にある「VoiceAttackのデータフォルダーを参照するにはここをクリック」というオプション画面にあるリンクをクリックすることです。

On a side note, and to clear up some confusion, VoiceAttack's data is NOT stored in .VAP files. .VAP files are the files that are created if you decide to export a single VoiceAttack profile (to share or simply as a backup). You may have had to import a .VAP file at one time or another. The information contained in the .VAP file that you imported is merged into your VoiceAttack.dat file and saved.
補足として、そして混乱を解消するために、VoiceAttackのデータは.VAPファイルに保存されません。.VAPファイルは、単一のVoiceAttackプロファイルをエクスポートする(共有するため、または単にバックアップとして)場合に作成されるファイルです。.VAPファイルを一度にインポートする必要があったかもしれません。インポートした.VAPファイルに含まれる情報は、VoiceAttack.datファイルにマージされて保存されます。

VoiceAttack Author Flags
VoiceAttack作成者フラグ

There may be times when you need a finer level of control over certain aspects of your profiles when it comes to releasing them out into the community. Below are a few flags that can be altered within commands and profiles to help shape the delivery of the content you create. Currently, there is no user interface to edit these items. You will first need to export your profiles as a standard .VAP that is not compressed (see, 'Exporting Profiles' earlier in this document). You must then use a text editor (like Notepad) to modify the XML data, save the .VAP and then re-import the profile back into VoiceAttack.
プロフィールをコミュニティにリリースする際に、プロフィールの特定の側面をより細かく制御する必要がある場合があります。以下は、作成するコンテンツの配信を形作るのに役立つコマンドとプロファイル内で変更できるいくつかのフラグです。現在、これらのアイテムを編集するためのユーザーインターフェイスはありません。最初に、圧縮されていない標準の.VAPとしてプロファイルをエクスポートする必要があります(このドキュメントの「プロファイルのエクスポート」を参照)。次に、テキストエディター(メモ帳など)を使用してXMLデータを変更し、.VAPを保存してから、プロファイルをVoiceAttackに再インポートする必要があります。

Note: If the intent of using the supplied author flags is for obfuscation purposes, please be advised that VoiceAttack.com does not accept any responsibility for the use of these flags, as the use of any/all of the flags is solely up to the author to employ them. As we are all well aware, any attempt to obfuscate any type of data is an open invitation for anybody to defeat that obfuscation, so no guarantee is made that your profiles and commands will be completely hidden.
注:提供された作成者フラグを使用する意図が難読化を目的としている場合、VoiceAttack.comはこれらのフラグの使用について一切の責任を負わないことにご注意ください。それらを採用する著者。誰もが知っているように、あらゆる種類のデータを難読化しようとする試みは、誰もがその難読化を打ち負かすための開かれた招待状なので、あなたのプロファイルとコマンドが完全に隠される保証はありません。

Profile-level flags
プロファイルレベルのフラグ

Inside the XML of the .VAP, within the Profile element, you will find a few elements that can only be modified outside of VoiceAttack. In the section above titled, 'VoiceAttack Profile Package Reference', we went over the <IO> and <IS> elements when creating a profile package (you can find out more about those in that section). In addition to those flags, the Profile element also contains a few more:
.VAPのXML内のProfile要素内には、VoiceAttackの外部でのみ変更可能ないくつかの要素があります。「VoiceAttackプロファイルパッケージリファレンス」というタイトルのセクションでは、プロファイルパッケージを作成するときに<IO>要素と<IS>要素について説明しました(これらの詳細については、このセクションを参照してください)。これらのフラグに加えて、Profile要素にはさらにいくつかが含まれています。

<IP> - The IP element will let you indicate to your users how you wish to manage the importing of your individual commands into other profiles. This is handy if you have commands in the profile you designed that are dependent on each other and allowing the commands to be imported individually into other profiles would cause those commands to not work (and, basically create a bottomless support pit). You can specify that a warning is displayed, or, a full stop message. To display a warning to indicate that individual command imports may not work on their own, and that you recommend importing the entire profile, change the IP element to '1': <IP>1</IP>. To prevent individual commands from being imported into other profiles at all, change the IP element to, '2': <IP>2</IP>. To disable, simply change or leave the value as, <IP>0</IP>. Probably not something you will use a lot, but it's there if you need it.
<IP> -IPエレメントを使用すると、個々のコマンドを他のプロファイルにインポートする方法をユーザーに指示できます。これは、設計したプロファイルに相互に依存するコマンドがあり、それらのコマンドを他のプロファイルに個別にインポートできるようにすると、それらのコマンドが機能しなくなる(そして、基本的に底なしのサポートピットを作成する)場合に便利です。警告を表示するか、完全停止メッセージを表示するかを指定できます。個々のコマンドのインポートが単独では機能しない可能性があり、プロファイル全体のインポートを推奨することを示す警告を表示するには、IP要素を「1」に変更します: <IP> 1 </ IP> 。個々のコマンドが他のプロファイルにまったくインポートされないようにするには、IP要素を「2」に変更します: <IP> 2 <。無効にするには、値を<IP> 0 </ IP>に変更するかそのままにしてください。おそらくあなたがたくさん使うものではありませんが、必要ならそこにあります。

<BE> - The BE element indicates that the profile can only be exported from VoiceAttack as compressed binary. To turn this flag on, simply set the BE element to, '1': <BE>1</BE>. To disable, simply change or leave the value as, <BE>0</BE>. Note that if you distribute your profile initially as a compressed binary and this flag is set to 1, there is no provided way to change that profile back. So, if you plan on distributing a compressed binary with this flag set, you will need to maintain at least two versions of your profile. A version that is used for authoring, and a version that will be used for distribution.
<BE> -BE 要素は、プロファイルがVoiceAttackから圧縮バイナリとしてのみエクスポートできることを示します。このフラグをオンにするには、BE要素を「1」に設定するだけです:<BE> 1 </ BE>。無効にするには、値を変更するか、<BE> 0 </ BE>のままにします。最初にプロファイルを圧縮バイナリとして配布し、このフラグが1に設定されている場合、そのプロファイルを元に戻す方法は提供されていないことに注意してください。したがって、このフラグを設定して圧縮バイナリを配布する予定がある場合は、プロファイルの少なくとも2つのバージョンを維持する必要があります。オーサリングに使用されるバージョン、および配布に使用されるバージョン。

<AuthorTag1>, <AuthorTag2>, <AuthorTag3> - The AuthorTag elements will allow you to indicate whatever you would like as text values that persist through export. These values can be retrieved by calling the proxy methods, 'Profile.AuthorTag1()', 'Profile.AuthorTag2()' and 'Profile.AuthorTag3()'. See, 'VoiceAttack Plugins (for the truly mad)' earlier in this document.
<AuthorTag1>、<AuthorTag2>、<AuthorTag3> -AuthorTag要素を使用すると、エクスポート中に保持されるテキスト値として何でも指定できます。これらの値は、プロキシメソッド「Profile.AuthorTag1()」、「Profile.AuthorTag2()」、および「Profile.AuthorTag3()」を呼び出すことで取得できます。このドキュメントで前述した「VoiceAttackプラグイン(本当に狂った人向け)」を参照してください。

<InternalID> – The InternalID element is a GUID flag for use by authors to be able to identify a profile. This value can be retrieved by calling the proxy method, ‘Profile.InternalID()’. See, ‘VoiceAttack Plugins (for the truly mad)’ earlier in this document.
<InternalID> –InternalID要素は、作成者がプロファイルを識別できるようにするためのGUIDフラグです。この値は、プロキシメソッド「Profile.InternalID()」を呼び出すことで取得できます。このドキュメントで前述した「VoiceAttackプラグイン（本当に狂った人向け）」を参照してください。

<AuthorID> – The AuthorID element is a GUID flag for use by authors to indicate the creator of the profile. This value can be retrieved by calling the proxy method, ‘Profile.AuthorID()’.
<AuthorID> –AuthorID要素は、プロファイルの作成者を示すために作成者が使用するGUIDフラグです。この値は、プロキシメソッド「Profile.AuthorID()」を呼び出すことで取得できます。

See, ‘VoiceAttack Plugins (for the truly mad)’ earlier in this document.
このドキュメントで前述した「VoiceAttackプラグイン（本当に狂った人向け）」を参照してください。

<ProductID> – The ProductID element is a GUID flag for your use by authors to distinctly identify profiles they create. This value can be retrieved by calling the proxy method, ‘Profile.ProductID()’. See, ‘VoiceAttack Plugins (for the truly mad)’ earlier in this document.
<ProductID> –ProductID要素は、作成者が作成するプロファイルを明確に識別するために作成者が使用するGUIDフラグです。この値は、プロキシメソッド「Profile.ProductID()」を呼び出すことで取得できます。このドキュメントで前述した「VoiceAttackプラグイン（本当に狂った人向け）」を参照してください。

<CR> – The CR element will allow you to restrict the end user from adding new commands to a profile (this also includes importing commands). This is primarily for protection against data loss in the event of profiles being overwritten, and encourage end users to, ‘include’ the profile rather than modifying it directly. Simply set this element to 1 to turn it on: <CR>1</CR>.
<CR> –CR要素により、エンドユーザーがプロファイルに新しいコマンドを追加することを制限できます（これにはコマンドのインポートも含まれます）。これは主に、プロファイルが上書きされた場合のデータ損失に対する保護のためであり、エンドユーザーにプロファイルを直接変更するのではなく「含める」ことを奨励します。この要素を1に設定してオンにします：<CR> 1 </ CR> 。

<CRM> – The CRM element works in conjunction with the <CR> element above. This will allow you to specify your own message about why commands are not allowed to be added. For instance, you may want to protect the end user from adding commands to prevent data loss due to a profile overwrite. The default message is, ‘The author of this profile has indicated that new commands are prohibited.’. <CRM>Commands may not be added to this profile, as this profile should only be, ‘included’ and never modified. </CRM>
<CRM> –CRM要素は、上記の<CR>要素と連動します。これにより、コマンドの追加が許可されない理由に関する独自のメッセージを指定できます。たとえば、コマンドの追加からエンドユーザーを保護して、プロファイルの上書きによるデータ損失を防ぐことができます。デフォルトのメッセージは、「このプロファイルの作成者は、新しいコマンドが禁止されていることを示しています。」です。<CRM>コマンドはこのプロファイルに追加できません。このプロファイルは「含める」だけで、変更しないでください。</ CRM>

<CLM> - The CLM element at the profile level serves as a blank message in case you do not want the overhead of indicating a CLM element for every command that is locked. Note that you can override this value at the command level (see <CLM> in the Command-level flags section below). <CLM>Commands in this profile are locked for your own protection.
<CLM> -プロファイルレベルのCLM要素は、ロックされているすべてのコマンドのCLM要素を示すオーバーヘッドが必要ない場合に、空白のメッセージとして機能します。コマンドレベルでこの値を上書きできることに注意してください（以下のコマンドレベルフラグセクションの<CLM>を参照）。<CLM>このプロファイルのコマンドは、ユーザー自身の保護のためにロックされています。

Carry on.</CLM>
続けて。</ CLM>

<PD> - The PD element will allow you to restrict the end user from duplicating your profile. Simply set this element to 1 to turn it on: <PD>1</PD>.
<PD> -PD要素を使用すると、エンドユーザーによるプロファイルの複製を制限できます。この要素を1に設定してオンにします： <PD> 1 </ PD> 。

<PR> <CO> <CV><OP> - The PR, CO, CV and OP elements are for future use.
<PR> <CO> <CV> <OP> -PR、CO、CV、およびOP要素は将来使用するためのものです。

Command-level flags
コマンドレベルのフラグ

Below are the available command-level flags that you can use. Again, you'll only be able to access this flag by editing an uncompressed .VAP file with a text editor.
以下は、使用可能なコマンドレベルのフラグです。繰り返しますが、このフラグにアクセスするには、非圧縮の.VAPファイルをテキストエディターで編集する必要があります。

<InternalID> - The InternalID element will allow you to specify your own GUID value to identify an individual command. This value can be accessed from the proxy method Command.InternalID(). See, 'VoiceAttack Plugins (for the truly mad)' earlier in this document.
<InternalID> -InternalID要素を使用すると、独自のGUID値を指定して個々のコマンドを識別できます。この値には、プロキシメソッドCommand.InternalID（）からアクセスできます。このドキュメントで前述した「VoiceAttackプラグイン（本当に狂った人向け）」を参照してください。

<CL> – The CL element will allow you to lock an individual command's contents from being accessed on its own (edited, viewed, duplicated, copy/pasted, copied to other profiles or imported into other profiles). This is handy for commands that are complex and allowing access would probably open up support issues, or, maybe you just don't want to share your work with everyone. The type of command you would use with this flag would most likely be a subcommand, as your users would not be able to edit any aspect of it (spoken phrase, hotkeys, category, description, etc.). Set this element to '1' to turn it on: <CL>1</CL>.
<CL> –CLエレメントを使用すると、個々のコマンドのコンテンツが単独でアクセスされるのをロックできます(編集、表示、複製、コピー/貼り付け、他のプロファイルへのコピー、または他のプロファイルへのインポート)。これは複雑なコマンドの場合に便利で、アクセスを許可するとサポートの問題が発生する可能性があります。または、作業を全員と共有したくない場合があります。このフラグで使用するコマンドのタイプは、ほとんどの場合サブコマンドになります。ユーザーがコマンドのどの部分も編集できないためです(話し言葉、ホットキー、カテゴリ、説明など)。この要素を「1」に設定してオンにします:
<CL> 1 </ CL> 。

<CLM> – The CLM element works in conjunction with the <CL> element above. This will allow you to specify your own message about why the command is locked. For instance, you may want to protect the end user from changing details about the command due to an impact on performance or loss of functionality. The default message is, 'The author of this profile has indicated that the content of the selected command is locked and cannot be accessed.'. Note that this message overrides the CLM element indicated at the profile level (see above).
<CLM> –CLM要素は、上記の<CL>要素と連動します。これにより、コマンドがロックされる理由に関する独自のメッセージを指定できます。たとえば、パフォーマンスへの影響や機能の損失により、コマンドに関する詳細が変更されないようにエンドユーザーを保護することができます。デフォルトのメッセージは、「このプロファイルの作成者は、選択したコマンドのコンテンツがロックされており、アクセスできないことを示しています。」です。このメッセージは、プロファイルレベルで示されたCLM要素を上書きすることに注意してください(上記を参照)。

<CLM>This command is locked for your own protection. Carry on.</CLM>
<CLM>このコマンドは、ユーザー自身の保護のためにロックされています。続けて。</ CLM>

Note that if your profiles are distributed uncompressed, or, compressed and not using the <BE> flag (above) set to, '1', the end-user will still be able to modify these flags, so, plan accordingly.
プロファイルが非圧縮で配布されている場合、または圧縮されており、<BE>フラグ(上記)を「1」に設定して使用していない場合、エンドユーザーはこれらのフラグを変更できるため、計画を立ててください。

For fun... maybe
楽しみのために...多分

Welcome to the end of the help document. Thanks for reading my mishmash that´s been growing for several years :) Some extra things thrown in, just for fun (or not) will be down here as I add them (or recall that I added them and neglected to document).
ヘルプドキュメントの最後にようこそ。数年にわたって成長している私のミッシュマッシュを読んでくれてありがとう:)ただ追加する(または追加して文書化を怠ったことを思い出してください)

If you right-click on the VoiceAttack icon in the top-left corner of the main screen, you'll see an option for, 'Cover of Darkness'. Checking this option this puts the main screen in, 'dark mode', which might be useful when using VoiceAttack at night. To go back, just click on the menu item again (note that this used to be a command line parameter, but that parameter has been removed).
メイン画面の左上隅にあるVoiceAttackアイコンを右クリックすると、' Cover of Darkness 'のオプションが表示されます。このオプションをオンにすると、メイン画面が「ダークモード」になります。これは、夜間にVoiceAttackを使用する場合に便利です。戻るには、メニュー項目をもう一度クリックします（これは以前はコマンドラインパラメーターでしたが、そのパラメーターは削除されていることに注意してください）。

Command line parameter, '-opacity' was added as a test. Passing a value of 0 to 100 affects the main screen's opacity level. 100 = not transparent at all, 0 = fully transparent. Example: - opacity 75 sets the opacity at 75%.
コマンドラインパラメータ ' -opacity 'がテストとして追加されました。0～100の値を渡すと、メイン画面の不透明度レベルに影響します。100 =まったく透明ではない、0 =完全に透明。例：-不透明度75は、不透明度を75％に設定します。

To override the default on/off sounds in VoiceAttack (listening on/off, joysticks on/off, etc.), just add a valid .wav file called, 'sys on.wav' and/or a file called, 'sys off.wav' to the same directory that VoiceAttack.exe is located (usually C:¥Program Files (x86)¥VoiceAttack). If you want to override the default interrupt sound (stop all commands.), just add a file called, 'sys stop.wav'. Note that these must be valid .wav files. If an error is encountered, the sounds are reverted back to the default sounds. Note also that these sounds will override any selected sounds on the options screen.
VoiceAttackでオン/オフ音にデフォルトを上書きするには、単に、「と呼ばれる有効な.wavファイルを追加、（、オン/オフジョイスティック/オフなどをリッスン）sys on.wav」および/またはと呼ばれるファイル' sys off.wavを'VoiceAttack.exeと同じディレクトリ（通常は C：¥ Program Files（x86）¥ VoiceAttack）に移動します。デフォルトの割り込み音を無効にする（すべてのコマンドを停止する）場合は、「sys stop.wav」というファイルを追加するだけです。これらは有効な.wavファイルでなければならないことに注意してください。エラーが発生した場合、音はデフォルトの音に戻ります。また、これらの音はオプション画面で選択された音を上書きすることに注意してください。

If you want to back up your VoiceAttack.dat file (the file that holds ALL your profiles), it is usually located in C:¥Users¥YOUR_USER_NAME¥AppData¥Roaming¥VoiceAttack.
VoiceAttack.datファイル（すべてのプロファイルを保持するファイル）をバックアップする場合、通常はC：¥ Users ¥ YOUR_USER_NAME ¥ AppData ¥ Roaming ¥ VoiceAttackにあります。

The, 'Backup' directory located in C:¥Users¥YOUR_USER_NAME¥AppData¥Roaming¥VoiceAttack holds up to the last ten changes made to your VoiceAttack profiles. To roll back to a previous change, simply move any one of the files out of the Backup directory up into the C:¥Users¥YOUR_USER_NAME¥AppData¥Roaming¥VoiceAttack directory and replace the current VoiceAttack.dat. You'll probably never ever use this, but it's there if you need it ;)
C：¥ Users ¥ YOUR_USER_NAME ¥ AppData ¥ Roaming ¥ VoiceAttackにある「バックアップ」ディレクトリは、VoiceAttackプロファイルに加えられた最後の10個の変更まで保持します。前の変更にロールバックするには、いずれかのファイルをバックアップディレクトリからC：¥ Users ¥ YOUR_USER_NAME ¥ AppData ¥ Roaming ¥ VoiceAttackディレクトリに移動し、現在のVoiceAttack.datを置き換えます。おそらくこれを使用することはないでしょうが、必要な場合はそこにあります;)