



Silvia Core

Contents

Introduction to Silvia Core	7
Creating an Android Application	7
Prerequisites	7
Installing the Java Development Kit (JDK).....	7
Installing the Android SDK	8
1. Acquiring the SDK Manager	8
2. Acquire the Correct Tools.....	9
Preparing the Target Android Device for Development	10
Creating an Application in Unity	13
Downloading and Installing Unity	13
Setting up the Unity Editor for Android Development	14
Deployment Test	18
SILVIA API	20
silvia.app	20
Functions	22
silvia.app.setVoiceFont	22
silvia.app.setVoiceType	24
silvia.app.getVoiceType.....	26
silvia.app.getVoiceFontGender	27
silvia.app.getVoiceFontName	27
silvia.app.getVoiceFontRate.....	28
silvia.app.loadVisemes	29
silvia.app.loadVisemesCrop	30
silvia.app.getVisemesFolder.....	31
silvia.app.enableVoiceOutput	32
silvia.app.voiceOutputEnabled	33
silvia.app.enableTextOutput	35
silvia.app.textOutputEnabled	37
silvia.app.enableSocketOutput	37
silvia.app.socketOutputEnabled	39
silvia.app.enableDiagOutput.....	41
silvia.app.diagOutputEnabled	41
silvia.app.enableApplicationMessage	42
silvia.app.applicationMessageEnabled	43

silvia.app.setVoiceOutput	45
silvia.app.getVoiceOutput	45
silvia.app.clearVoiceOutput	46
silvia.app.isSpeaking	48
silvia.app.setIsSpeaking	48
silvia.app.setTextOutput	50
silvia.app.getTextOutput	51
silvia.app.clearTextOutput	52
silvia.app.setSocketOutput	53
silvia.app.setDiagOutput	54
silvia.app.getDiagOutput	56
silvia.app.clearDiagOutput	56
silvia.app.setApplicationMessage	57
silvia.app.getApplicationMessage	58
silvia.app.clearApplicationMessage	59
silvia.app.setListening	59
silvia.app.isListening	61
silvia.app.consoleOut	62
silvia.brain	63
Functions	67
silvia.brain.setUserSecurityLevel	67
silvia.brain.getUserSecurityLevel	68
silvia.brain.addConcepts	69
silvia.brain.getResponse	70
silvia.brain.getResponseBehaviorID	72
silvia.brain.getResponseAbsorberID	73
silvia.brain.getResponseWeight	73
silvia.brain.transformNarrativeMode	74
silvia.brain.executePostEvents	75
silvia.brain.setJump	75
silvia.brain.setBypassResponse	77
silvia.brain.removeStopwords	78
silvia.brain.setDynamicAttraction	79
silvia.brain.setDynamicDepth	79
silvia.brain.setDynamicFalloffDepth	81
silvia.brain.setDynamicFalloff	83

silvia.brain.setDynamicAdaptation	85
silvia.brain.generateDynamic.....	87
silvia.brain.generateDynamicLimited	89
silvia.brain.generateDynamicFromMemory	90
silvia.brain.dynamicHasAllConceptsInOne.....	91
silvia.brain.setAbsorberThreshold	92
silvia.brain.setReusableThreshold	93
silvia.brain.setAddressByName.....	94
silvia.brain.getAddressByName	94
silvia.brain.setAllEars	96
silvia.brain.getAllEars	98
silvia.brain.loadCommandAssembly	98
silvia.data	99
Functions	105
silvia.data.clear.....	105
silvia.data.setBinding	107
silvia.data.setBindingTypeModifier	109
silvia.data.getBindingTypeModifier	110
silvia.data.testBinding	113
silvia.data.getBoundConcept	115
silvia.data.getAllBoundConcepts	118
silvia.data.tuneConcepts	119
silvia.data.cleanConcepts.....	120
silvia.data.getBehaviorID	120
silvia.data.getBehaviorIDFromIndex	121
silvia.data.getBehaviorCount	123
silvia.data.getBehaviorGroup.....	123
silvia.data.getBehaviorSubGroup.....	125
silvia.data.getBehaviorName	125
silvia.data.removeBehavior.....	127
silvia.data.addAbsorber	127
silvia.data.setAbsorberText.....	129
silvia.data.getAbsorberText	130
silvia.data.removeAbsorber	131
silvia.data.addExuder	133
silvia.data.setExuderText	133

silvia.data.getExuderText	134
silvia.data.removeExuder	135
silvia.data.setBehaviorScript	136
silvia.data.setExuderscript	138
silvia.data.setBehaviorData	139
silvia.data.getBehaviorData	141
silvia.data.setBehaviorSecurityLevel	141
silvia.data.getBehaviorSecurityLevel	143
silvia.data.setAbsorberData	143
silvia.data.getAbsorberData	145
silvia.data.setExuderData	146
silvia.data.getExuderData	147
silvia.data.setExudersecurityLevel	147
silvia.data.getExudersecurityLevel	148
silvia.data.setAbsorberExact	149
silvia.data.getAbsorberExact	149
silvia.data.setAbsorberReject	151
silvia.data.getAbsorberReject	152
silvia.data.setExuderExact	153
silvia.data.getExuderExact	154
silvia.data.setExuderReuse	154
silvia.data.getExuderReuse	156
silvia.data.setExuderDynamic	157
silvia.data.getExuderDynamic	158
silvia.data.setExuderContext	159
silvia.data.getExuderContext	159
silvia.data.getBehaviorCreatedYear	160
silvia.data.getBehaviorCreatedMonth	162
silvia.data.getBehaviorCreatedDay	162
silvia.data.getBehaviorCreatedHour	164
silvia.data.getBehaviorCreatedMinute	164
silvia.data.getBehaviorLastModifiedYear	165
silvia.data.getBehaviorLastModifiedMonth	165
silvia.data.getBehaviorLastModifiedDay	166
silvia.data.getBehaviorLastModifiedHour	167
silvia.data.getBehaviorLastModifiedMinute	167

silvia.data.addResponse.....	169
silvia.data.setContext.....	169
silvia.data.setReuse.....	170
silvia.data.setBehaviorName	171
silvia.data.setBehaviorGroup	171
silvia.data.setBehaviorSubGroup	173
silvia.feedback	173
Functions	175
silvia.feedback.setActive.....	175
silvia.feedback.isActive	177
silvia.feedback.clarify	179
silvia.feedback.addFeedback	179
silvia.feedback.suggestFeedback	181
silvia.feedback.setInterval	182
silvia.feedback.setThreshold.....	183
silvia.feedback.setProbability	185
silvia.feedback.pause	185
silvia.feedback.resume.....	187
silvia.feedback.search	189
silvia.feedback.searchRemoveFromStack.....	190
silvia.feedback.searchGetString.....	192
silvia.feedback.searchGetTime	192
silvia.feedback.searchNext.....	194
silvia.feedback.write	195
silvia.feedback.read	195
silvia.mem	196
Functions	198
silvia.mem.load	198

Introduction to Silvia Core

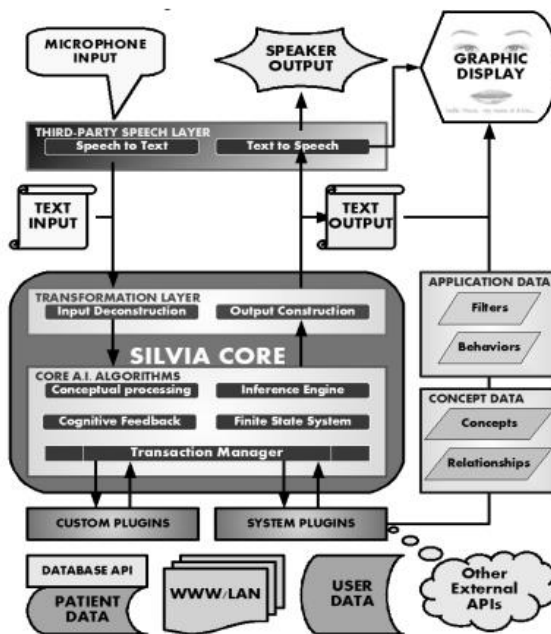
Silvia Core is the heart of all Silvia applications. Core drives Silvia Studio, Silvia Server, as well as third party applications with embedded Silvia. Silvia Core is a single .dll file that is capable of use in any operating system with support for Mono. Mono is a software platform designed to allow developers to easily create cross platform applications as part of the .NET Foundation. For example, Silvia Core can be embedded in an Android, iOS, or Unity application to provide Conversational AI to an existing application. Silvia Core is meant to be for single use, that is, one conversation at a time. For a multithreaded, multiple user environment, use Silvia Server.

Silvia Studio is essentially an external application using all of the Core API functions to become an application builder for Silvia technology. Likewise, Silvia Server is a wrapper around Core that scales and juggles thousands of simultaneous requests.

SILVIA Application Architecture Example



- SILVIA AI CORE
- Enabling Tech
- SILVIA CORE Data
 - Filters
 - Behaviors
 - Concepts
- Plugins
- External Interfaces
- User Data



As a fully functional, independent, small-sized conversational AI engine, it can accommodate inclusion in very small and very large spaces. The API, included at the end of this manual, is fully functional across all platforms, regardless of size. The potential uses for Silvia Core are only limited by imagination.

Some of the most common uses for Core have been for iOS, Android, and Unity applications. This manual will help you get started with both Android and Unity applications.

Creating an Android Application

Prerequisites

Installing the Java Development Kit (JDK)

Android applications require the Java Development Kit, or JDK:

1. Download the most current JDK for your platform.

2. Accept the License Agreement and select your operating system to download the JDK (See below).
3. While installing, **keep all of the install directories to their defaults.**

Java SE Development Kit 8u181

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☒ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.95 MB	jdk-8u181-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.89 MB	jdk-8u181-linux-arm64-vfp-hflt.tar.gz
Linux x86	165.06 MB	jdk-8u181-linux-i586.rpm
Linux x86	179.87 MB	jdk-8u181-linux-i586.tar.gz
Linux x64	162.15 MB	jdk-8u181-linux-x64.rpm
Linux x64	177.05 MB	jdk-8u181-linux-x64.tar.gz
Mac OS X x64	242.83 MB	jdk-8u181-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.17 MB	jdk-8u181-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.34 MB	jdk-8u181-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.83 MB	jdk-8u181-solaris-x64.tar.Z
Solaris x64	92.11 MB	jdk-8u181-solaris-x64.tar.gz
Windows x86	194.41 MB	jdk-8u181-windows-i586.exe
Windows x64	202.73 MB	jdk-8u181-windows-x64.exe

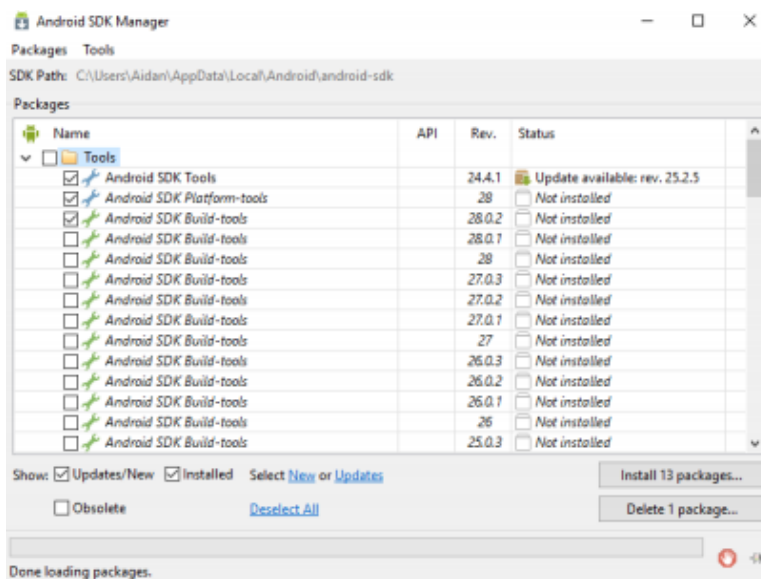
Installing the Android SDK

Note: There are pitfalls to overcome when installing the SDK. The methods used in this guide may slightly stray from the 'recommended' installation routes, but should save a significant amount of time.

1. Acquiring the SDK Manager

The SDK Manager allows downloads of all of the essential build-tools and SDK versions. Usually, this tool is tucked away in an Android Studio installation, but with the following methods, we can install a stand-alone version of the SDK manager without installing Android Studio.







- Download the stand-alone SDK Manager from Google.
- Again, **keep all of the installation directories to their defaults.**
- Proceed with the installation and open the SDK manager once finished. You should be presented with a window that looks like this:













2. Acquire the Correct Tools

Select the platform tools. Note: there are several ambiguous options. Pay attention to the categories of each option.













- i) In the Tools folder, select the top-most options: Android SDK Tools, Android SDK Platform-tools, and Android SDK Build-tools:

Packages			
 Name	API	Rev.	Status
<input checked="" type="checkbox"/>  Tools			
<input checked="" type="checkbox"/>  Android SDK Tools		24.4.1	 Update available: rev. 25.2.5
<input checked="" type="checkbox"/>  Android SDK Platform-tools		28	<input type="checkbox"/> Not installed
<input checked="" type="checkbox"/>  Android SDK Build-tools		28.0.2	<input type="checkbox"/> Not installed

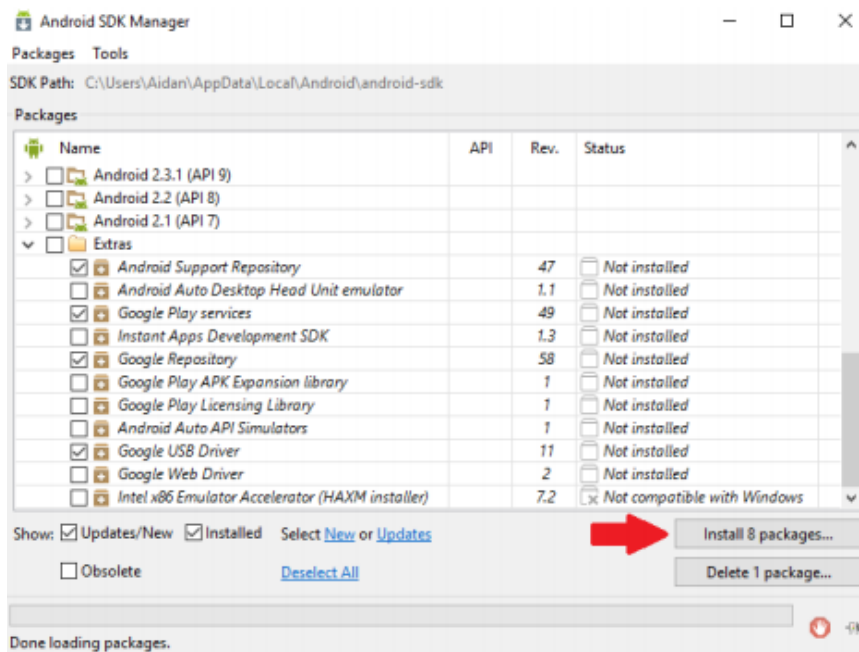
- ii) Next, select your Android API level. It is recommended to uncheck any “preview” labeled folders and stick with the latest stable release of the API.
- iii) **Locate the latest non-preview version of the Android API. At the time of writing this guide, it is Android 9 (API 28).** Note: Unless you require an emulator, uncheck all of the system images. These files are very large and are not required.

<input checked="" type="checkbox"/>  Android 9 (API 28)			
 <input checked="" type="checkbox"/>  SDK Platform	28	6	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android TV Intel x86 Atom System Image	28	5	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Intel x86 Atom_64 System Image	28	4	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Intel x86 Atom System Image	28	4	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google Play Intel x86 Atom_64 System Image	28	5	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google Play Intel x86 Atom System Image	28	5	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google APIs Intel x86 Atom_64 System Image	28	5	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google APIs Intel x86 Atom System Image	28	5	<input type="checkbox"/> Not installed

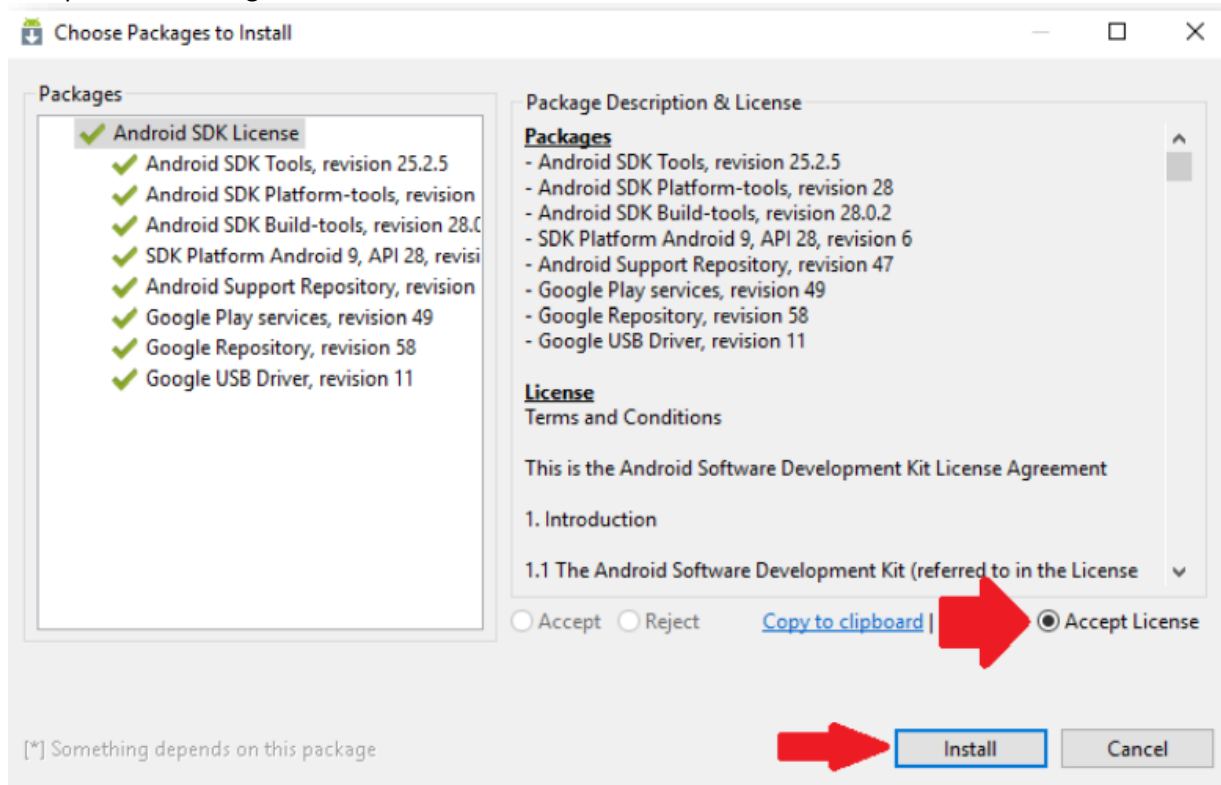
- iv) Finally, in the Extras folder, select: Android Support Repository, Google Repository, Google Play Services, and Google USB Driver:

<input checked="" type="checkbox"/>  Extras			
<input checked="" type="checkbox"/>  Android Support Repository	47		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android Auto Desktop Head Unit emulator	1.1		<input type="checkbox"/> Not installed
<input checked="" type="checkbox"/>  Google Play services	49		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Instant Apps Development SDK	1.3		<input type="checkbox"/> Not installed
<input checked="" type="checkbox"/>  Google Repository	58		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google Play APK Expansion library	1		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google Play Licensing Library	1		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android Auto API Simulators	1		<input type="checkbox"/> Not installed
<input checked="" type="checkbox"/>  Google USB Driver	11		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google Web Driver	2		<input type="checkbox"/> Not installed
<input type="checkbox"/>  Intel x86 Emulator Accelerator (HAXM installer)	7.2		<input checked="" type="checkbox"/> Not compatible with Windows

- b) Install the SDK Manager:



c) Accept the license agreement and install.



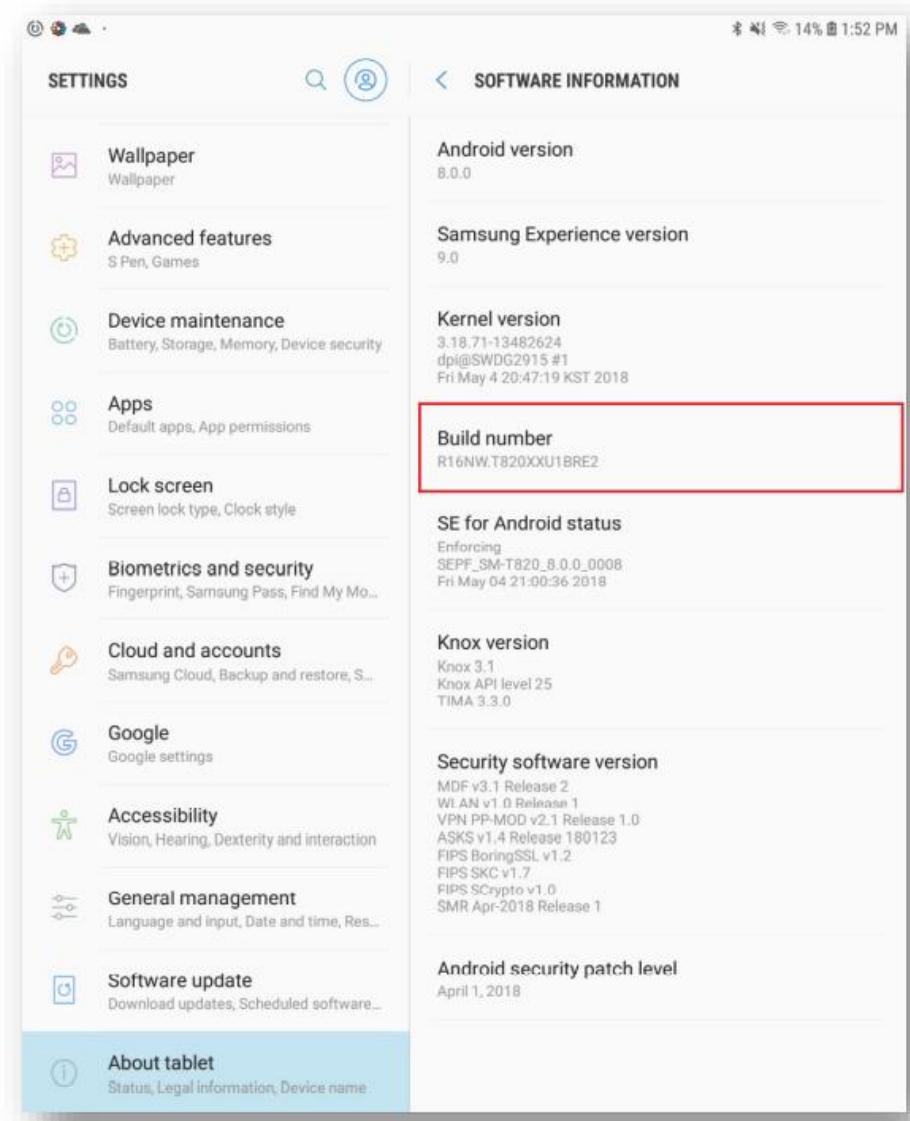
Preparing the Target Android Device for Development

In order to debug on a physical Android device, you need to enable “Developer Mode” and turn-on “USB Debugging.” To enable “Developer Mode”:

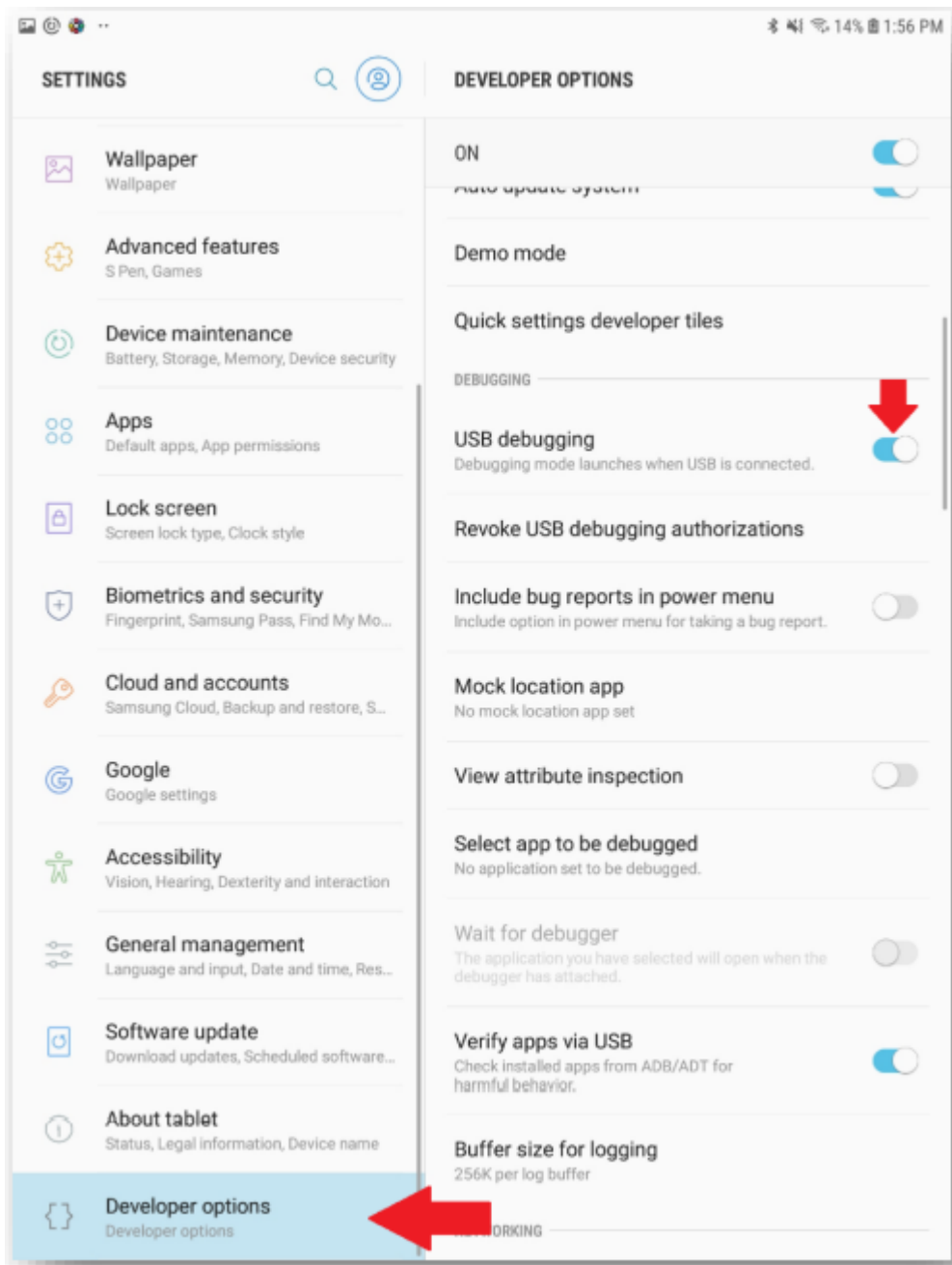
1. Start by going to your device’s settings:

- a) Look for your device's **build number**. Depending on your specific installation of Android, the build number might be behind slightly different menus, but it should generally reside within the "About" tab. For instance, this tablet has its build number under it's "About Tablet" menu within its "Software Information" sub-category.

Simply tap the "Build Number" box seven times to enable Developer Mode.



- b) There should now be a new "Developer Options" menu within your settings.



2. Go into the developer options and check **“USB Debugging.”**

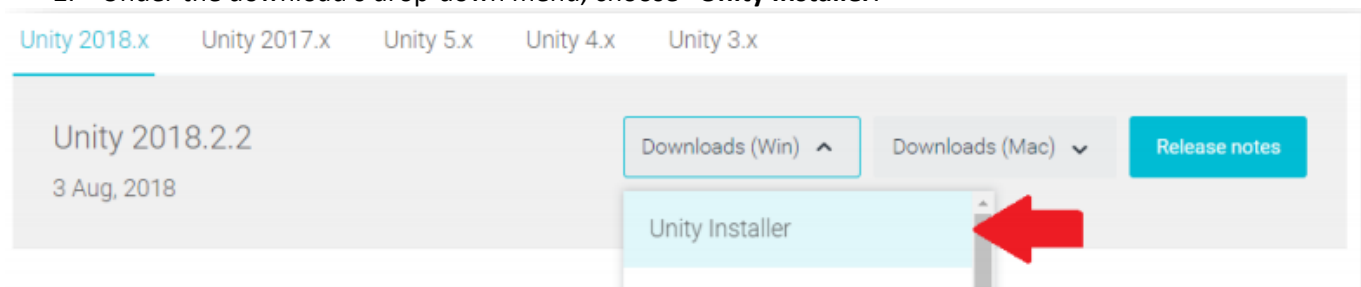


3. **Plug-in your device via USB to your computer.** New drivers should automatically install. Note: You may be prompted by your Android device to “Trust” a computer’s RSA key fingerprint. Select the options to “Always allow from this computer” and proceed.

Creating an Application in Unity

Downloading and Installing Unity¹

1. Visit the Unity Download archive and select the version that matches your team’s project. Otherwise, choose the latest version available from <https://unity3d.com/get-unity/download/archive>.
2. Under the download’s drop-down menu, choose “Unity Installer.”



¹ Note: Skip this step if you have already installed Unity.

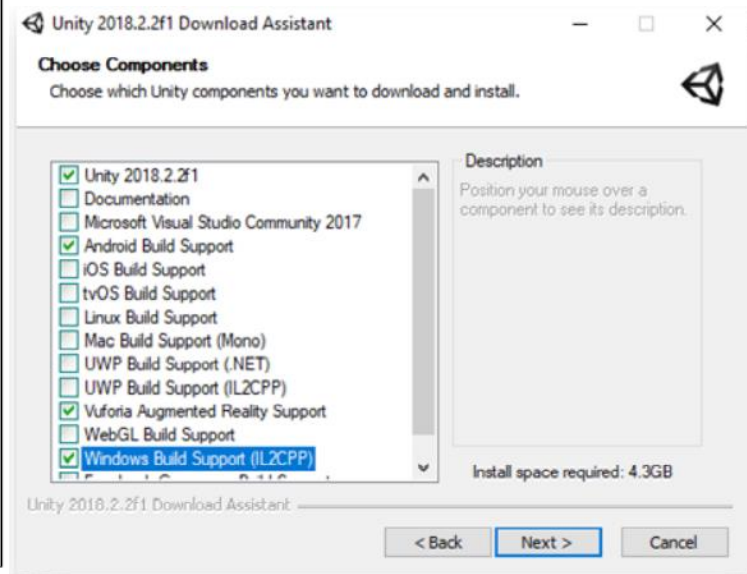
Select the following installer options:

- Unity ...
- Android Build Support
- Windows Build Support (IL2CPP)

Optionally install:

- Any other build platform that you'd like.
- Visual Studio if you don't already have an IDE to work in.
- Vuforia for Augmented Reality support.

If you already have Unity installed, you can use this installer to add Android Build Support to an existing Unity installation. Just be sure to **uncheck** all other options.

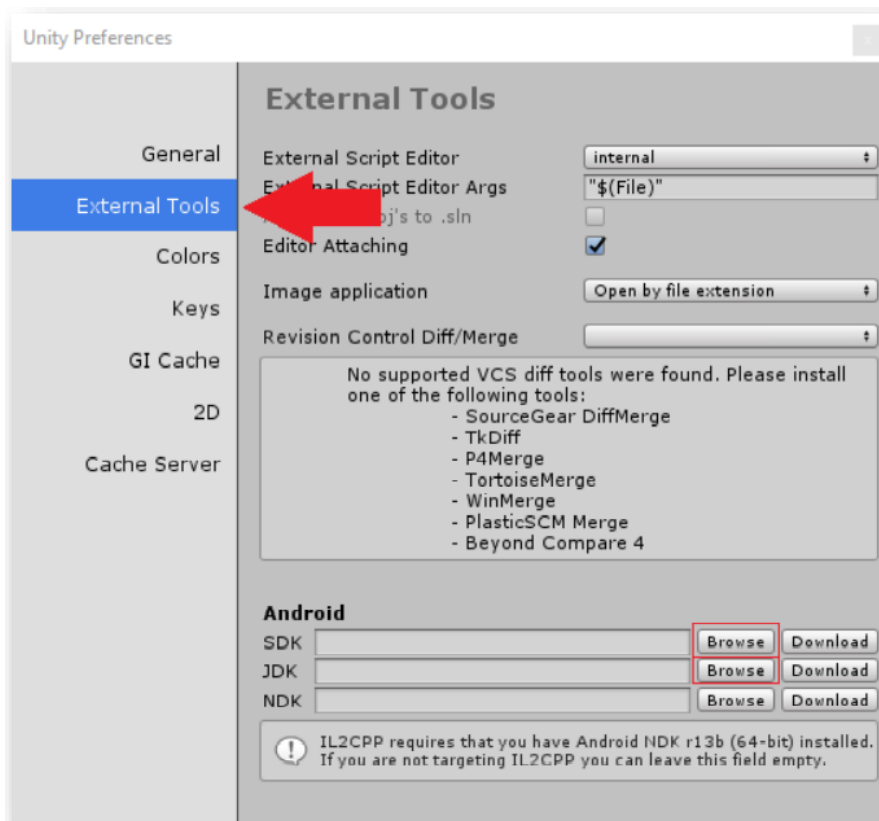


3. Continue with the installation process until Unity is ready to launch.

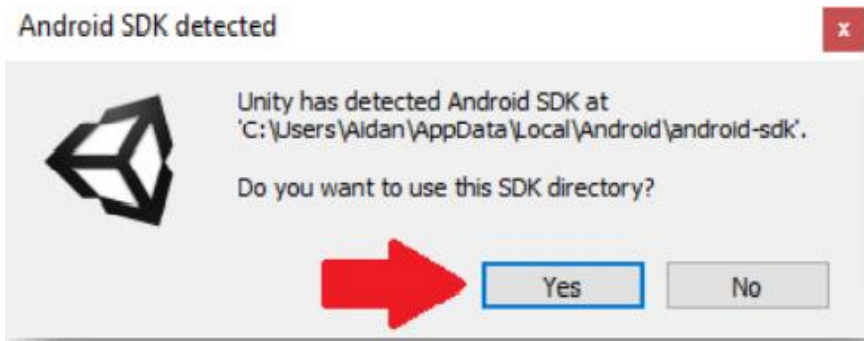
Setting up the Unity Editor for Android Development

After taking steps in 'Creating an Android Application' and 'Downloading and Installing Unity,' next configure your work station to build Android applications in Unity.

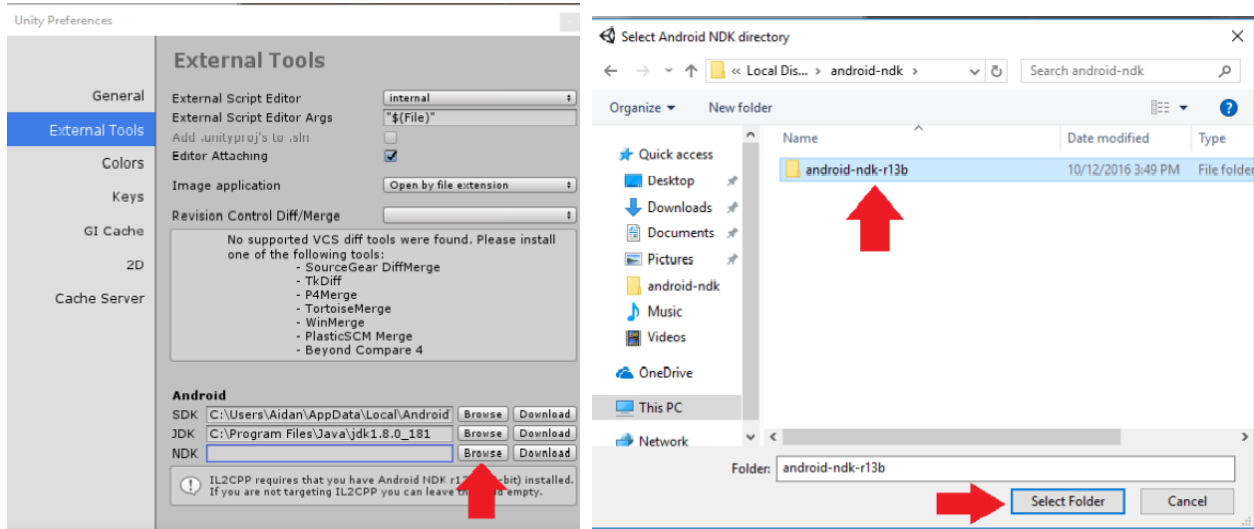
1. Start by opening the Unity editor. You can create a dummy project for this or open a preexisting project.
2. On the top menu bar, go to **Edit->Preferences** and head to **the External Tools** tab.



3. At the bottom of this menu, you will notice three open text-fields labeled, “SDK,” “JDK,” and “NDK.” For both the SDK and JDK category, select the “Browse” button. If you kept the default installation directories for these tools, Unity should auto-detect them and give you the option to use them. Click “Yes.” Otherwise, locate the JDK and SDK folders manually.

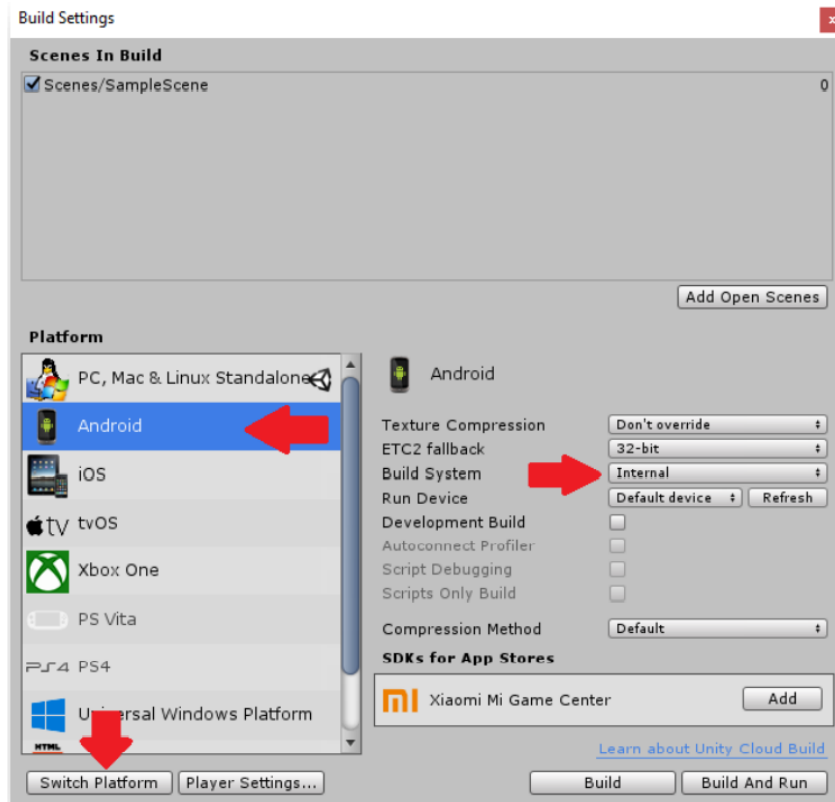


4. We will also need to install the NDK since the SILVIA Core relies on IL2CPP. **For this option, click the “Download” button first.** Your web browser should now open up and download a compressed ZIP file. Once this file is done downloading, **extract** the contents of the NDK folder to an easy-to-find location. (i.e. C:\Android-NDK). **Then, browse for the NDK folder.**



Your Unity environment is now set up and ready to go. However, there are just a few more application-specific steps left until you are ready to deploy.

1. On Unity's top-bar menu, select **File->Build Settings**.
2. Click on the **Android** button and hit **"Switch platform."**
3. On the **"Build System"** drop-down, change "Gradle" to **"Internal."**
4. Add any scenes you would like to be included with the build in the area above by clicking "Add open scenes" or by dragging in a scene file from Unity's Assets window. Scene number 0 will always load first.



5. Select the **"Player Settings"** button.

6. Inside of the “Other Settings” tab, there are a few options that we need to change.
7. Under “Identification,” there is an option called “Package Name.” This needs to be a worldwide unique identifier exclusive to your application in a specific format. Typically, the package name format is similar to a reversed “domain name.” For example, since our company website is CognitiveCode.com, and our new application is named “MyFirstApplication,” our package name could be something like “com.cognitivecode.myfirstapplication”.
8. Next are the API levels. Usually, the default Minimum API level and Target API are fine, but if your device is running a version of Android that is lower than what is set in the Target API level, you’ll need to adjust the Target API level to match your device’s edition of Android.
9. In the **Configuration** category, set your **Scripting Backend** to **IL2CPP** and your **API Compatibility Level** to **.NET 3.0** (not .NET 3.0 Subset).

The screenshot shows the 'Other Settings' window in Unity. The 'Identification' section is highlighted with a red box and contains the following settings:

Setting	Value
Package Name	com.cognitivecode.myfirstapplication
Version*	0.1
Bundle Version Code	1
Minimum API Level	Android 4.1 'Jelly Bean' (API level 16)
Target API Level	Automatic (highest installed)

The 'Configuration' section is also highlighted with a red box and contains the following settings:

Setting	Value
Scripting Runtime Version*	.NET 3.5 Equivalent
Scripting Backend	IL2CPP
Api Compatibility Level*	.NET 2.0
C++ Compiler Configuration	Release
Mute Other Audio Sources*	<input type="checkbox"/>

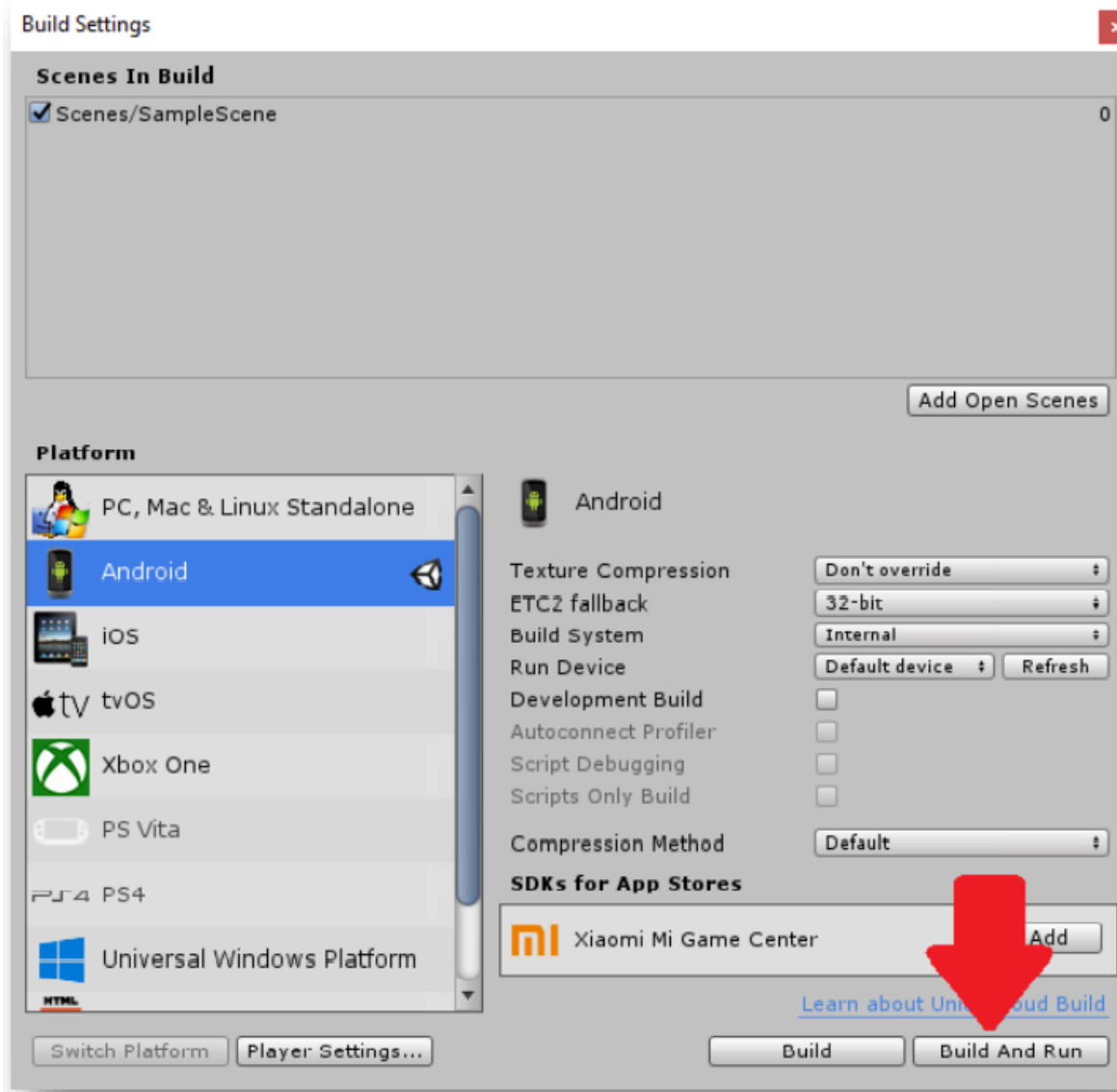
Below the highlighted sections, the 'Target Architectures' section is visible, showing 'ARMv7' and 'x86' checked, and 'ARM64' unchecked.

Note: These settings will allow you to locally test and debug your Android application, but they will not be sufficient for publishing an application on the Google Play Store. When you are ready to release, you will need

to update your Version Number and Bundle Version Code found in the “Identification” section. You will also need to generate a Keystore. Options for Keystores can be found in “Publishing Settings.” THESE INSTRUCTIONS WORK AT THE TIME OF THIS WRITING, BUT WE HAVE NO CONTROL OVER THE ANDROID OPERATING SYSTEM FROM GOOGLE—THEY MAY CHANGE WITH FUTURE RELEASES OF ANDROID.

Deployment Test

1. Plug-in your Android device and ensure that USB debugging is on. Go to **File->Build Settings->Build and Run** or use the hotkey shortcut **Ctrl+B**.
2. You will be prompted for a save-location for the APK file. This can be anywhere, but an easy solution would be to create a “Builds” folder inside of your Project folder.
3. Give your APK a name and save.
4. If everything is configured correctly, Visual Studio should now produce an APK file and automatically push/launch the application on your target device.



There is a visual studio solution file available as a template for building an Android application. Please talk with your Cognitive Code representative about getting the .sln and project files if you do not have it.

SILVIA API

silvia.app

This class, implemented as part of the SILVIA C#/LUA API, provides access to SILVIA's application interfaces. These interfaces are for applications to communicate with the SILVIA Core about application-specific data. For instance, the parent application can sample the SILVIA Core's text output, and modify a TextBox that is visible to the user based on that sampling.

Summary

silvia.app This class, implemented as part of the SILVIA C#/LUA API, provides access to SILVIA's application interfaces.

FUNCTIONS

<u>silvia.app.setVoiceFont</u>	Sets the voice font for output of AI speech (TTS)
<u>silvia.app.setVoiceType</u>	Sets the type (TTS Provider) of voice font for output of AI speech (TTS)
<u>silvia.app.getVoiceType</u>	Returns the type (TTS Provider) of voice font for output of AI speech (TTS)
<u>silvia.app.getVoiceFontGender</u>	Gets the gender of the last successfully set voice font for speech (TTS).
<u>silvia.app.getVoiceFontName</u>	Gets the name of the last successfully set voice font for speech (TTS).
<u>silvia.app.getVoiceFontRate</u>	Gets the numeric rate of the last successfully set voice font for speech (TTS).
<u>silvia.app.loadVisemes</u>	Loads visemes for lip-synced animation from the given folder
<u>silvia.app.loadVisemesCrop</u>	Loads visemes for lip-synced animation from the given folder, including a mouth cropping coefficient
<u>silvia.app.getVisemesFolder</u>	Returns the currently set visemes folder for lip-synced animation
<u>silvia.app.enableVoiceOutput</u>	Enables or disables the attached voice output
<u>silvia.app.voiceOutputEnabled</u>	Returns the enabled flag of the attached voice output
<u>silvia.app.enableTextOutput</u>	Enables or disables the attached text output
<u>silvia.app.textOutputEnabled</u>	Returns the enabled flag of the attached text output
<u>silvia.app.enableSocketOutput</u>	Enables or disables the attached socket output
<u>silvia.app.socketOutputEnabled</u>	Returns the enabled flag of the attached socket output
<u>silvia.app.enableDiagOutput</u>	Enables or disables the attached diagnostic output

[silvia.app.diagOutputEnabled](#) Returns the enabled flag of the attached diagnostic output

[silvia.app.enableApplicationMessage](#) Enables or disables the attached application message output

<u>silvia.app.applicationMessageEnabled</u>	Returns the enabled flag of the attached application message output
<u>silvia.app.setVoiceOutput</u>	Sets the voice output object's text field for subsequent output
<u>silvia.app.getVoiceOutput</u>	Returns the first (FIFO) string in the voice output object
<u>silvia.app.clearVoiceOutput</u>	Clears the voice output FIFO stack
<u>silvia.app.isSpeaking</u>	Checks the status of the application's voice output
<u>silvia.app.setIsSpeaking</u>	Forces an override of the status of the application's voice output
<u>silvia.app.setTextOutput</u>	Pushes the output onto the text output object's FIFO stack
<u>silvia.app.getTextOutput</u>	Returns the first (FIFO) string in the text output object
<u>silvia.app.clearTextOutput</u>	Clears the text output FIFO stack
<u>silvia.app.setSocketOutput</u>	Sends the string to the currently attached socket
<u>silvia.app.setDiagOutput</u>	Pushes the output onto the diagnostic output object's FIFO stack
<u>silvia.app.getDiagOutput</u>	Returns the first (FIFO) string in the diagnostic output object
<u>silvia.app.clearDiagOutput</u>	Clears the diagnostic FIFO stack
<u>silvia.app.setApplicationMessage</u>	Pushes the message output onto the application message object's FIFO stack
<u>silvia.app.getApplicationMessage</u>	Returns the first (FIFO) string in the application message object
<u>silvia.app.clearApplicationMessage</u>	Clears the application message FIFO stack
<u>silvia.app.setListening</u>	Enables or disables the parent application's speech recognition "hearing"
<u>silvia.app.isListening</u>	Returns the boolean status of the parent application's speech recognition "hearing"
<u>silvia.app.consoleOut</u>	Sends a text message to the SILVIA Studio debugging console, if available

Functions

[silvia.app.setVoiceFont](#)

Sets the voice font for output of AI speech (TTS)

Description

If a “SetVoiceFont” function has been registered with the core via the application, this method allows for scripted changes of the voice font used for speech output. Very useful for switching AI characters on the fly. Note that the application, not the core, is responsible for registering its own SetVoiceFont function via a delegate, as this feature is not native to the platform-independent core.

Example Usage (C#)

```
bool success = _core.ApiApp().SetVoiceFont("Female", "Audrey16", 0);
```

Example Usage (LUA)

```
success = silvia.app.setVoiceFont("Female", "Audrey16", 0)
```

Parameters

gender	The string to specify if the desired font is "Male" or "Female"
fontName	The string to specify the unique name of the font
rate	The numeric speaking rate of the voice. For MSWindows, this is between -10 and 10

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.setVoiceOutput](#)
- [silvia.app.loadVisemes](#)
- [silvia.app.getVoiceFontGender](#)
- [silvia.app.getVoiceFontName](#)
- [silvia.app.getVoiceFontRate](#)

silvia.app.setVoiceType

Sets the type (TTS Provider) of voice font for output of AI speech (TTS)

Description

This method allows for scripted changes of the voice font provider used for speech output. The type of voice determines which special tags are allowed or stripped in voice output.

Note that ALL speech tags are stripped from the text output.

Example Usage (C#)

```
bool success = _core.ApiApp().SetVoiceType("loquendo");
```


Example Usage (LUA)

```
success = silvia.app.setVoiceType("loquendo")
```

Parameters

type	The string to specify the desired font type. Currently allowed types include: "undefined", "att", "cepstral", "loquendo", and "microsoft".
------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.getVoiceType](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.loadVisemes](#)
- [silvia.app.getVoiceFontGender](#)
- [silvia.app.getVoiceFontName](#)
- [silvia.app.getVoiceFontRate](#)

[silvia.app.getVoiceType](#)

Returns the type (TTS Provider) of voice font for output of AI speech (TTS)

Description

This method returns the currently selected voice font provider used for speech output. The type of voice determines which special tags are allowed or stripped in voice output. Note that ALL speech tags are stripped from the text output.

Example Usage (C#)

```
String voicetype = _core.ApiApp().GetVoiceType();
```

Example Usage (LUA)

```
voicetype = silvia.app.getVoiceType()
```

Parameters

none

Returns

A string representing the currently selected voice font type. Current possible types include

"undefined", "att", "cepstral", "loquendo", and "microsoft".

See Also

- [silvia.app.setVoiceType](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.loadVisemes](#)
- [silvia.app.getVoiceFontGender](#)
- [silvia.app.getVoiceFontName](#)
- [silvia.app.getVoiceFontRate](#)

silvia.app.getVoiceFontGender

Gets the gender of the last successfully set voice font for speech (TTS).

Description

If a “SetVoiceFont” function has been registered with the core via the application, and a voice font has been successfully set, this method returns the gender of that voice font, either “Male” or “Female”. Note that the application, not the core, is responsible for registering its own SetVoiceFont function via a delegate, as this feature is not native to the platform-independent core.

Example Usage (C#)

```
String gender = _core.ApiApp().GetVoiceFontGender();
```

Example Usage (LUA)

```
gender = silvia.app.getVoiceFontGender()
```

Parameters

none

Returns

A string representing the gender, “Male” or “Female” of the current voice font. Returns a nil value if no voice font has been set.

See Also

- [silvia.app.setVoiceOutput](#)
- [silvia.app.loadVisemes](#)
- [silvia.app.setVoiceFont](#)
- [silvia.app.getVoiceFontName](#)
- [silvia.app.getVoiceFontRate](#)

silvia.app.getVoiceFontName

Gets the name of the last successfully set voice font for speech (TTS).

Description

If a “SetVoiceFont” function has been registered with the core via the application, and a voice font has been successfully set, this method returns the name of that voice font. Note that the application, not the core, is responsible for registering its own SetVoiceFont function via a delegate, as this feature is not native to the platform-independent core.

Example Usage (C#)

```
String name = _core.ApiApp().GetVoiceFontName();
```

Example Usage (LUA)

```
name = silvia.app.getVoiceFontName()
```

Parameters

none

Returns

A string representing the name of the current voice font. Returns a nil value if no voice font has been set.

See Also

- [silvia.app.setVoiceOutput](#)
- [silvia.app.loadVisemes](#)
- [silvia.app.setVoiceFont](#)
- [silvia.app.getVoiceFontGender](#)
- [silvia.app.getVoiceFontRate](#)

silvia.app.getVoiceFontRate

Gets the numeric rate of the last successfully set voice font for speech (TTS).

Description

If a “SetVoiceFont” function has been registered with the core via the application, and a voice font has been successfully set, this method returns the rate of output for that voice font. A rate of 0 means that the output is at a “normal” speed. Numbers between 1 and 10 mean that the voice output is faster than normal, whereas numbers between -1 and -10 denote slower than normal output. Note that the application, not the core, is responsible for registering its own SetVoiceFont function via a delegate, as this feature is not native to the platform-independent core.

Example Usage (C#)

```
int rate = _core.ApiApp().GetVoiceFontRate();
```

Example Usage (LUA)

```
rate = silvia.app.getVoiceFontRate()
```

Parameters

none

Returns

A number representing the output rate of the current voice font. Returns a 0 value by default if no voice font has been set.

See Also

- [silvia.app.setVoiceOutput](#)
- [silvia.app.loadVisemes](#)
- [silvia.app.setVoiceFont](#)
- [silvia.app.getVoiceFontGender](#)
- [silvia.app.getVoiceFontName](#)

silvia.app.loadVisemes

Loads visemes for lip-synced animation from the given folder

Description

If a “LoadVisemes” function has been registered with the core via the application, this method allows for scripted changes of the character images used for visual display of lip- synced animation. Like the “setVoiceFont” function, this is very useful for switching AI characters on the fly. Note that the application, not the core, is responsible for registering its own LoadVisemes function via a delegate. While support for animation is embedded in the SILVIA Studio for windows, this feature is not native to the platform-independent core.

Example Usage (C#)

```
bool success = _core.ApiApp().LoadVisemes(“robotGuy/images”, true);
```

Example Usage (LUA)

```
success = silvia.app.loadVisemes(“robotGuy/images”, true)
```

Parameters

folder	The string to specify the relative or absolute folder where the new viseme images reside
useVideo	The boolean flag for enabling streaming character animation video

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.loadVisemesCrop](#)
- [silvia.app.setVoiceFont](#)
- [silvia.app.getVisemesFolder](#)

silvia.app.loadVisemesCrop

Loads visemes for lip-synced animation from the given folder, including a mouth cropping coefficient

Description

If a “LoadVisemes” function has been registered with the core via the application, this method allows for scripted changes of the character images used for visual display of lip- synced animation. Like the “setVoiceFont” function, this is very useful for switching AI characters on the fly. An additional feature of this method is that you can specify a cropping coefficient for the mouth visemes. This is useful for when you have expressive eye animations that have been rendered at full resolution and not pre-cropped. In this case, cropping will be performed at runtime. Note that the application, not the core, is responsible for registering its own LoadVisemes function via a delegate. While support for animation is embedded in SILVIA Studio for windows, this feature is not native to the platform-independent core.

Example Usage (C#)

```
bool success = _core.ApiApp().LoadVisemesCrop("robotGuy/images", true, 0.33f);
```

Example Usage (LUA)

```
success = silvia.app.loadVisemesCrop("robotGuy/images", true, 0.33)
```

Parameters

folder	The string to specify the relative or absolute folder where the new viseme images reside
useVideo	The boolean flag for enabling streaming character animation video
mouthCrop	The floating point coefficient for where the mouth should be cropped

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.loadVisemes](#)
- [silvia.app.setVoiceFont](#)
- [silvia.app.getVisemesFolder](#)

silvia.app.getVisemesFolder

Returns the currently set visemes folder for lip-synced animation

Description

If a “LoadVisemes” function has been registered with the core via the application, and visemes have been successfully loaded using said function, this method returns the folder of that most recent successfully loaded set of visemes. Note that the application, not the core, is responsible for registering its own LoadVisemes function via a delegate. While support for animation is embedded in SILVIA Studio for windows, this feature is not native to the platform-independent core.

Example Usage (C#)

```
String folder = _core.ApiApp().GetVisemesFolder();
```

Example Usage (LUA)

```
folder = silvia.app.getVisemesFolder()
```

Parameters

folder	The string to specify the relative or absolute folder where the new viseme images reside
--------	--

Returns

A string containing the relative or absolute folder where the current viseme images reside. nil is returned if no visemes have been successfully set.

See Also

- [silvia.app.setVoiceFont](#)
- [silvia.app.loadVisemes](#)

`silvia.app.enableVoiceOutput`

Enables or disables the attached voice output

Description

This method enables or disables the SILVIA core's ability to write to the voice output FIFO stack. Note that the application must fetch the contents of the FIFO stack using "GetVoiceOutput", and implement some sort of TTS system, as the SILVIA core does not support TTS natively.

Example Usage (C#)

```
_core.ApiApp().EnableVoiceOutput(true);
```

Example Usage (LUA)

```
silvia.app.enableVoiceOutput(true)
```

Parameters

enabled	The boolean flag to turn the feature on or off. The default setting is true.
---------	--

Returns

None.

See Also

- [silvia.app.enableTextOutput](#)
- [silvia.app.enableSocketOutput](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.getVoiceOutput](#)
- [silvia.app.clearVoiceOutput](#)

silvia.app.voiceOutputEnabled

Returns the enabled flag of the attached voice output

Description

This method checks the state, enabled or disabled, of the SILVIA core's ability to write to the voice output FIFO stack.

Example Usage (C#)

```
bool enabled = _core.ApiApp().VoiceOutputEnabled();
```

Example Usage (LUA)

```
enabled = silvia.app.voiceOutputEnabled()
```

Parameters

none

Returns

A boolean flag representing the enabled (true) or disabled (false) state of the voice output.

See Also

- [silvia.app.textOutputEnabled](#)
- [silvia.app.socketOutputEnabled](#)
- [silvia.app.diagOutputEnabled](#)
- [silvia.app.enableVoiceOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.getVoiceOutput](#)
- [silvia.app.clearVoiceOutput](#)

silvia.app.enableTextOutput

Enables or disables the attached text output

Description

This method enables or disables the SILVIA core's ability to write to the text output FIFO stack. Note that the application must fetch the contents of the FIFO stack using "GetTextOuput", and implement some sort of text display or other output system, as the SILVIA core does not support text display natively.

Example Usage (C#)

```
_core.ApiApp().EnableTextOutput(true);
```

Example Usage (LUA)

```
silvia.app.enableTextOutput(true)
```

Parameters

enabled

The boolean flag to turn the feature on or off. The default setting is true.

Returns

None.

See Also

- [silvia.app.enableVoiceOutput](#)

- [silvia.app.enableSocketOutput](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.getTextOutput](#)
- [silvia.app.clearTextOutput](#)

silvia.app.textOutputEnabled

Returns the enabled flag of the attached text output

Description

This method checks the state, enabled or disabled, of the SILVIA core's ability to write to the text output FIFO stack.

Example Usage (C#)

```
bool enabled = _core.ApiApp().TextOutputEnabled();
```

Example Usage (LUA)

```
enabled = silvia.app.textOutputEnabled()
```

Parameters

none

Returns

A boolean flag representing the enabled (true) or disabled (false) state of the text output.

See Also

- [silvia.app.voiceOutputEnabled](#)
- [silvia.app.socketOutputEnabled](#)
- [silvia.app.diagOutputEnabled](#)
- [silvia.app.enableTextOutput](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.getTextOutput](#)
- [silvia.app.clearTextOutput](#)

silvia.app.enableSocketOutput

Enables or disables the attached socket output

Description

If a socket connection has been created and the appropriate LUA function has been globally defined and attached to the silvia.socket table via the application, this LUA function enables or disables the sending of the output to that

socket. Note that while this may be set from C#, the actual socket output is via LUA only. In C#, you may set up your own socket output via direct scripting, or through a loaded and registered plugin.

Example Usage (C#)

```
_core.ApiApp().EnableSocketOutput(true);
```

Example Usage (LUA)

```
silvia.app.enableSocketOutput(true)
```

Parameters

enabled	The boolean flag to turn the feature on or off. The default setting is false.
---------	---

Returns

None.

See Also

- [silvia.app.enableTextOutput](#)
- [silvia.app.enableVoiceOutput](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.setSocketOutput](#)

silvia.app.socketOutputEnabled

Returns the enabled flag of the attached socket output

Description

This method checks the state, enabled or disabled, of the SILVIA core's ability to write to the attached (LUA) socket output. Note that while this state may be checked from C#, any actual socket output is via LUA only. In C#, you may set up your own socket output via direct scripting, or through a loaded and registered plugin.

Example Usage (C#)

```
bool enabled = _core.ApiApp().SocketOutputEnabled();
```

Example Usage (LUA)

```
enabled = silvia.app.socketOutputEnabled()
```

Parameters

none

Returns

A boolean flag representing the enabled (true) or disabled (false) state of the socket output.

See Also

- [silvia.app.voiceOutputEnabled](#)
- [silvia.app.textOutputEnabled](#)
- [silvia.app.diagOutputEnabled](#)
- [silvia.app.enableSocketOutput](#)
- [silvia.app.setSocketOutput](#)

silvia.app.enableDiagOutput

Enables or disables the attached diagnostic output

Description

This method enables or disables the SILVIA core's ability to write to the diagnostic output FIFO stack. Note that the application must fetch the contents of the FIFO stack using "GetDiagOutput", and implement some sort of diagnostic display or logging system, as the SILVIA core does not support text display or logging natively.

Example Usage (C#)

```
_core.ApiApp().EnableDiagOutput(true);
```

Example Usage (LUA)

```
silvia.app.enableDiagOutput(true)
```

Parameters

enabled	The boolean flag to turn the feature on or off. The default setting is true.
---------	--

Returns

None.

See Also

- [silvia.app.enableTextOutput](#)
- [silvia.app.enableSocketOutput](#)
- [silvia.app.enableVoiceOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.getDiagOutput](#)
- [silvia.app.clearDiagOutput](#)

silvia.app.diagOutputEnabled

Returns the enabled flag of the attached diagnostic output

Description

This method checks the state, enabled or disabled, of the SILVIA core's ability to write to the diagnostic output FIFO stack.

Example Usage (C#)

```
bool enabled = _core.ApiApp().DiagOutputEnabled();
```

Example Usage (LUA)

```
enabled = silvia.app.diagOutputEnabled()
```

Parameters

none

Returns

A boolean flag representing the enabled (true) or disabled (false) state of the diagnostic output.

See Also

- [silvia.app.voiceOutputEnabled](#)
- [silvia.app.socketOutputEnabled](#)
- [silvia.app.textOutputEnabled](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.getDiagOutput](#)
- [silvia.app.clearDiagOutput](#)

[silvia.app.enableApplicationMessage](#)

Enables or disables the attached application message output

Description

This method enables or disables the SILVIA core's ability to write to the application message FIFO stack. Note that the application must fetch the contents of the FIFO stack using "GetApplicationMessage", and implement some sort message handling for the application.

Example Usage (C#)

```
_core.ApiApp().EnableApplicationMessage(true);
```

Example Usage (LUA)

```
silvia.app.enableApplicationMessage(true)
```

Parameters

<code>enabled</code>	The boolean flag to turn the feature on or off. The default setting is true.
----------------------	--

Returns

None.

See Also

- [`silvia.app.applicationMessageEnabled`](#)
- [`silvia.app.setApplicationMessage`](#)
- [`silvia.app.getApplicationMessage`](#)
- [`silvia.app.clearApplicationMessage`](#)

`silvia.app.applicationMessageEnabled`

Returns the enabled flag of the attached application message output

Description

This method checks the state, enabled or disabled, of the SILVIA core's ability to write to the application message FIFO stack.

Example Usage (C#)

```
bool enabled = _core.ApiApp().ApplicationMessageEnabled();
```

Example Usage (LUA)

```
enabled = silvia.app.applicationMessageEnabled()
```

Parameters

none

Returns

A boolean flag representing the enabled (true) or disabled (false) state of the application message output.

See Also

- [`silvia.app.enableApplicationMessage`](#)
- [`silvia.app.setApplicationMessage`](#)
- [`silvia.app.getApplicationMessage`](#)

- [silvia.app.clearApplicationMessage](#)

silvia.app.setVoiceOutput

Sets the voice output object's text field for subsequent output

Description

If the voice output has been enabled (default), via the application, this method pushes the output of a string via the voice output stack. Either the application or a script may poll and fetch voice output messages via the `getVoiceOutput` method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such messages.

Example Usage (C#)

```
bool success = _core.ApiApp().SetVoiceOutput("I'm feeling much better now, Dave.");
```

Example Usage (LUA)

```
success = silvia.app.setVoiceOutput("I'm feeling much better now, Dave.")
```

Parameters

output	The string for voice output via the AI output handler.
--------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.setVoiceFont](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.setSocketOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.enableVoiceOutput](#)
- [silvia.app.getVoiceOutput](#)
- [silvia.app.clearVoiceOutput](#)

silvia.app.getVoiceOutput

Returns the first (FIFO) string in the voice output object

Description

If the voice output has been enabled (default), via the application, this method fetches the first string from the voice output stack. Either the application or a script may poll and fetch voice output via this method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such output.

Example Usage (C#)

```
String voice = _core.ApiApp().GetVoiceOutput();
```

Example Usage (LUA)

```
voice = silvia.app.getVoiceOutput()
```

Parameters

none

Returns

The string value representing the voice output. nil is returned if disabled or empty.

See Also

- [silvia.app.getTextOutput](#)
- [silvia.app.getDiagOutput](#)
- [silvia.app.enableVoiceOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.clearVoiceOutput](#)

silvia.app.clearVoiceOutput

Clears the voice output FIFO stack

Description

This method clears out the voice output FIFO stack. Either an API call from the application or a call from a script may be used to invoke this method as a housecleaning tool, in order to clear out any unused or otherwise undesirable voice output messages.

Example Usage (C#)

```
bool cleared = _core.ApiApp().ClearVoiceOutput();
```

Example Usage (LUA)

```
cleared = silvia.app.clearVoiceOutput()
```

Parameters

none

Returns

The boolean success or failure of the operation. If the FIFO stack was already empty, then false is returned.

See Also

- [silvia.app.clearTextOutput](#)
- [silvia.app.clearDiagOutput](#)
- [silvia.app.enableVoiceOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.getVoiceOutput](#)

silvia.app.isSpeaking

Checks the status of the application's voice output

Description

This method returns a true or false. Great for holding off execution of functions until the app is done with speech output.

Example Usage (C#)

```
bool speaking = _core.ApiApp().IsSpeaking();
```

Example Usage (LUA)

```
speaking = silvia.app.isSpeaking()
```

Parameters

none

Returns

The boolean condition of the application's speaking state. Default is false.

See Also

- [silvia.app.enableVoiceOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.getVoiceOutput](#)

silvia.app.setIsSpeaking

Forces an override of the status of the application's voice output

Description

This method forces a set of the "isSpeaking" value. Great for holding off execution of functions until the app is done with speech output. From script, always set "force" to true.

Example Usage (C#)

```
_core.ApiApp().SetIsSpeaking(true, true);
```

Example Usage (LUA)

```
silvia.app.setIsSpeaking(true, true)
```

Parameters

enabled	the boolean flag to set the status on or off
force	the boolean flag to force the status to remain on or off for a few seconds

Returns

None.

See Also

- [silvia.app.enableVoiceOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.getVoiceOutput](#)
- [silvia.app.isSpeaking](#)

silvia.app.setTextOutput

Pushes the output onto the text output object's FIFO stack

Description

If the text output has been enabled (default), via the application, this method pushes the output of a string to the text output stack. Either the application or a script may poll and fetch text output via the `getTextOutput` method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such output.

Example Usage (C#)

```
bool success = _core.ApiApp().SetTextOutput("Shall we continue in text only mode?");
```

Example Usage (LUA)

```
success = silvia.app.setTextOutput("Shall we continue in text only mode?")
```

Parameters

output	The string for text output.
--------	-----------------------------

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.setSocketOutput](#)
- [silvia.app.setVoiceOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.enableTextOutput](#)
- [silvia.app.getTextOutput](#)
- [silvia.app.clearTextOutput](#)

silvia.app.getTextOutput

Returns the first (FIFO) string in the text output object

Description

If the text output has been enabled (default), via the application, this method fetches the first string from the text output stack. Either the application or a script may poll and fetch text output via this method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such output.

Example Usage (C#)

```
String text = _core.ApiApp().GetTextOutput();
```

Example Usage (LUA)

```
text = silvia.app.getTextOutput()
```

Parameters

none

Returns

The string value representing the text output. nil is returned if disabled or empty.

See Also

- [silvia.app.getVoiceOutput](#)
- [silvia.app.getDiagOutput](#)
- [silvia.app.enableTextOutput](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.clearTextOutput](#)

`silvia.app.clearTextOutput`

Clears the text output FIFO stack

Description

This method clears out the text output FIFO stack. Either an API call from the application or a call from script may be used to invoke this method as a housecleaning tool, in order to clear out any unused or otherwise undesirable text output messages.

Example Usage (C#)

```
bool cleared = _core.ApiApp().ClearTextOutput();
```

Example Usage (LUA)

```
cleared = silvia.app.clearTextOutput()
```

Parameters

none

Returns

The boolean success or failure of the operation. If the FIFO stack was already empty, then false is returned.

See Also

- [silvia.app.clearVoiceOutput](#)
- [silvia.app.clearDiagOutput](#)
- [silvia.app.enableTextOutput](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.getTextOutput](#)

silvia.app.setSocketOutput

Sends the string to the currently attached socket

Description

If the socket connection and output function has been established via a SILVIA LUA script (see socket.slv for an example), this method forces the sending of the string as a message to that socket's connection. Note that the application specific LUA script, not the core, is responsible for the setup and attachment of the appropriate socket scripts and the connection to an external client or server.

Example Usage (C#)

```
bool success = _core.ApiApp().SetSocketOutput("I am an in game character.");
```

Example Usage (LUA)

```
success = silvia.app.setSocketOutput("I am an in game character.")
```

Parameters

output	The string for socket message output via the pre-attached socket
--------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.setVoiceOutput](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.enableSocketOutput](#)

silvia.app.setDiagOutput

Pushes the output onto the diagnostic output object's FIFO stack

Description

If the diagnostic output has been enabled (default), via the application, this method pushes the output of a string via the diagnostic output stack. Either the application or a script may poll and fetch diagnostic output via the `getDiagOutput` method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such output.

Example Usage (C#)

```
bool success = _core.ApiApp().SetDiagOutput("MYWARNING user response was incorrect.");
```

Example Usage (LUA)

```
success = silvia.app.setDiagOutput("MYWARNING user response was incorrect.")
```

Parameters

output	The string for diagnostic output.
--------	-----------------------------------

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.setVoiceOutput](#)
- [silvia.app.setTextOutput](#)
- [silvia.app.setSocketOutput](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.getDiagOutput](#)
- [silvia.app.clearDiagOutput](#)

silvia.app.getDiagOutput

Returns the first (FIFO) string in the diagnostic output object

Description

If the diagnostic output has been enabled (default), via the application, this method fetches the first string from the diagnostic output stack. Either the application or a script may poll and fetch diagnostic output via this method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such output.

Example Usage (C#)

```
String diag = _core.ApiApp().GetDiagOutput();
```

Example Usage (LUA)

```
diag = silvia.app.getDiagOutput()
```

Parameters

none

Returns

The string value representing the diagnostic message. nil is returned if disabled or empty.

See Also

- [silvia.app.getVoiceOutput](#)
- [silvia.app.getTextOutput](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.clearDiagOutput](#)

silvia.app.clearDiagOutput

Clears the diagnostic FIFO stack

Description

This method clears out the diagnostic FIFO stack. Either an API call from the application or a call from script may be used to invoke this method as a housecleaning tool, in order to clear out any unused or otherwise undesirable diagnostic messages.

Example Usage (C#)

```
bool cleared = _core.ApiApp().ClearDiagOutput();
```

Example Usage (LUA)

```
cleared = silvia.app.clearDiagOutput()
```

Parameters

none

Returns

The boolean success or failure of the operation. If the FIFO stack was already empty, then false is returned.

See Also

- [silvia.app.clearVoiceOutput](#)
- [silvia.app.clearTextOutput](#)
- [silvia.app.enableDiagOutput](#)
- [silvia.app.setDiagOutput](#)
- [silvia.app.getDiagOutput](#)

silvia.app.setApplicationMessage

Pushes the message output onto the application message object's FIFO stack

Description

If the application message output has been enabled (default), via the application, this method pushes the output of a string via the application message stack. Either the application or a script may poll and fetch application messages via the `getApplicationMessage` method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such messages.

Example Usage (C#)

```
bool success = _core.ApiApp().SetApplicationMessage("Close Main Form");
```

Example Usage (LUA)

```
success = silvia.app.setApplicationMessage("Close Main Form")
```

Parameters

output	The string representing the message to be sent to the application.
--------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.enableApplicationMessage](#)
- [silvia.app.getApplicationMessage](#)
- [silvia.app.clearApplicationMessage](#)

silvia.app.getApplicationMessage

Returns the first (FIFO) string in the application message object

Description

If the application message output has been enabled (default), via the application, this method fetches the first string from the application message stack. Either the application or a LUA/C# script may poll and fetch application messages via this method. Note that the application, not the core, is responsible for handling the fetch and subsequent operations on such messages.

Example Usage (C#)

```
String appMessage = _core.ApiApp().GetApplicationMessage();
```

Example Usage (LUA)

```
appMessage = silvia.app.getApplicationMessage()
```

Parameters

none

Returns

The string value representing the message for the application. nil is returned if disabled or empty.

See Also

- [silvia.app.enableApplicationMessage](#)
- [silvia.app.setApplicationMessage](#)
- [silvia.app.clearApplicationMessage](#)

silvia.app.clearApplicationMessage

Clears the application message FIFO stack

Description

This method clears out the application message FIFO stack. Either an API call from the application or a call from script may be used to invoke this method as a housecleaning tool, in order to clear out any unused or otherwise undesirable application messages.

Example Usage (C#)

```
bool cleared = _core.ApiApp().ClearApplicationMessage();
```

Example Usage (LUA)

```
cleared = silvia.app.clearApplicationMessage()
```

Parameters

none

Returns

The boolean success or failure of the operation. If the FIFO stack was already empty, then false is returned.

See Also

- [silvia.app.enableApplicationMessage](#)
- [silvia.app.setApplicationMessage](#)
- [silvia.app.getApplicationMessage](#)

silvia.app.setListening

Enables or disables the parent application's speech recognition "hearing"

Description

If the listening toggle object (System.Windows.Forms.CheckBox) has been registered via the application, this method sets that CheckBox.Checked value. Note that the application, not the core, is responsible for attaching and handling the CheckBox.CheckedChanged event, and for setting up a platform specific speech recognition solution.

Example Usage (C#)

```
bool success = _core.ApiApp().SetListening(true);
```

Example Usage (LUA)

```
success = silvia.app.setListening(true)
```

Parameters

enabled	a boolean value to enable or disable listening in the parent application
---------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.app.isListening](#)

[silvia.app.isListening](#)

Returns the boolean status of the parent application's speech recognition "hearing"

Description

If the listening toggle object (System.Windows.Forms.CheckBox) has been registered via the application, this method returns that CheckBox.Checked value. Note that the application, not the core, is responsible for attaching and handling the CheckBox.CheckedChanged event, and for setting up a platform specific speech recognition solution.

Example Usage (C#)

```
bool listening = _core.ApiApp().IsListening();
```

Example Usage (LUA)

```
listening = silvia.app.isListening()
```

Parameters

none

Returns

The boolean value of the listening status, always false if no registered CheckBox object.

See Also

- [silvia.app.setListening](#)

`silvia.app.consoleOut`

Sends a text message to the SILVIA Studio debugging console, if available

Description

If the runtime is SILVIA Studio, and the debugging console is open, the given text message string is posted to the console, and a boolean value of true is returned. If the console is not open, or the runtime is not SILVIA Studio, nothing is posted and a value of false is returned.

Example Usage (C#)

```
bool success = _core.ApiApp().ConsoleOut("I reached this point in my script.");
```

Example Usage (LUA)

```
success = silvia.app.consoleOut("I reached this point in my script.")
```

Parameters

message	a string value representing the message to be posted to the console
---------	---

Returns

The boolean value representing the success or failure of the operation

See Also

- [silvia.app.setDiagOutput](#)
- [silvia.app.getDiagOutput](#)

silvia.brain

This class, implemented as part of the SILVIA API, provides access to high-level SILVIA runtime functionality. Specifically, this class controls basic SILVIA brain operations, and provides an interface to invoke brain interactions, and to control certain brain parameters in order to customize global aspects of how a brain responds to input. Examples are shown in Java, C#, and LUA.

Summary

<u>silvia.brain</u>	This class, implemented as part of the SILVIA API, provides access to high-level SILVIA runtime functionality.
----------------------------	--

FUNCTIONS

silvia.brain.setUserSecurityLevel	Sets the numeric integer security level for the current user.
---	---

[silvia.brain.getUserSecurityLevel](#)

Returns the integer security level for the current user.

[silvia.brain.addConcepts](#)

Invokes the core SILVIA algorithms to register the concepts in a string

<u>silvia.brain.getResponse</u>	Invokes the core SILVIA algorithms to get a response to the input
<u>silvia.brain.getResponseBehaviorID</u>	Returns the behavior id in the given position on the response stack
<u>silvia.brain.getResponseAbsorberID</u>	Returns the Absorber id in the given position on the response stack
<u>silvia.brain.getResponseWeight</u>	Returns the numeric weight of the match in the given position on the response stack
<u>silvia.brain.transformNarrativeMode</u>	Invokes the core SILVIA algorithms to transform the string from second to first-person or first to second person.
<u>silvia.brain.executePostEvents</u>	Invokes the core SILVIA algorithms to execute post-response events and scripts
<u>silvia.brain.setJump</u>	Sets the group and name of the behavior to “jump” to next
<u>silvia.brain.setBypassResponse</u>	On the next input, the “GetResponse” function will ignore the input and jump to the next input
<u>silvia.brain.removeStopwords</u>	Takes an input string and removes the stopword concepts, returning the result
<u>silvia.brain.setDynamicAttraction</u>	Modifies the state of the SILVIA Core algorithms, changing the “attraction” parameter for dynamic output
<u>silvia.brain.setDynamicDepth</u>	Modifies the state of the SILVIA Core algorithms, changing the “depth” parameter for dynamic output
<u>silvia.brain.setDynamicFalloffDepth</u>	Modifies the state of the SILVIA Core algorithms, changing the “falloff depth” parameter for dynamic output
<u>silvia.brain.setDynamicFalloff</u>	Modifies the state of the SILVIA Core algorithms, changing the “falloff” parameter for dynamic output
<u>silvia.brain.setDynamicAdaptation</u>	Modifies the state of the SILVIA Core algorithms, setting the “adaptation” parameter for dynamic output
<u>silvia.brain.generateDynamic</u>	Invokes the core SILVIA algorithms to generate a dynamic output
<u>silvia.brain.generateDynamicLimited</u>	Invokes the core SILVIA algorithms to generate a dynamic output from a limited set of concepts
<u>silvia.brain.generateDynamicFromMemory</u>	Invokes the core SILVIA algorithms to generate a dynamic output from the feedback stack

<u>silvia.brain.dynamicHasAllConceptsInOne</u>	Returns a true or false indication of result suitability after a call to generateDynamic
<u>silvia.brain.setAbsorberThreshold</u>	Invokes the core SILVIA algorithms to change the “threshold” for Absorber acceptance
<u>silvia.brain.setReusableThreshold</u>	Invokes the core SILVIA algorithms to change the “threshold” for Exuder matching -----
<u>silvia.brain.setAddressByName</u>	Enables or disables the SILVIA core’s requirement to be addressed by name before

responding

<u>silvia.brain.getAddressByName</u>	Returns the boolean enabled/disabled status of the SILVIA core's AddressByName mode
<u>silvia.brain.setAllEars</u>	Enables or disables the SILVIA core's ability to respond to the next input
<u>silvia.brain.getAllEars</u>	Returns the boolean enabled/disabled status of the SILVIA core's AllEars mode
<u>silvia.brain.loadCommandAssembly</u>	Loads the name command assembly so that the functions are available to SILVIA

Functions

silvia.brain.setUserSecurityLevel

Sets the numeric integer security level for the current user.

Description

This method sets the current security level for the user. The security range is 0 to n, where 0 is the lowest “public” level of security. Within the SILVIA brain data itself, any behavior and/or Exuder security value higher than 0 must be met with an equal or greater user security level for the behavior/Exuder to be invoked. Note that a higher behavior security level can override a lower Exuder security level.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetUserSecurityLevel(3);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetUserSecurityLevel(3);
```

Example Usage (LUA)

```
success = silvia.brain.setUserSecurityLevel(3)
```

Parameters

level	The numeric integer value representing the new security level to apply to the user
-------	--

Returns

The boolean success or failure of the operation.

`silvia.brain.getUserSecurityLevel`

Returns the integer security level for the current user.

Description

This method returns the current security level for the user. The security range is 0 to n, where 0 is the lowest “public” level of security. Within the SILVIA brain data itself, any behavior and/or Exuder security value higher than 0 must be met with an equal or greater user security level for the behavior/Exuder to be invoked. Note that a higher behavior security level can override a lower Exuder security level.

Example Usage (Java/C#)

```
int level = _core.ApiBrain().GetUserSecurityLevel();
```

Example Usage (LUA)

```
level = silvia.brain.getUserSecurityLevel()
```

Parameters

none

Returns

level	The number value representing the current security level of the user.
-------	---

silvia.brain.addConcepts

Invokes the core SILVIA algorithms to register the concepts in a string

Description

This method invokes AI run-time to add the concepts contained in the given string. Any concepts that are already registered will be ignored. This function is useful for pre-loading concepts that may be used dynamically within a variable or other dynamic construct, but are not yet part of SILVIA’s current dictionary.

Example Usage (Java)

```
boolean success = _core.ApiBrain().AddConcepts(“how are you today”, false, false);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().AddConcepts(“how are you today”, false, false);
```

Example Usage (LUA)

```
success = silvia.brain.addConcepts(“how are you today”, false, false)
```

Parameters

input	The string representing the concept(s) to be processed.
punctuate	The boolean flag to determine if punctuation is considered
conceptualize	The boolean flag to determine if existing concatenated concepts are considered

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.getResponse](#)

silvia.brain.getResponse

Invokes the core SILVIA algorithms to get a response to the input

Description

This method invokes AI run-time to produce a response. This response can be returned as a text string for output, but this function can also potentially invoke scripts, and can change the state of the AI depending on the input, the current state of the variable data, the current knowledge base, and behavioral data. The exceptions are any registered events and post-Exuder scripts. These must be invoked explicitly via the `silvia.brain.executePostEvents` method.

Example Usage (Java/C#)

```
String result = _core.ApiBrain().GetResponse("how are you today",  
"logs/mylogfile.txt");
```

Example Usage (LUA)

```
result = silvia.brain.getResponse("how are you today", "logs/mylogfile.txt")
```

Parameters

input	The string representing the input to be processed.
-------	--

logFile

An optional file to store logging information. With pre-existing files,

logging is additive.

Returns

The string representing the output produced by SILVIA.

See Also

- [silvia.brain.executePostEvents](#)

[silvia.brain.getResponseBehaviorID](#)

Returns the behavior id in the given position on the response stack

Description

After invoking the AI run-time to produce a response, a stack of zero-indexed “best possible” behavioral results is available for query. Using these methods, the application developer can determine at runtime exactly which behaviors and Absorbers were considered the top candidate matches for the user input. This particular method returns the integer id of the behavior in the given position on the stack. Note that this is the same data as shown in the text output window of SILVIA Studio when in debug mode.

Example Usage (Java/C#)

```
int behavior = _core.ApiBrain().GetResponseBehaviorID(0);
```

Example Usage (LUA)

```
behavior = silvia.brain.getResponseBehaviorID(0)
```

Parameters

index	The integer zero-based index into the “best match” stack
-------	--

Returns

The integer index of the best matching behavior in that stack position. A -1 is returned if there is no valid match.

See Also

- [silvia.brain.getResponse](#)
- [silvia.brain.getResponseAbsorberID](#)
- [silvia.brain.getResponseWeight](#)

[silvia.brain.getResponseAbsorberID](#)

Returns the Absorber id in the given position on the response stack

Description

After invoking the AI run-time to produce a response, a stack of zero-indexed “best possible” behavioral results is available for query. Using these methods, the application developer can determine at runtime exactly which behaviors and Absorbers were considered the top candidate matches for the user input. This particular method returns the integer id of the Absorber in the given position on the stack. Note that this is the same data as shown in the text output window of SILVIA Studio when in debug mode.

Example Usage (Java/C#)

```
int Absorber = _core.ApiBrain().GetResponseAbsorberID(0);
```

Example Usage (LUA)

```
Absorber = silvia.brain.getResponseAbsorberID(0)
```

Parameters

index	The integer zero-based index into the “best match” stack
-------	--

Returns

The integer index of the best matching Absorber in that stack position. A -1 value is returned if there is no valid match.

See Also

- [silvia.brain.getResponse](#)
- [silvia.brain.getResponseBehaviorID](#)
- [silvia.brain.getResponseWeight](#)

[silvia.brain.getResponseWeight](#)

Returns the numeric weight of the match in the given position on the response stack

Description

After invoking the AI run-time to produce a response, a stack of zero-indexed “best possible” behavioral results is available for query. Using these methods, the application developer can determine at runtime exactly which behaviors and Absorbers were considered the top candidate matches for the user input. This particular method returns the numeric weight of the match in the given position on the stack. A higher value indicates a better match. Note that this is the same data as shown in the text output window of SILVIA Studio when in debug mode.

Example Usage (Java/C#)

```
float weight = _core.ApiBrain().GetResponseWeight(0);
```

Example Usage (LUA)

```
weight = silvia.brain.getResponseWeight(0)
```

Parameters

index	The integer zero-based index into the “best match” stack
-------	--

Returns

The numeric weight of the best matching behavior in that stack position. A -100.0 is returned if there is no valid match.

See Also

- [silvia.brain.getResponse](#)
- [silvia.brain.getResponseBehaviorID](#)
- [silvia.brain.getResponseAbsorberID](#)

silvia.brain.transformNarrativeMode

Invokes the core SILVIA algorithms to transform the string from second to first-person or first to second person.

Description

This method invokes the AI run-time to explicitly transform a piece of input so that the frame of reference is correctly switched between speakers. This method invokes algorithms that use statistical language rules and explicit conceptual relationships to correctly identify and perform the proper transformations on the given input. As an English language example, “you” becomes “i”, “me” becomes “you”, and so forth.

Example Usage (Java/C#)

```
String result = _core.ApiBrain().TransformNarrativeMode(“your father”);
```

Example Usage (LUA)

```
result = silvia.brain.transformNarrativeMode(“your father”) where “your  
father” is returned as “my father”
```

Parameters

input	The string representing the input to be processed.
-------	--

Returns

The string representing the output produced by SILVIA's transformNarrativeMode function.

[silvia.brain.executePostEvents](#)

Invokes the core SILVIA algorithms to execute post-response events and scripts

Description

This function invokes AI run-time to execute any outstanding events or scripts that have been registered by a call to [silvia.brain.getResponse](#).

Example Usage (Java)

```
boolean result = _core.ApiBrain().ExecutePostEvents();
```

Example Usage (C#)

```
bool result = _core.ApiBrain().ExecutePostEvents();
```

Example Usage (LUA)

```
result = silvia.brain.executePostEvents()
```

Parameters

none

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.getResponse](#)

[silvia.brain.setJump](#)

Sets the group and name of the behavior to "jump" to next

Description

This function forces the invocation of a particular behavior. This can be executed immediately in the script, or can be placed in a timed function for later execution.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetJump("movies", "likes2001", 1.0f);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetJump("movies", "likes2001", 1.0f);
```

Example Usage (LUA)

```
success = silvia.brain.setJump("movies", "likes2001", 1.0)
```

Parameters

group	The string representing the group identifier of the behavior.
name	The string representing the name identifier of the behavior.
probability	The coefficient between 0.0 and 1.0 representing the probability that the jump will occur.

Returns

The boolean success or failure of the operation.

silvia.brain.setBypassResponse

On the next input, the "GetResponse" function will ignore the input and jump to the given behavior

Description

This method forces the invocation of a particular behavior. However, this invocation does not occur immediately, but on the next input given by the user.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetBypassResponse("game", "assume_yes_answer");
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetBypassResponse("game", "assume_yes_answer");
```

Example Usage (LUA)

```
success = silvia.brain.setBypassResponse("game", "assume_yes_answer")
```

Parameters

group	The string representing the group identifier of the behavior.
name	The string representing the name identifier of the behavior.

Returns

The boolean success or failure of the operation.

`silvia.brain.removeStopwords`

Takes an input string and removes the stopwords concepts, returning the result

Description

Certain conceptual processing may require only the most important “high-level” concepts to be passed in from user input. This utility function automatically produces an output string from an input string of concepts that has the “low-level” stopwords removed. These stopwords are found in the “linguistics” folder of the application.

The stopwords table needs to be loaded from a file when first creating a SILVIA core. Depending on the platform, the parameter will either be a stopwords file name or a pre-read list of Strings passed in to the `CreateCore` function. If this is left null, there will be no stopwords to consider for removal or other related internal processing.

The second parameter in `ApiBrain().RemoveStopwords` is a list of concepts in string form that should be left in the returned result, even if they exist in the stopwords table.

For instance, the stopword "when" may be important for this particular utterance, so you may specify this in the second parameter and it will not be removed automatically.

Example Usage (Java/C#)

```
String concepts = "where is the state of california"; String importantconcepts =  
_core.ApiBrain().RemoveStopwords(concepts, "where what who why when how");
```

Example Usage (LUA)

```
concepts = "where is the state of alabama" importantconcepts = silvia.brain.removeStopwords(concepts,  
"where what who why when how")
```

Parameters

<code>concepts</code>	The string of space-separated concepts to be stripped of stopwords
<code>include</code>	The string of space-separated concepts to be considered important, overriding any status as stopwords

Returns

The string of space-separated concepts, with removed stopwords.

[silvia.brain.setDynamicAttraction](#)

Modifies the state of the SILVIA Core algorithms, changing the “attaction” parameter for dynamic output

Description

This method helps control SILVIA’s dynamic output generation, and sets the integer multiplier for SILVIA’s attraction to key concepts. The higher the attraction, the more SILVIA gravitates toward explicitly defined concepts during output generation.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetDynamicAttraction(10);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetDynamicAttraction(10);
```

Example Usage (LUA)

```
success = silvia.brain.setDynamicAttraction(10)
```

Parameters

The numeric attraction value, greater than 0

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.setDynamicDepth](#)
- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.setDynamicAdaptation](#)
- [silvia.brain.generateDynamic](#)
- [silvia.brain.generateDynamicLimited](#)

[silvia.brain.setDynamicDepth](#)

Modifies the state of the SILVIA Core algorithms, changing the “depth” parameter for dynamic output

Description

This method helps control SILVIA’s dynamic output generation, and sets the integer depth of SILVIA’s lexical traversal algorithm. The higher the number, the more rigid SILVIA will be in following implicit grammatical rules. In

simple terms, higher numbers will produce more rigidly logical output, whereas lower numbers will result in output that is closer to free- association.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetDynamicDepth(4);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetDynamicDepth(4);
```

Example Usage (LUA)

```
success = silvia.brain.setDynamicDepth(4)
```

Parameters

The numeric depth value, greater than 0

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.setDynamicAttraction](#)
- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.setDynamicAdaptation](#)
- [silvia.brain.generateDynamic](#)
- [silvia.brain.generateDynamicLimited](#)

silvia.brain.setDynamicFalloffDepth

Modifies the state of the SILVIA Core algorithms, changing the “falloff depth” parameter for dynamic output

Description

This method helps control SILVIA’s dynamic output generation, and sets the integer falloff depth of SILVIA’s lexical traversal algorithm. This number determines the depth at which a statistical falloff begins. A higher number, closer to the set dynamic depth value, will give little or no room for falloff, giving completely rigid output generation within the range. A value that is significantly lower than the basic depth value will give the SILVIA algorithms room to vary within the outside range defined by the basic depth value, with the amount of variation to be determined by a “falloff” multiplier between 0.0 and 1.0

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetDynamicFalloffDepth(2);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetDynamicFalloffDepth(2);
```

Example Usage (LUA)

```
success = silvia.brain.setDynamicFalloffDepth(2)
```

Parameters

The numeric falloff depth value, greater than 0, <= dynamic depth

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.setDynamicAttraction](#)
- [silvia.brain.setDynamicDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.setDynamicAdaptation](#)
- [silvia.brain.generateDynamic](#)
- [silvia.brain.generateDynamicLimited](#)

silvia.brain.setDynamicFalloff

Modifies the state of the SILVIA Core algorithms, changing the “falloff” parameter for dynamic output

Description

This method helps control SILVIA’s dynamic output generation, and sets the numeric falloff of SILVIA’s lexical traversal algorithm. This number determines how statistical probability will decay between the range of “falloffDepth” and “depth”. A value of 1.0 will produce no statistical falloff, whereas a value of 0.5 will multiply the statistical likelihood of a rigid conceptual match by 0.5 for each unit of distance past falloffDepth. In simple terms, this value provides a mechanism for lexical rigidity to decay within the process of output construction, giving finer control over SILVIA’s output construction algorithms.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetDynamicFalloff(0.9f);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetDynamicFalloff(0.9f);
```

Example Usage (LUA)

```
success = silvia.brain.setDynamicFalloff(0.9)
```

Parameters

The numeric falloff value, between 0.0 and 1.0

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.setDynamicAttraction](#)
- [silvia.brain.setDynamicDepth](#)
- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicAdaptation](#)
- [silvia.brain.generateDynamic](#)
- [silvia.brain.generateDynamicLimited](#)

silvia.brain.setDynamicAdaptation

Modifies the state of the SILVIA Core algorithms, setting the “adaptation” parameter for dynamic output

Description

This method helps control SILVIA’s dynamic output generation, and sets the boolean adaptation of SILVIA’s lexical traversal algorithm. This boolean determines whether SILVIA automatically and iteratively adapts and reduces rigidity during output construction to insure that all concepts are expressed. In simple terms, when this is enabled, SILVIA is allowed to adaptively become lexically sloppy if necessary to make sure that all key concepts are expressed. If not enabled, SILVIA may not be able to combine all of the required concepts of a particular expression. This inability will be due to a limited knowledge base, but in these cases, SILVIA will still attempt to express something that is both conceptually relevant and lexically correct containing as many of the required concepts as possible. By default, adaptation is disabled.

Example Usage (Java/C#)

```
_core.ApiBrain().SetDynamicAdaptation(true);
```

Example Usage (LUA)

```
silvia.brain.setDynamicAdaptation(true)
```

Parameters

The boolean value enabling or disabling adaptation

Returns

None.

See Also

- [silvia.brain.setDynamicAttraction](#)

- [silvia.brain.setDynamicDepth](#)

- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.generateDynamic](#)
- [silvia.brain.generateDynamicLimited](#)

silvia.brain.generateDynamic

Invokes the core SILVIA algorithms to generate a dynamic output

Description

This method invokes SILVIA’s dynamic output generation. Two strings are passed to the method, the first containing the space-separated concepts for expression, and the second containing the space-separated concepts for exclusion. For instance, one might call the method with “artificial intelligence” as the first string, and “movies books” as the second. This will force SILVIA to generate something to say about artificial intelligence, but she will not say anything about artificial intelligence in movies or books. When invoked, this method uses the parameters set by the “setDynamic*” functions.

Note that within the SILVIA Training UI, dynamic concept sets are specified within Exuders using “[” and “]” opening and closing brackets, and excluded concepts are specified with the “#” symbol, used to open and close blocks of excluded concepts. For instance, the above example would be specified within the Exuder as: [artificial intelligence #movies books#]

Alternately, if the DYNAMIC check is enabled, the Exuder would not need the “[” and “]” symbols, and would read: artificial intelligence #movies books#

Example Usage (Java/C#)

```
String output = _core.ApiBrain().GenerateDynamic("rock and roll", "keith richards");
```

Example Usage (LUA)

```
output = silvia.brain.generateDynamic("rock and roll", "keith richards")
```

Parameters

concepts	the string containing the set of concepts to be expressed in the output
excluded	the string containing the set of concepts to exclude from the output

Returns

A string containing the dynamically generated output.

See Also

- [silvia.brain.setDynamicAttraction](#)
- [silvia.brain.setDynamicDepth](#)

- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.setDynamicAdaptation](#)
- [silvia.brain.generateDynamicLimited](#)

silvia.brain.generateDynamicLimited

Invokes the core SILVIA algorithms to generate a dynamic output from a limited set of behaviors

Description

This method invokes SILVIA's dynamic output generation. Two strings are passed to the method, the first containing the space-separated concepts for expression, and the second containing the space-separated concepts for exclusion. For instance, one might call the method with "artificial intelligence" as the first string, and "movies books" as the second. This will force SILVIA to generate something to say about artificial intelligence, but she will not say anything about artificial intelligence in movies or books. When invoked, this method uses the parameters set by the "setDynamic*" functions.

However, unlike the ordinary generateDynamic method, this generateDynamicLimited method also takes in an optional group and name string. If a non-nil group value is specified, the dynamic output will ONLY draw from behaviors/Exuders within that group for generating output. If a non-nil name is specified, then the output generation is further limited to use ONLY that single group/name specified behavior.

Note that within the SILVIA Training UI, dynamic concept sets are specified within Exuders using "[" and "]" opening and closing brackets, and excluded concepts are specified with the "#" symbol, used to open and close blocks of excluded concepts. For instance, the above example would be specified within the Exuder as: [artificial intelligence #movies books#]

Alternately, if the DYNAMIC check is enabled, the Exuder would not need the "[" and "]" symbols, and would read: artificial intelligence #movies books#

Example Usage (Java/C#)

```
String output = _core.ApiBrain().GenerateDynamicLimited("rock and roll", "keith richards", "music",
"rock_Exuders");
```

Example Usage (LUA)

```
output = silvia.brain.generateDynamicLimited("rock and roll", "keith richards", "music", "rock_Exuders")
```

Parameters

concepts	the string containing the set of concepts to be expressed in the output
excluded	the string containing the set of concepts to exclude from the output

group	the optional string value specifying the behavior group to be used
name	the optional string value specifying the behavior name to be used

Returns

A string containing the dynamically generated output.

See Also

- [silvia.brain.setDynamicAttraction](#)
- [silvia.brain.setDynamicDepth](#)
- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.setDynamicAdaptation](#)
- [silvia.brain.generateDynamic](#)

silvia.brain.generateDynamicFromMemory

Invokes the core SILVIA algorithms to generate a dynamic output from the feedback stack

Description

This method invokes SILVIA's dynamic output generation. Two strings are passed to the method, the first containing the space-separated concepts for expression, and the second containing the space-separated concepts for exclusion. For instance, one might call the method with "artificial intelligence" as the first string, and "movies books" as the second. This will force SILVIA to generate something to say about artificial intelligence, but she will not say anything about artificial intelligence in movies or books. When invoked, this method uses the parameters set by the "setDynamic*" functions.

Note that within the SILVIA Training UI, dynamic concept sets are specified within Exuders using "[" and "]" opening and closing brackets, and excluded concepts are specified with the "#" symbol, used to open and close blocks of excluded concepts. For instance, the above example would be specified within the Exuder as: [artificial intelligence #movies books#]

Alternately, if the DYNAMIC check is enabled, the Exuder would not need the "[" and "]" symbols, and would read: artificial intelligence #movies books#

Example Usage (Java/C#)

```
String output = _core.ApiBrain().GenerateDynamicFromMemory("leslie", "rock and roll", "keith richards", true);
```

Example Usage (LUA)

```
output = silvia.brain.generateDynamicFromMemory("leslie", "rock and roll", "keith richards", true)
```

Parameters

user	the string specifying the username tag for memory searches
concepts	the string containing the set of concepts to be expressed in the output
excluded	the string containing the set of concepts to exclude from the output
invert	the boolean value to invert the first/second person of the pronouns

Returns

A string containing the dynamically generated output.

See Also

- [silvia.brain.setDynamicAttraction](#)
- [silvia.brain.setDynamicDepth](#)
- [silvia.brain.setDynamicFalloffDepth](#)
- [silvia.brain.setDynamicFalloff](#)
- [silvia.brain.setDynamicAdaptation](#)

silvia.brain.dynamicHasAllConceptsInOne

Returns a true or false indication of result suitability after a call to generateDynamic

Description

This method is called after invocation of SILVIA's dynamic output generation. The boolean value returned indicates if at least one Exuder considered has all of the important, non- stopword concepts in the string that was passed to the previous call to "generateDynamic". This is particularly useful if the application developer needs to filter dynamically generated results based on accuracy.

For example, if an Exuder exists containing the concepts "my favorite color is blue", and the concepts "favorite color" are passed to the "generateDynamic" function, the result of THIS function will be true.

However, if "favorite" and "color" exist in various Exuders, but never appear together within the same Exuder, the result of THIS function will be false.

Example Usage (Java/C#)

```
bool hasAllConcepts = _core.ApiBrain().DynamicHasAllConceptsInOne();
```

Example Usage (LUA)

```
hasAllConcepts = silvia.brain.dynamicHasAllConceptsInOne()
```

Parameters

none

Returns

A boolean value representing the condition.

See Also

- [silvia.brain.generateDynamic](#)

silvia.brain.setAbsorberThreshold

Invokes the core SILVIA algorithms to change the “threshold” for Absorber acceptance

Description

This method helps control SILVIA’s ability to reject user input that falls too far outside of the AI’s ability to answer. For instance, a threshold of 0.5 means that the input must match roughly half of the conceptual/linguistic content of an Absorber before that Absorber would be considered at all as possibly relevant. If no relevant Absorbers are found for consideration, then either the behavior named “default” is used to generate a resultant output, or if no behavior named “default” is available, SILVIA will output one of several possible phrases expressing her inability to understand the input. The default threshold value is 0.33

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetAbsorberThreshold(0.25f);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetAbsorberThreshold(0.25f);
```

Example Usage (LUA)

```
success = silvia.brain.setAbsorberThreshold(0.25)
```

Parameters

The numeric coefficient value used to limit acceptance/rejection of input

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.setReusableThreshold](#)

silvia.brain.setReusableThreshold

Invokes the core SILVIA algorithms to change the “threshold” for Exuder matching acceptance

Description

This method is a companion to `setAbsorberThreshold` because it helps control SILVIA’s ability to accept or reject Exuders that conceptially match the user input. If this threshold is set low enough, AND the user creates input that is rejected by the Absorber matching algorithms, alternative responses may be spontaneously generated by SILVIA using the reusable Exuders that have some conceptual and contextual relationship to the user input. Note that only “interesting” topical concepts are considered, so as to provide reasonable triggers and to minimize irrelevant matches.

As with Absorber matching, if no suitable Exuders are found using this line of interpretive defense, SILVIA will either drop to an output using the “default” behavior, or barring such a behavior’s existence, will output one of several possible phrases expressing her inability to understand the input. The default threshold value is 100.0, indicating that this feature is basically “off”.

Example Usage (Java)

```
boolean success = _core.ApiBrain().SetReusableThreshold(1.5f);
```

Example Usage (C#)

```
bool success = _core.ApiBrain().SetReusableThreshold(1.5f);
```

Example Usage (LUA)

```
success = silvia.brain.setReusableThreshold(1.5)
```

Parameters

The numeric value used to limit acceptance/rejection of input when compared to Exuders

Returns

The boolean success or failure of the operation.

See Also

- [silvia.brain.setAbsorberThreshold](#)

silvia.brain.setAddressByName

Enables or disables the SILVIA core's requirement to be addressed by name before responding

Description

A desirable operating mode is one where SILVIA can listen to user's while they speak, but where the application will only accept input if SILVIA is addressed directly by name. This allows users to speak freely, with SILVIA running in the background and NOT responding. To elicit a response, one would have to begin an utterance with the application's name, which defaults to "silvia".

Example Usage (Java/C#)

```
_core.ApiBrain().SetAddressByName(true);
```

Example Usage (LUA)

```
silvia.brain.setAddressByName(true)
```

Parameters

enabled	a boolean value to enable or disable this mode in the SILVIA core
---------	---

Returns

None.

See Also

- [silvia.brain.getAddressByName](#)

silvia.brain.getAddressByName

Returns the boolean enabled/disabled status of the SILVIA core's AddressByName mode

Description

A desirable operating mode is one where SILVIA can listen to user's while they speak, but where the application will only accept input if SILVIA is addressed directly by name. This function returns the boolean status of this mode.

Example Usage (Java)

```
boolean addressMe = _core.ApiBrain().GetAddressByName();
```

Example Usage (C#)

```
bool addressMe = _core.ApiBrain().GetAddressByName();
```

Example Usage (LUA)

```
addressMe = silvia.brain.getAddressByName()
```

Parameters

none

Returns

The boolean value of the AddressByName status.

See Also

- [silvia.brain.setAddressByName](#)

silvia.brain.setAllEars

Enables or disables the SILVIA core's ability to respond to the next input

Description

A desirable operating mode is one where SILVIA can listen to user's while they speak, but where the application will only accept input if SILVIA is addressed directly by name. This allows users to speak freely, with SILVIA running in the background and NOT responding. To elicit a response, one would have to begin an utterance with the application's name, which defaults to "silvia". This function, used in combination with an enabled "AddressMe", can enable or disable SILVIA's ability to use the next input "as is".

Example Usage (Java/C#)

```
_core.ApiBrain().SetAllEars(true);
```

Example Usage (LUA)

```
silvia.brain.setAllEars(true)
```

Parameters

enabled	a boolean value to enable or disable this mode in the SILVIA core
---------	---

Returns

None.

See Also

- [silvia.brain.setAddressByName](#)
- [silvia.brain.getAddressByName](#)

- [silvia.brain.getAllEars](#)

[silvia.brain.getAllEars](#)

Returns the boolean enabled/disabled status of the SILVIA core's AllEars mode

Description

A desirable operating mode is one where SILVIA can listen to user's while they speak, but where the application will only accept input if SILVIA is addressed directly by name. The "AllEars" component determines if SILVIA can currently accept all input, and this function returns the boolean status of this mode.

Example Usage (Java)

```
boolean allEars = _core.ApiBrain().GetAllEars();
```

Example Usage (C#)

```
bool allEars = _core.ApiBrain().GetAllEars();
```

Example Usage (LUA)

```
allEars = silvia.brain.getAllEars()
```

Parameters

none

Returns

The boolean value of the AllEars status.

See Also

- [silvia.brain.setAddressByName](#)
- [silvia.brain.getAddressByName](#)
- [silvia.brain.setAllEars](#)

[silvia.brain.loadCommandAssembly](#)

Loads the name command assembly so that the functions are available to SILVIA

Description

Custom functionality may be created via one or more plugins. This command allows for the scripted loading of such plugins. Usually, this method will be called from a boot behavior's post Exuder script. This is supported on interpreted .NET/Mono enabled platforms only.

Example Usage (C#)

```
bool loaded = _core.ApiBrain().LoadCommandAssembly("silvia_commands", "silvia_command",  
"sv_command");
```

Example Usage (LUA)

```
loaded = silvia.brain.loadCommandAssembly("silvia_commands", "silvia_command", "sv_command")
```

Parameters

assembly	a string representing the name of the assembly (dll or so) to be loaded
namespace	a string specifying the namespace where the class/methods are to be found
class	a string specifying the named class that contains the methods to be accessed

Returns

The boolean value representing the success or failure of the assembly's loading.

See Also

none

silvia.data

This class is available as both a C# class and a LUA table, and provides the functions for comprehensive read/write access to SILVIA's behavior and concept data.

Summary

<u>silvia.data</u>	This class is available as both a C# class and a LUA table, and provides the functions for comprehensive read/write access to SILVIA's behavior and concept data.
---------------------------	---

FUNCTIONS

<u>silvia.data.clear</u>	Clears and initializes the SILVIA data in the active SILVIA Core
<u>silvia.data.setBinding</u>	Creates a conceptual binding between two concepts

[silvia.data.setBindingTypeModifier](#) Alters the numeric weighting modifier for a particular conceptual relationship type

<u>silvia.data.getBindingTypeModifier</u>	Returns the numeric weighting modifier for a particular conceptual relationship type
<u>silvia.data.testBinding</u>	Determines if two concepts are bound, and in what direction
<u>silvia.data.getBoundConcept</u>	Returns the specified bound concept, or nil if none found
<u>silvia.data.getAllBoundConcepts</u>	Returns the all of the bound concepts and binding types for given concept
<u>silvia.data.tuneConcepts</u>	Statistically tunes and balances conceptual data based on a variety of factors
<u>silvia.data.cleanConcepts</u>	Removes unused and ontologically disconnected concepts from memory
<u>silvia.data.getBehaviorID</u>	Gets or creates a behavior with the given group and name
<u>silvia.data.getBehaviorIDFromIndex</u>	Gets a behavior ID from the give 0-base index
<u>silvia.data.getBehaviorCount</u>	Returns the total number of behaviors in the current brain data
<u>silvia.data.getBehaviorGroup</u>	Returns the group of the given behavior
<u>silvia.data.getBehaviorSubGroup</u>	Returns the sub-group of the given behavior
<u>silvia.data.getBehaviorName</u>	Returns the name of the given behavior
<u>silvia.data.removeBehavior</u>	Deletes the behavior with the given ID
<u>silvia.data.addAbsorber</u>	Creates a new Absorber with the given user input pattern
<u>silvia.data.setAbsorberText</u>	Modifies a given Absorber with the given AI input pattern
<u>silvia.data.getAbsorberText</u>	Returns a given Absorber's AI input pattern
<u>silvia.data.removeAbsorber</u>	Deletes the Absorber represented by the given behavior and Absorber ID pair
<u>silvia.data.addExuder</u>	Creates a new Exuder with the given AI output pattern
<u>silvia.data.setExuderText</u>	Modifies a given Exuder with the given AI output pattern
<u>silvia.data.getExuderText</u>	Returns a given Exuder's AI output pattern
<u>silvia.data.removeExuder</u>	Deletes the Exuder represented by the given behavior and Exuder ID pair
<u>silvia.data.setBehaviorScript</u>	Sets the programmatic script content for the behavior.
<u>silvia.data.setExuderscript</u>	Sets the programmatic script content for the given indexed Exuder within the given behavior.

<u>silvia.data.setBehaviorData</u>	Sets the arbitrary, application specific string data content for the behavior.
<u>silvia.data.getBehaviorData</u>	Returns the arbitrary, application specific string data content for the behavior.
<u>silvia.data.setBehaviorSecurityLevel</u>	Sets the numeric integer security level for the given behavior.
<u>silvia.data.getBehaviorSecurityLevel</u>	Returns the integer security level for the given behavior.

<u>silvia.data.setAbsorberData</u>	Sets the arbitrary, application specific string data content for the given Absorber.
<u>silvia.data.getAbsorberData</u>	Returns the arbitrary, application specific string data content for the given Absorber.
<u>silvia.data.setExuderData</u>	Sets the arbitrary, application specific string data content for the given Exuder.
<u>silvia.data.getExuderData</u>	Returns the arbitrary, application specific string data content for the given Exuder.
<u>silvia.data.setExudersecurityLevel</u>	Sets the numeric integer security level for the given Exuder.
<u>silvia.data.getExudersecurityLevel</u>	Returns the integer security level for the given Exuder
<u>silvia.data.setAbsorberExact</u>	Enables or disables the Absorber's requirement for an exact conceptual match on the user input.
<u>silvia.data.getAbsorberExact</u>	Returns a true or false value of the Absorber's requirement for an exact conceptual match on the user input.
<u>silvia.data.setAbsorberReject</u>	Enables or disables the Absorber's use as a rejection filter
<u>silvia.data.getAbsorberReject</u>	Returns the state of the Absorber's use as a rejection filter
<u>silvia.data.setExuderExact</u>	Enables or disables the AI requirement to express (output) exactly what is contained in the Exuder.
<u>silvia.data.getExuderExact</u>	Returns the boolean state of the AI requirement to express (output) exactly what is contained in the Exuder.
<u>silvia.data.setExuderReuse</u>	Enables or disables the AI's ability to draw on a given Exuder in generating dynamic output.
<u>silvia.data.getExuderReuse</u>	Returns the AI's ability to draw on a given Exuder in generating dynamic output.
<u>silvia.data.setExuderDynamic</u>	Enables or disables the AI's ability to use the concepts in an Exuder to generate new, dynamic output.
<u>silvia.data.getExuderDynamic</u>	Returns the state of the AI's ability to use the concepts in a particular Exuder to generate new, dynamic output.
<u>silvia.data.setExuderContext</u>	Sets the given Exuder's conceptual context field
<u>silvia.data.getExuderContext</u>	Returns the given Exuder's conceptual context field
<u>silvia.data.getBehaviorCreatedYear</u>	Gets the given behavior's creation date "year" value
<u>silvia.data.getBehaviorCreatedMonth</u>	Gets the given behavior's creation date "month" value

[silvia.data.getBehaviorCreatedDay](#) Gets the given behavior's creation date "day" value

[silvia.data.getBehaviorCreatedHour](#) Gets the given behavior's creation date "hour" value

[silvia.data.getBehaviorCreatedMinute](#) Gets the given behavior's creation date "minute" value

[silvia.data.getBehaviorLastModifiedYe](#) Gets the given behavior's last modified date "year" value

ar

silvia.data.getBehaviorLastModifiedMonth Gets the given behavior's last modified date "month" value

silvia.data.getBehaviorLastModifiedDay Gets the given behavior's last modified date "day" value

y

silvia.data.getBehaviorLastModifiedHour Gets the given behavior's last modified date "hour" value

silvia.data.getBehaviorLastModifiedMinute Gets the given behavior's last modified date "minute" value

silvia.data.addResponse Adds a paired input/output response to the behavior data

silvia.data.setContext Sets the context for a previously learned response

silvia.data.setReuse Sets the previously learned response as re-useable or not

silvia.data.setBehaviorName Sets the name for the most recently learned behavior

silvia.data.setBehaviorGroup Sets the group for the most recently learned behavior

silvia.data.setBehaviorSubGroup Sets the sub-group for the most recently learned behavior

Functions

silvia.data.clear

Clears and initialized the SILVIA data in the active SILVIA Core

Description

This function deletes all knowledge from the current SILVIA brain, essentially creating a new "blank slate" in SILVIA's memory.

Example Usage (C#)

```
bool success = _core.ApiData().Clear()
```

Example Usage (LUA)

```
success = silvia.data.clear()
```

Parameters

none

Returns

The boolean success or failure of the operation.

silvia.data.setBinding

Creates a conceptual binding between two concepts

Description

This function adds knowledge to the current SILVIA brain by connecting two concepts with a binding of a specific type.

Example Usage (C#)

```
bool success = _core.ApiData().SetBinding("tokyo", "japan", "location");
```

Example Usage (LUA)

```
success = silvia.data.setBinding("tokyo", "japan", "location")
```

Parameters

concept	The string or variable name representing the base concept of the binding.
bound	The string or variable name representing the concept to be bound.
type	The string representing the binding type.

The current exposed lexical binding types are

- "root"
- "synonym", "synonymSibling"
- "antonym"
- "similar", "similarSibling"
- "different"
- "related", "relatedSibling"
- "unrelated"
- "child", "sibling", "parent"
- "plural", "singular"
- "misspelling", "correction"
- "verbal", "written"

The current exposed conceptual binding types are loosely based on the MIT MediaLab's ConceptNet

- "relatedTo", "thematic", "superThematic"

- "isA", "propertyOf", "partOf", "madeOf", "definedAs"
- "capableOf"

- "prerequisite", "firstEvent", "eventOf", "lastEvent"
- "location"
- "effect", "desirousEffect"
- "usedFor", "action"
- "motivation", "desire"

And finally

- "unknown", "invalid"

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.testBinding](#)
- [silvia.data.getBoundConcept](#)
- <silvia.data.getAllBoundConcept>

silvia.data.setBindingTypeModifier

Alters the numeric weighting modifier for a particular conceptual relationship type

Description

This function alters the connection strength between concepts bound by the given type. Note that if the binding type does not exist, it will be created and given the modifier value.

Example Usage (C#)

```
bool success = _core.ApiData().SetBindingTypeModifier("child", 0.65f);
```

Example Usage (LUA)

```
success = silvia.data.setBindingTypeModifier("child", 0.65)
```

Parameters

type	The string representing the binding type or a variable containing the name of the type
modifier	The number representing a scalar multiplier for the connection type, with a value of 0.0 indicating a halt condition

The current exposed lexical binding types are

- "root"
- "synonym", "synonymSibling"
- "antonym"
- "similar", "similarSibling"
- "different"
- "related", "relatedSibling"
- "unrelated"
- "child", "sibling", "parent"
- "plural", "singular"
- "misspelling", "correction"
- "verbal", "written"

The current exposed conceptual binding types are loosely based on the MIT MediaLab's ConceptNet

- "relatedTo", "thematic", "superThematic"
- "isA", "propertyOf", "partOf", "madeOf", "definedAs"
- "capableOf"
- "prerequisite", "firstEvent", "eventOf", "lastEvent"
- "location"
- "effect", "desirousEffect"
- "usedFor", "action"
- "motivation", "desire"

And finally

- "unknown", "invalid"

Returns

The boolean success or failure of the operation. The method fails if the binding type is nil or zero-length.

See Also

- [silvia.data.setBinding](#)
- [silvia.data.testBinding](#)
- [silvia.data.getBoundConcept](#)
- <silvia.data.getAllBoundConcept>
- [silvia.data.getBindingTypeModifier](#)

[silvia.data.getBindingTypeModifier](#)

Returns the numeric weighting modifier for a particular conceptual relationship type

Description

This function returns the connection strength between concepts bound by the given type. Note that if the binding type does not exist, it will be created, and given a default modifier of the 0.0 halt condition.

Example Usage (C#)

```
float modifier = _core.ApiData().GetBindingTypeModifier("synonym");
```

Example Usage (LUA)

```
modifier = silvia.data.getBindingTypeModifier("synonym")
```

Parameters

type	The string representing the binding type or a variable containing the name of the type
------	--

The current exposed lexical binding types are

- "root"
- "synonym", "synonymSibling"
- "antonym"
- "similar", "similarSibling"
- "different"
- "related", "relatedSibling"
- "unrelated"
- "child", "sibling", "parent"
- "plural", "singular"
- "misspelling", "correction"
- "verbal", "written"

The current exposed conceptual binding types are loosely based on the MIT MediaLab's ConceptNet

- "relatedTo", "thematic", "superThematic"
- "isA", "propertyOf", "partOf", "madeOf", "definedAs"
- "capableOf"
- "prerequisite", "firstEvent", "eventOf", "lastEvent"
- "location"
- "effect", "desirousEffect"
- "usedFor", "action"
- "motivation", "desire"

And finally

- "unknown", "invalid"

Returns

modifier	The number representing a scalar multiplier for the connection type, with a value of 0.0 indicating a halt condition
----------	--

See Also

- [silvia.data.setBinding](#)
- [silvia.data.testBinding](#)
- [silvia.data.getBoundConcept](#)
- `<silvia.data.getAllBoundConcept>`
- [silvia.data.setBindingTypeModifier](#)

`silvia.data.testBinding`

Determines if two concepts are bound, and in what direction

Description

This function tests knowledge within the current SILVIA brain by returning a string representing a valid a binding direction or an “invalid”.

Example Usage (C#)

```
String direction = _core.ApiData().TestBinding("tokyo", "japan", "location");
```

Example Usage (LUA)

```
direction = silvia.data.testBinding("tokyo", "japan", "location")
```

Parameters

concept	The string or variable name representing the base concept of the potential binding.
bound	The string or variable name representing the concept that is potentially bound.
type	The string representing the binding type to be tested. “unknown” is used for generic tests.

The current exposed lexical binding types are

- "root"
- "synonym", "synonymSibling"

- "antonym"
- "similar", "similarSibling"
- "different"
- "related", "relatedSibling"
- "unrelated"
- "child", "sibling", "parent"
- "plural", "singular"
- "misspelling", "correction"
- "verbal", "written"

The current exposed conceptual binding types are loosely based on the MIT MediaLab's ConceptNet

- "relatedTo", "thematic", "superThematic"
- "isA", "propertyOf", "partOf", "madeOf", "definedAs"
- "capableOf"
- "prerequisite", "firstEvent", "eventOf", "lastEvent"
- "location"
- "effect", "desirousEffect"
- "usedFor", "action"
- "motivation", "desire"

And finally

- "unknown", "invalid"

Returns

A string representing a binding direction of "forward", "reverse", or "invalid" if no binding was found.

See Also

- [silvia.data.setBinding](#)
- [silvia.data.getBoundConcept](#)
- [silvia.data.getAllBoundConcepts](#)

[silvia.data.getBoundConcept](#)

Returns the specified bound concept, or nil if none found

Description

This function tests knowledge within the current SILVIA brain by returning a string representing one of the bound concepts of the specified type. If no valid bound concept is found, the function returns nil.

Example Usage (C#)

```
String boundConcept = _core.ApiData().GetBoundConcept("tokyo", "location");
```

Example Usage (LUA)

```
boundConcept = silvia.data.getBoundConcept("tokyo", "location")
```

Parameters

concept	The string or variable name representing the base concept of the potential binding.
type	The string representing the binding type to be tested. "unknown" is used for generic tests.

The current exposed lexical binding types are

- "root"
- "synonym", "synonymSibling"
- "antonym"
- "similar", "similarSibling"
- "different"
- "related", "relatedSibling"
- "unrelated"
- "child", "sibling", "parent"
- "plural", "singular"
- "misspelling", "correction"
- "verbal", "written"

The current exposed conceptual binding types are loosely based on the MIT MediaLab's ConceptNet

- "relatedTo", "thematic", "superThematic"
- "isA", "propertyOf", "partOf", "madeOf", "definedAs"
- "capableOf"
- "prerequisite", "firstEvent", "eventOf", "lastEvent"
- "location"
- "effect", "desirousEffect"
- "usedFor", "action"
- "motivation", "desire"

And finally

- "unknown", "invalid"

Returns

A string representing a concept that is bound by the type to the original concept. A nil is returned if no valid match

is found. If more than one concept is bound by a binding of type, the bound concept returned is selected randomly.

See Also

- [silvia.data.setBinding](#)
- [silvia.data.testBinding](#)
- [silvia.data.getAllBoundConcepts](#)

silvia.data.getAllBoundConcepts

Returns the all of the bound concepts and binding types for given concept

Description

This function tests knowledge within the current SILVIA brain by returning a string array representing one or more bound concepts attached to the specified root concept type. The return format is in paired strings, with even strings (starting at 0) containing the bound concept names, and the alternating odd strings (starting at 1) containing the binding type as a string. If no valid bound concepts are found, the function returns nil.

Example Usage (C#)

```
String[] bindings = _core.ApiData().GetAllBoundConcepts("dog"); if(bindings != null)
{ String firstBound = String[0]; String firstType = String[1]; }
```

Example Usage (LUA)

```
bindings = silvia.data.getAllBoundConcepts("dog") return bindings[0], bindings[1]
```

Parameters

concept	The string or variable name representing the base concept of the potential bindings.
---------	--

The current exposed lexical binding types are

- "root"
- "synonym", "synonymSibling"
- "antonym"
- "similar", "similarSibling"
- "different"
- "related", "relatedSibling"
- "unrelated"
- "child", "sibling", "parent"
- "plural", "singular"
- "misspelling", "correction"
- "verbal", "written"

The current exposed conceptual binding types are loosely based on the MIT MediaLab's ConceptNet

- "relatedTo", "thematic", "superThematic"
- "isA", "propertyOf", "partOf", "madeOf", "definedAs"
- "capableOf"
- "prerequisite", "firstEvent", "eventOf", "lastEvent"
- "location"
- "effect", "desirousEffect"
- "usedFor", "action"
- "motivation", "desire"

And finally

- "unknown", "invalid"

Returns

A series of string pairs representing a bound concept, followed by its type A nil is returned if no valid match is found.

See Also

- [silvia.data.setBinding](#)
- [silvia.data.testBinding](#)
- [silvia.data.getBoundConcept](#)

[silvia.data.tuneConcepts](#)

Statistically tunes and balances conceptual data based on a variety of factors

Description

This function modifies knowledge in the current SILVIA brain by using statistical analysis, lexical information, and other criteria to produce well-balanced and responsive mathematical values for the current collection of concepts in the SILVIA brain. This function is commonly invoked when finalized brain data needs to be balanced for interactive use, and the tuned conceptual data is saved with trained brain files.

Example Usage (C#)

```
bool success = _core.ApiData().TuneConcepts(true, true, false);
```

Example Usage (LUA)

```
success = silvia.data.tuneConcepts(true, true, false)
```

Parameters

<code>useAbsorbers</code>	The boolean flag to determine if Absorber data is considered during the tuning process.
<code>AbsorberAsDocument</code>	The boolean flag to separate Absorbers as unique documents during the tuning process.
<code>useExuders</code>	The boolean flag to determine if Exuder data is considered during the tuning process.

Returns

The boolean success or failure of the operation.

[silvia.data.cleanConcepts](#)

Removes unused and ontologically disconnected concepts from memory

Description

This function is an optimization procedure that removes any extraneous conceptual data not pertinent to the currently loaded knowledge base. During the course of development, concepts may be created or loaded by trainers “in-passing”, and can potentially end up as useless, disconnected data. Invoking this function “cleans up” the concept data so that only useful and used information is retained.

Example Usage (C#)

```
bool success = _core.ApiData().CleanConcepts();
```

Example Usage (LUA)

```
success = silvia.data.cleanConcepts()
```

Parameters

none

Returns

The boolean success or failure of the operation.

[silvia.data.getBehaviorID](#)

Gets or creates a behavior with the given group and name

Description

This function either returns an existing ID value if the given behavior exists, or creates a new behavior with the given group and name, modifying the current SILVIA brain to do so. The returned integer ID value may be used to further modify the given behavior.

Example Usage (C#)

```
int id = _core.ApiData().GetBehaviorID("automotive", "greeting");
```

Example Usage (LUA)

```
id = silvia.data.getBehaviorID("automotive", "greeting")
```

Parameters

group	The string value representing the behavior's group.
name	The string name, that combined with the group, provides a unique pair identifier for the behavior.

Returns

The integer number value representing the unique ID of the new or existing behavior. The value is -1 if the operation failed.

[silvia.data.getBehaviorIDFromIndex](#)

Gets a behavior ID from the given 0-base index

Description

This function returns an existing ID value if the given behavior exists. The returned integer ID value may be used to further modify the given behavior.

Example Usage (C#)

```
int id = _core.ApiData().GetBehaviorIDFromIndex(index);
```

Example Usage (LUA)

```
id = silvia.data.getBehaviorIDFromIndex(index)
```

Parameters

index

The integer value representing the 0-based index of the behavior in the data.

Returns

The integer number value representing the unique ID of the existing behavior. The value is - 1 if the operation failed.

[silvia.data.getBehaviorCount](#)

Returns the total number of behaviors in the current brain data

Description

This method returns the number of behaviors in the brain. One possible use is to employ the returned integer as a limit for programatically iterating through all of the behaviors.

Example Usage (C#)

```
int count = _core.ApiData().GetBehaviorCount();
```

Example Usage (LUA)

```
count = silvia.data.getBehaviorCount()
```

Parameters

none

Returns

The integer number value representing the total number of behaviors. The value is -1 if the operation failed.

[silvia.data.getBehaviorGroup](#)

Returns the group of the given behavior

Description

This method returns the group value of the identified behavior.

Example Usage (C#)

```
String group = _core.ApiData().GetBehaviorGroup(144);
```

Example Usage (LUA)

```
group = silvia.data.getBehaviorGroup(144)
```

Parameters

id	The integer numeric value or variable representing the unique ID
----	--

of the behavior to be queried.

Returns

The string value representing the group that the behavior belongs to. A nil value is returned if the operation failed.

[silvia.data.getBehaviorSubGroup](#)

Returns the sub-group of the given behavior

Description

This method returns the sub-group value of the identified behavior.

Example Usage (C#)

```
String subGroup = _core.ApiData().GetBehaviorSubGroup(144);
```

Example Usage (LUA)

```
subGroup = silvia.data.getBehaviorSubGroup(144)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior to be queried.
----	---

Returns

The string value representing the sub-group that the behavior belongs to. A nil value is returned if the operation failed, or if the identified behavior does not belong to a subgroup.

[silvia.data.getBehaviorName](#)

Returns the name of the given behavior

Description

This method returns the name value of the identified behavior.

Example Usage (C#)

```
String name = _core.ApiData().GetBehaviorName(144);
```

Example Usage (LUA)

```
name = silvia.data.getBehaviorName(144)
```

Parameters

id	The integer numeric value or variable representing the unique ID
----	--

of the behavior to be queried.

Returns

The string value representing the name of the behavior. A nil value is returned if the operation failed.

[silvia.data.removeBehavior](#)

Deletes the behavior with the given ID

Description

This function deletes an existing behavior with the given ID value, modifying the current SILVIA brain to do so.

Example Usage (C#)

```
bool success = _core.ApiData().RemoveBehavior(144);
```

Example Usage (LUA)

```
success = silvia.data.removeBehavior(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior to be deleted.
----	---

Returns

The boolean success or failure of the operation.

[silvia.data.addAbsorber](#)

Creates a new Absorber with the given user input pattern

Description

This function creates a new Absorber in the identified behavior, modifying the current SILVIA brain to do so. The returned integer Absorber ID value may be used to further modify the newly created Absorber.

Example Usage (C#)

```
int absID = _core.ApiData().AddAbsorber(147, "what is your name");
```

Example Usage (LUA)

```
absID = silvia.data.addAbsorber(147, "what is your name")
```


Parameters

id	The numeric integer id of the behavior to modify by adding an Absorber.
pattern	The string value representing the new Absorber input pattern.

Returns

The integer number value representing the indexed ID for the new Absorber, separate from the behavior ID. The value is -1 if the operation failed.

[silvia.data.setAbsorberText](#)

Modifies a given Absorber with the given AI input pattern

Description

This function replaces one of the input patterns in the identified behavior, modifying the current SILVIA brain to do so.

Example Usage (C#)

```
bool success = _core.ApiData().SetAbsorberText(147, 0, "what is your name, if you don't mind me asking?");
```

Example Usage (LUA)

```
success = silvia.data.SetAbsorberText(147, 0, "what is your name, if you don't mind me asking?")
```

Parameters

id	The numeric integer id of the behavior to modify by changing an Absorber
AbsorberID	The numeric integer id of the Absorber to modify
pattern	The string value representing the updated Absorber output pattern

Returns

The boolean value representing the success or failure of the operation.

`silvia.data.getAbsorberText`

Returns a given Absorber's AI input pattern

Description

This function returns one of the input patterns in the identified behavior.

Example Usage (C#)

```
String pattern = _core.ApiData().GetAbsorberText(147, 0);
```

Example Usage (LUA)

```
pattern = silvia.data.getAbsorberText(147, 0)
```

Parameters

id	The numeric integer id of the behavior from which to read the data
AbsorberID	The numeric integer id of the Absorber to read

Returns

The string value representing the returned input pattern, or a nil value if unsuccessful

[silvia.data.removeAbsorber](#)

Deletes the Absorber represented by the given behavior and Absorber ID pair

Description

This method deletes an existing Absorber within an existing behavior, modifying the current SILVIA brain to do so.

Example Usage (C#)

```
bool success = _core.ApiData().RemoveAbsorber(147, 0);
```

Example Usage (LUA)

```
success = silvia.data.removeAbsorber(147, 0)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Absorber to be deleted.
absid	The integer numeric value or variable representing the indexed ID of the

Absorber, separate from the behavior ID.

Returns

The boolean success or failure of the operation.

`silvia.data.addExuder`

Creates a new Exuder with the given AI output pattern

Description

This function creates a new Exuder in the identified behavior, modifying the current SILVIA brain to do so. The returned integer Exuder ID value may be used to further modify the newly created Exuder.

Example Usage (C#)

```
int exID = _core.ApiData().AddExuder(147, "my name is $_a. it is terrific to meet you.");
```

Example Usage (LUA)

```
exID = silvia.data.addExuder(147, "my name is $_a. it is terrific to meet you.")
```

Parameters

<code>id</code>	The numeric integer id of the behavior to modify by adding an Exuder.
<code>pattern</code>	The string value representing the new Exuder output pattern.

Returns

The integer number value representing the indexed ID for the new Exuder, separate from the behavior ID. The value is -1 if the operation failed.

`silvia.data.setExuderText`

Modifies a given Exuder with the given AI output pattern

Description

This function replaces one of the output patterns in the identified behavior, modifying the current SILVIA brain to do so.

Example Usage (C#)

```
bool success = _core.ApiData().SetExuderText(147, 2, "my name is $_a. it is terrific to meet you.");
```

Example Usage (LUA)

```
success = silvia.data.setExuderText(147, 2, "my name is $_a. it is terrific to meet  
you.")
```

Parameters

id	The numeric integer id of the behavior to modify by changing an Exuder.
ExuderID	The numeric integer id of the Exuder to modify
pattern	The string value representing the updated Exuder output pattern.

Returns

The boolean value representing the success or failure of the operation.

[silvia.data.getExuderText](#)

Returns a given Exuder's AI output pattern

Description

This function returns one of the output patterns in the identified behavior.

Example Usage (C#)

```
String pattern = _core.ApiData().GetExuderText(147, 0);
```

Example Usage (LUA)

```
pattern = silvia.data.getExuderText(147, 0)
```

Parameters

id	The numeric integer id of the behavior from which to read the data
ExuderID	The numeric integer id of the Exuder to read

Returns

The string value representing the returned output pattern, or a nil value if unsuccessful.

`silvia.data.removeExuder`

Deletes the Exuder represented by the given behavior and Exuder ID pair

Description

This method deletes an existing Exuder within an existing behavior, modifying the current SILVIA brain to do so.

Example Usage (C#)

```
bool success = _core.ApiData().RemoveExuder(147, 0);
```

Example Usage (LUA)

```
success = silvia.data.removeExuder(147, 0)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder to be deleted.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.

Returns

The boolean success or failure of the operation.

[silvia.data.setBehaviorScript](#)

Sets the programmatic script content for the behavior.

Description

This function sets the script content for any one of three execution stages in a particular identified behavior, modifying the current SILVIA brain to do so. These stages include “val” (validation), “pre” (pre-behavior), and “post” (post-behavior). Two scripting languages, “LUA” (LUA), and “cs” (C-Sharp), are currently supported, and which one you use will depend on your target runtime platform.

Example Usages (C#)

```
bool success = _core.ApiData().SetBehaviorScript(147, “val”, “cs”, myValidationScript); success =  
_core.ApiData().SetBehaviorScript(147, “pre”, “cs”, myPreScript); success =  
_core.ApiData().SetBehaviorScript(147, “post”, “cs”, myPostScript);
```

Example Usages (LUA)

```
success = silvia.data.setBehaviorScript(147, “val”, “LUA”, “if (myvariable == nil) then return false end”)  
success = silvia.data.setBehaviorScript(147, “pre”, “LUA”, “myvariable = myvariable + 1”) success =  
silvia.data.setBehaviorScript(147, “post”, “LUA”, “if (myvariable
```



```
> 5) then myvariable = nil end")
```

Parameters

id	The numeric integer id of the behavior to modify by setting the script.
stage	The string value representing one of the three scripting stages, “val”, “pre”, or “post”.
language	The string value representing one of the two supported scripting languages, “LUA”, or “cs”.
script	The string value representing the script code to apply to the given behavior.

Returns

The boolean success or failure of the operation.

[silvia.data.setExuderscript](#)

Sets the programmatic script content for the given indexed Exuder within the given behavior.

Description

This function sets the script content for any one of three execution stages in a particular identified Exuder, modifying the current SILVIA brain to do so. These stages include “val” (validation), “pre” (pre-behavior), and “post” (post-behavior). Two scripting languages, “LUA” (LUA), and “cs” (C-Sharp), are currently supported, and which one you use will depend on your target runtime platform.

Example Usages (C#)

```
bool success = _core.ApiData().SetExuderscript(147, “val”, “cs”, myValidationScript);
success = _core.ApiData().SetExuderscript(147, “pre”, “cs”, myPreScript); success =
_core.ApiData().SetExuderscript(147, “post”, “cs”, myPostScript);
```

Example Usages (LUA)

```
success = silvia.data.setExuderscript(147, 1, “val”, “LUA”, “if (myvariable == nil) then return false end”)
success = silvia.data.setExuderscript(147, 1, “pre”, “LUA”, “myvariable = myvariable + 1”) success =
silvia.data.setExuderscript(147, 1, “post”, “LUA”, “if (myvariable > 5) then myvariable = nil end”)
```

Parameters

id	The numeric integer value or variable id of the behavior to modify by
----	---

	setting the script.
exid	The numeric integer value or variable representing the zero-indexed ID of the Exuder, separate from the behavior ID.
stage	The string value representing one of the three scripting stages, “val”, “pre”, or “post”.
language	The string value representing one of the two supported scripting languages, “LUA”, or “cs”.
script	The string value representing the script code to apply to the given Exuder withing the behavior.

Returns

The boolean success or failure of the operation.

[silvia.data.setBehaviorData](#)

Sets the arbitrary, application specific string data content for the behavior.

Description

This function sets the data content for a particular identified behavior, modifying the current SILVIA brain to do so. This arbitrary string data can be fetched programatically, or can also be served up automatically by the SILVIA Server when a particular behavior is invoked.

Example Usage (C#)

```
bool success = _core.ApiData().SetBehaviorData(147,
"images/73/54/userImage_423.jpg");
```

Example Usage (LUA)

```
success = silvia.data.setBehaviorData(147, "images/73/54/userImage_423.jpg")
```

Parameters

id	The numeric integer id of the behavior to modify by setting the arbitrary data.
----	---

data

The string value representing the new data to apply to the given behavior.

Returns

The boolean success or failure of the operation.

`silvia.data.getBehaviorData`

Returns the arbitrary, application specific string data content for the behavior.

Description

This function gets the data content for a particular identified behavior.

Example Usage (C#)

```
String data = _core.ApiData().GetBehaviorData(147);
```

Example Usage (LUA)

```
data = silvia.data.getBehaviorData(147)
```

Parameters

id	The numeric integer id of the behavior containing the arbitrary data.
----	---

Returns

data	The string value representing the data currently attached to the given behavior.
------	--

`silvia.data.setBehaviorSecurityLevel`

Sets the numeric integer security level for the given behavior.

Description

This function sets the security level for a specific behavior, modifying the current SILVIA brain to do so. The security range is 0 - n, where 0 is “open” to anyone, and any value above 0 must be met with a matching or greater user security level for the behavior to be invoked. Note that a higher behavior security level can override a lower Exuder security level.

Example Usage (C#)

```
bool success = _core.ApiData().SetBehaviorSecurityLevel(147, 2);
```

Example Usage (LUA)

```
success = silvia.data.setBehaviorSecurityLevel(147, 2)
```

Parameters

id	The numeric integer id of the behavior to modify by setting the security level
level	The numeric integer value representing the new security level to apply to the given behavior

Returns

The boolean success or failure of the operation.

[silvia.data.getBehaviorSecurityLevel](#)

Returns the integer security level for the given behavior.

Description

This function gets the security level for a particular identified behavior. The range is 0 - n, where 0 is “open” to anyone, and any value above 0 must be met with a matching or greater user security level for the behavior to be invoked. Note that a higher behavior security level can override a lower Exuder security level.

Example Usage (C#)

```
int level = _core.ApiData().GetBehaviorSecurityLevel(147);
```

Example Usage (LUA)

```
level = silvia.data.getBehaviorSecurityLevel(147)
```

Parameters

id	The numeric integer id of the behavior being queried for its security level
----	---

Returns

level	The number value representing the security level of the given behavior.
-------	---

[silvia.data.setAbsorberData](#)

Sets the arbitrary, application specific string data content for the given Absorber.

Description

This function sets the data content for a particular identified Absorber, modifying the current SILVIA brain to do so. This arbitrary string data can be fetched programatically, or can also

be served up automatically by the SILVIA Server when a particular Absorber within a behavior is invoked.

Example Usage (C#)

```
bool success = _core.ApiData().SetAbsorberData(147, 2,
"images/73/54/userImage_423.jpg");
```

Example Usage (LUA)

```
success = silvia.data.setAbsorberData(147, 2, "images/73/54/userImage_423.jpg")
```

Parameters

id	The numeric integer id of the behavior where the Absorber to modify is located
absID	The numeric integer id of the Absorber with the behavior to modify by setting the arbitrary data
data	The string value representing the new data to apply to the given Absorber within the given behavior.

Returns

The boolean success or failure of the operation.

[silvia.data.getAbsorberData](#)

Returns the arbitrary, application specific string data content for the given Absorber.

Description

This function gets the data content for a particular identified Absorber within a specific behavior.

Example Usage (C#)

```
String data = _core.ApiData().GetAbsorberData(147, 2);
```

Example Usage (LUA)

```
data = silvia.data.getAbsorberData(147, 2)
```

Parameters

id	The numeric integer id of the behavior where the Absorber is located.
absID	The numeric integer id of the Absorber containing the arbitrary data

Returns

data	The string value representing the data currently attached to the given Absorber within the given behavior.
------	--

[silvia.data.setExuderData](#)

Sets the arbitrary, application specific string data content for the given Exuder.

Description

This function sets the data content for a particular identified Exuder, modifying the current SILVIA brain to do so. This arbitrary string data can be fetched programatically, or can also be served up automatically by the SILVIA Server when a particular Exuder within a behavior is invoked.

Example Usage (C#)

```
bool success = _core.ApiData().SetExuderData(147, 1,
"images/73/54/userImage_423.jpg");
```

Example Usage (LUA)

```
success = silvia.data.setExuderData(147, 1, "images/73/54/userImage_423.jpg")
```

Parameters

id	The numeric integer id of the behavior where the Exuder to modify is located
exID	The numeric integer id of the Exuder with the behavior to modify by setting the arbitrary data
data	The string value representing the new data to apply to the given Exuder within the given behavior.

Returns

The boolean success or failure of the operation.

[silvia.data.getExuderData](#)

Returns the arbitrary, application specific string data content for the given Exuder.

Description

This function gets the data content for a particular identified Exuder within a specific behavior.

Example Usage (C#)

```
String data = _core.ApiData().GetExuderData(147, 1);
```

Example Usage (LUA)

```
data = silvia.data.getExuderData(147, 1)
```

Parameters

id	The numeric integer id of the behavior where the Exuder is located.
exID	The numeric integer id of the Exuder containing the arbitrary data

Returns

data	The string value representing the data currently attached to the given Exuder within the given behavior.
------	--

[silvia.data.setExudersecurityLevel](#)

Sets the numeric integer security level for the given Exuder.

Description

This function sets the security level for a particular identified Exuder within a specific behavior, modifying the current SILVIA brain to do so. The security range is 0 - n, where 0 is "open" to anyone, and any value above 0 must be met with a matching or greater user security level for the Exuder to be invoked. Note that a higher behavior security level can override a lower Exuder security level.

Example Usage (C#)

```
bool success = _core.ApiData().SetExudersecurityLevel(147, 1, 2);
```

Example Usage (LUA)

```
success = silvia.data.setExudersecurityLevel(147, 1, 2)
```

Parameters

id	The numeric integer id of the behavior where the Exuder to modify is located
exID	The numeric integer id of the Exuder with the behavior to modify by setting the security level
level	The numeric integer value representing the new security level to apply to the given Exuder within the given behavior

Returns

The boolean success or failure of the operation.

[silvia.data.getExudersecurityLevel](#)

Returns the integer security level for the given Exuder

Description

This function gets the security level for a particular identified Exuder within a specific behavior. The range is 0 - n, where 0 is “open” to anyone, and any value above 0 must be met with a matching or greater user security level for the Exuder to be invoked. Note that a higher behavior security level can override a lower Exuder security level.

Example Usage (C#)

```
int level = _core.ApiData().GetExudersecurityLevel(147, 1);
```

Example Usage (LUA)

```
level = silvia.data.getExudersecurityLevel(147, 1)
```

Parameters

id	The numeric integer id of the behavior where the Exuder is located
exID	The numeric integer id of the Exuder being queried for its security level

Returns

level	The number value representing the security level of the given Exuder within the given behavior.
-------	---

silvia.data.setAbsorberExact

Enables or disables the Absorber's requirement for an exact conceptual match on the user input.

Description

This method sets the "exact" state of an existing Absorber within an existing behavior, modifying the current SILVIA brain to do so. Normally, when matching Absorbers to user input, SILVIA accepts reasonably close conceptual matches, within a certain threshold (see `silvia.brain.setAbsorberThreshold`). When the "exact" state is enabled for a particular Absorber, the conceptual input given by the user must COMPLETELY match the given Absorber for that behavior to be considered. This is a useful constraint for mission critical input, where no mistakes on the user's part during input are to be allowed.

Example Usage (C#)

```
bool success = _core.ApiData().SetAbsorberExact(147, 0, true);
```

Example Usage (LUA)

```
success = silvia.data.setAbsorberExact(147, 0, true)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Absorber to be constrained.
absid	The integer numeric value or variable representing the indexed ID of the Absorber, separate from the behavior ID.
exact	The boolean value, true or false, to enable or disable the "exact" constraint for the given Absorber.

Returns

The boolean success or failure of the operation.

silvia.data.getAbsorberExact

Returns a true or false value of the Absorber's requirement for an exact conceptual match

on the user input.

Description

This method gets the “exact” state of an existing Absorber within an existing behavior. Normally, when matching Absorbers to user input, SILVIA accepts reasonably close conceptual matches, within a certain threshold (see `silvia.brain.setAbsorberThreshold`). When the “exact” state is enabled for a particular Absorber, the conceptual input given by the user must COMPLETELY match the given Absorber for that behavior to be considered. This is a useful constraint for mission critical input, where no mistakes on the user’s part during input are to be allowed.

Example Usage (C#)

```
bool exact = _core.ApiData().GetAbsorberExact(147, 0);
```

Example Usage (LUA)

```
exact = silvia.data.getAbsorberExact(147, 0)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior.
absid	The integer numeric value or variable representing the indexed ID of the Absorber, separate from the behavior ID.

Returns

The boolean value, true or false, representing the state of the “exact” constraint for the given Absorber.

[silvia.data.setAbsorberReject](#)

Enables or disables the Absorber’s use as a rejection filter

Description

This method sets the “reject” state of an existing Absorber within an existing behavior, modifying the current SILVIA brain to do so. Normally, when matching Absorbers to user input, SILVIA uses these Absorbers as filters to find the closest conceptual matches to user input, within a certain threshold (see `silvia.brain.setAbsorberThreshold`). However, when the “reject” state is enabled for a particular Absorber, if the conceptual input given by the user most closely matches that Absorber within a behavior, then that entire behavior is rejected from consideration. This is a useful constraint when you want to accept certain input as valid, but reject similar input as invalid, such as the difference between, “i like ice cream”, and “i don’t like ice cream”. By making one of these a rejection filter, you can easily further constrain what is acceptable input for a particular behavior to get invoked.

Example Usage (C#)

```
bool success = _core.ApiData().SetAbsorberReject(147, 1, true);
```

Example Usage (LUA)

```
success = silvia.data.setAbsorberReject(147, 1, true)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Absorber to be constrained.
absid	The integer numeric value or variable representing the indexed ID of the Absorber, separate from the behavior ID.
reject	The boolean value, true or false, to enable or disable the “reject” constraint for the given Absorber.

Returns

The boolean success or failure of the operation.

[silvia.data.getAbsorberReject](#)

Returns the state of the Absorber’s use as a rejection filter

Description

This method returns the “reject” state of an existing Absorber within an existing behavior. Normally, when matching Absorbers to user input, SILVIA uses these Absorbers as filters to find the closest conceptual matches to user input, within a certain threshold (see [silvia.brain.setAbsorberThreshold](#)). However, when the “reject” state is enabled for a particular Absorber, if the conceptual input given by the user most closely matches that Absorber within a behavior, then that entire behavior is rejected from consideration. This is a useful constraint when you want to accept certain input as valid, but reject similar input as invalid, such as the difference between, “i like ice cream”, and “i don’t like ice cream”. By making one of these a rejection filter, you can easily further constrain what is acceptable input for a particular behavior to get invoked.

Example Usage (C#)

```
bool reject = _core.ApiData().GetAbsorberReject(147, 1);
```

Example Usage (LUA)

```
reject = silvia.data.getAbsorberReject(147, 1)
```


Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Absorber.
absid	The integer numeric value or variable representing the indexed ID of the Absorber, separate from the behavior ID.

Returns

The boolean value, true or false, representing the state of the “reject” constraint for the given Absorber.

`silvia.data.setExuderExact`

Enables or disables the AI requirement to express (output) exactly what is contained in the Exuder.

Description

This method sets the “exact” state of an existing Exuder within an existing behavior, modifying the current SILVIA brain to do so. Normally, when generating output, a SILVIA AI can intelligently modify Exuders during the output generation phase, using conceptual substitution, language variations, and other methods to introduce variety. However, when the “exact” state is enabled for a particular Exuder, if that Exuder is invoked during output, it will be used “as-is”, without such variation. This is a useful constraint for mission critical output, where the AI’s wording and phrasing must be consistent every time.

Example Usage (C#)

```
bool success = _core.ApiData().SetExuderExact(147, 0, true);
```

Example Usage (LUA)

```
success = silvia.data.setExuderExact(147, 0, true)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder to be constrained.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.
exact	The boolean value, true or false, to enable or disable the “exact” constraint for the given Exuder.

Returns

The boolean success or failure of the operation.

`silvia.data.getExuderExact`

Returns the boolean state of the AI requirement to express (output) exactly what is contained in the Exuder.

Description

This method returns the “exact” state of an existing Exuder within an existing behavior. Normally, when generating output, a SILVIA AI can intelligently modify Exuders during the output generation phase, using conceptual substitution, language variations, and other methods to introduce variety. However, when the “exact” state is enabled for a particular Exuder, if that Exuder is invoked during output, it will be used “as-is”, without such variation. This is a useful constraint for mission critical output, where the AI’s wording and phrasing must be consistent every time.

Example Usage (C#)

```
bool exact = _core.ApiData().GetExuderExact(147, 0);
```

Example Usage (LUA)

```
exact = silvia.data.getExuderExact(147, 0)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.

Returns

The boolean value, true or false, representing the state of the “exact” constraint for the given Exuder.

`silvia.data.setExuderReuse`

Enables or disables the AI’s ability to draw on a given Exuder in generating dynamic output.

Description

This method sets the “reuse” state of an existing Exuder within an existing behavior, modifying the current SILVIA brain to do so. The SILVIA AI is capable of generating “dynamic” output, from a set of concepts. In practice, this

means that SILVIA can come up with new and interesting ways of expressing those concepts using existing language data.

By setting the “reuse” state to true for a particular Exuder, you are adding that Exuder to SILVIA’s language data, to be drawn from when generating new output.

Example Usage (C#)

```
bool success = _core.ApiData().SetExuderReuse(147, 2, true);
```

Example Usage (LUA)

```
success = silvia.data.setExuderReuse(147, 2, true)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder to be constrained.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.
reuse	The boolean value, true or false, to enable or disable the “reuse” constraint for the given Exuder.

Returns

The boolean success or failure of the operation.

[silvia.data.getExuderReuse](#)

Returns the AI’s ability to draw on a given Exuder in generating dynamic output.

Description

This method returns the “reuse” state of an existing Exuder within an existing behavior. The SILVIA AI is capable of generating “dynamic” output, from a set of concepts. In practice, this means that SILVIA can come up with new and interesting ways of expressing those concepts using existing language data. By setting the “reuse” state to true for a particular Exuder, you are adding that Exuder to SILVIA’s pool of reusable language data, to be drawn from when generating new output.

Example Usage (C#)

```
bool reuse = _core.ApiData().GetExuderReuse(147, 2);
```

Example Usage (LUA)

```
reuse = silvia.data.getExuderReuse(147, 2)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.
reuse	The boolean value, true or false, to enable or disable the “reuse” constraint for the given Exuder.

Returns

The boolean value, true or false, representing the state of the “reuse” constraint for the given Exuder.

[silvia.data.setExuderDynamic](#)

Enables or disables the AI’s ability to use the concepts in an Exuder to generate new, dynamic output.

Description

This method sets the “dynamic” state of an existing Exuder within an existing behavior, modifying the current SILVIA brain to do so. The SILVIA AI is capable of generating such “dynamic” output from the set of concepts within the Exuder. In practice, this means that SILVIA can come up with new and interesting ways of expressing those concepts using existing language data. The new output is generated using any Exuders that have the “reuse” flag set as source data for inferred language rules and conceptual linking. For instance, a “dynamic” Exuder might contain the following: “enjoy artificial intelligence robots”. If invoked, SILVIA will use her existing “reuse” Exuder data and concept bindings to dynamically create something interesting to say about her enjoyment of artificial intelligence and robots, with the likelihood that each output will be different from the last.

Example Usage (C#)

```
bool success = _core.ApiData().SetExuderDynamic(147, 4, true);
```

Example Usage (LUA)

```
success = silvia.data.setExuderDynamic(147, 4, true)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder to be constrained.
----	---

exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.
dynamic	The boolean value, true or false, to enable or disable the “dynamic” constraint for the given Exuder.

Returns

The boolean success or failure of the operation.

[silvia.data.getExuderDynamic](#)

Returns the state of the AI’s ability to use the concepts in a particular Exuder to generate new, dynamic output.

Description

This method returns the “dynamic” state of an existing Exuder within an existing behavior. The SILVIA AI is capable of generating such “dynamic” output from the set of concepts within the Exuder. In practice, this means that SILVIA can come up with new and interesting ways of expressing those concepts using existing language data. The new output is generated using any Exuders that have the “reuse” flag set as source data for inferred language rules and conceptual linking. For instance, a “dynamic” Exuder might contain the following: “enjoy artificial intelligence robots”. If invoked, SILVIA will use her existing “reuse” Exuder data and concept bindings to dynamically create something interesting to say about her enjoyment of artificial intelligence and robots, with the likelihood that each output will be different from the last.

Example Usage (C#)

```
bool dynamic = _core.ApiData().GetExuderDynamic(147, 4);
```

Example Usage (LUA)

```
dynamic = silvia.data.getExuderDynamic(147, 4)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.
dynamic	The boolean value, true or false, to enable or disable the “dynamic”

constraint for the given Exuder.

Returns

The boolean value, true or false, representing the state of the “dynamic” constraint for the given Exuder.

[silvia.data.setExuderContext](#)

Sets the given Exuder’s conceptual context field

Description

This method sets the context field of an existing Exuder within an existing behavior, modifying the current SILVIA brain to do so. Each Exuder may contain an optional context, where certain concepts must be within recent memory as part of the discussion. If the context constraint is NOT met during the course of conversation, then that Exuder is discarded from consideration in the output phase.

Example Usage (C#)

```
bool success = _core.ApiData().SetExuderContext(147, 3, “star wars”);
```

Example Usage (LUA)

```
success = silvia.data.setExuderContext(147, 3, “star wars”)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder to be constrained.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.
context	The string value containing the conversational context constraints for the given Exuder.

Returns

The boolean success or failure of the operation.

[silvia.data.getExuderContext](#)

Returns the given Exuder’s conceptual context field

Description

This method returns the context field of an existing Exuder within an existing behavior. Each Exuder may contain an optional context, where certain concepts must be within recent memory as part of the discussion. If the context constraint is NOT met during the course of conversation, then that Exuder is discarded from consideration in the output phase.

Example Usage (C#)

```
String context = _core.ApiData().GetExuderContext(147, 3);
```

Example Usage (LUA)

```
context = silvia.data.getExuderContext(147, 3)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior containing the Exuder.
exid	The integer numeric value or variable representing the indexed ID of the Exuder, separate from the behavior ID.

Returns

The string value containing the current conversational context constraints for the given Exuder.

[silvia.data.getBehaviorCreatedYear](#)

Gets the given behavior's creation date "year" value

Description

This method returns the year that the given behavior, indicated by the id, was created.

Example Usage (C#)

```
int year = _core.ApiData().GetBehaviorCreatedYear(147);
```

Example Usage (LUA)

```
year = silvia.data.getBehaviorCreatedYear(147)
```

Parameters

id

The integer numeric value or variable representing the unique ID of the

behavior

Returns

The integer value (4 decimal) representing the behavior's year of creation.

[silvia.data.getBehaviorCreatedMonth](#)

Gets the given behavior's creation date "month" value

Description

This method returns the month that the given behavior, indicated by the id, was created. This value will be between 1 and 12, representing the 12 months between January and December, inclusive.

Example Usage (C#)

```
int month = _core.ApiData().GetBehaviorCreatedMonth(147);
```

Example Usage (LUA)

```
month = silvia.data.getBehaviorCreatedMonth(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's month of creation.

[silvia.data.getBehaviorCreatedDay](#)

Gets the given behavior's creation date "day" value

Description

This method returns the day of the month that the given behavior, indicated by the id, was created. This value will be between 1 and 31.

Example Usage (C#)

```
int day = _core.ApiData().GetBehaviorCreatedDay(147);
```

Example Usage (LUA)

```
day = silvia.data.getBehaviorCreatedDay(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's day of creation within the month.

[silvia.data.getBehaviorCreatedHour](#)

Gets the given behavior's creation date "hour" value

Description

This method returns the hour that the given behavior, indicated by the id, was created. This value will be between 0 and 23, representing the 24 hours in any given day. If the hour value is less than 12, then the behavior was created in the AM, otherwise it was created in the PM.

Example Usage (C#)

```
int hour = _core.ApiData().GetBehaviorCreatedHour(147);
```

Example Usage (LUA)

```
hour = silvia.data.getBehaviorCreatedHour(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's hour of creation.

[silvia.data.getBehaviorCreatedMinute](#)

Gets the given behavior's creation date "minute" value

Description

This method returns the minute that the given behavior, indicated by the id, was created. This value will be between 0 and 59, representing the 60 minutes in any given hour. When combined with the "hour" value, this will give the complete time of day value for the behavior's creation.

Example Usage (C#)

```
int minute = _core.ApiData().GetBehaviorCreatedMinute(147);
```

Example Usage (LUA)

```
minute = silvia.data.getBehaviorCreatedMinute(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's minute of creation.

[silvia.data.getBehaviorLastModifiedYear](#)

Gets the given behavior's last modified date "year" value

Description

This method returns the year that the given behavior, indicated by the id, was last modified.

Example Usage (C#)

```
int year = _core.ApiData().GetBehaviorLastModifiedYear(147);
```

Example Usage (LUA)

```
year = silvia.data.getBehaviorLastModifiedYear(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (4 decimal) representing the behavior's year of most recent modification.

[silvia.data.getBehaviorLastModifiedMonth](#)

Gets the given behavior's last modified date "month" value

Description

This method returns the month that the given behavior, indicated by the id, was last modified. This value will be between 1 and 12, representing the 12 months between January and December, inclusive.

Example Usage (C#)

```
int month = _core.ApiData().GetBehaviorLastModifiedMonth(147);
```

Example Usage (LUA)

```
month = silvia.data.getBehaviorLastModifiedMonth(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's month of last modification,

[silvia.data.getBehaviorLastModifiedDay](#)

Gets the given behavior's last modified date "day" value

Description

This method returns the day of the month that the given behavior, indicated by the id, was last modified. This value will be between 1 and 31.

Example Usage (C#)

```
int day = _core.ApiData().GetBehaviorLastModifiedDay(147);
```

Example Usage (LUA)

```
day = silvia.data.getBehaviorLastModifiedDay(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's day of last modification within the month.

[silvia.data.getBehaviorLastModifiedHour](#)

Gets the given behavior's last modified date "hour" value

Description

This method returns the hour that the given behavior, indicated by the id, was last modified. This value will be between 0 and 23, representing the 24 hours in any given day. If the hour value is less than 12, then the behavior was last modified in the AM, otherwise it was last modified in the PM.

Example Usage (C#)

```
int hour = _core.ApiData().GetBehaviorLastModifiedHour(147);
```

Example Usage (LUA)

```
hour = silvia.data.getBehaviorLastModifiedHour(147)
```

Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's hour of last modification.

[silvia.data.getBehaviorLastModifiedMinute](#)

Gets the given behavior's last modified date "minute" value

Description

This method returns the minute that the given behavior, indicated by the id, was last modified. This value will be between 0 and 59, representing the 60 minutes in any given hour. When combined with the "hour" value, this will give the complete time of day value for the behavior's most recent modification.

Example Usage (C#)

```
int minute = _core.ApiData().GetBehaviorLastModifiedMinute(147);
```

Example Usage (LUA)

```
minute = silvia.data.getBehaviorLastModifiedMinute(147)
```


Parameters

id	The integer numeric value or variable representing the unique ID of the behavior
----	--

Returns

The integer value (1 or 2 decimal) representing the behavior's minute of last modification.

[silvia.data.addResponse](#)

Adds a paired input/output response to the behavior data

Description

This function adds knowledge to the current SILVIA brain in the form of paired input/response templates. Note that if a matching learned Absorber already exists, the response will be added to the existing behavior instead. A variable name starting with "\$" may be used for either input or output, in which case the value contained in the variable will be used instead.

Example Usage

```
success = silvia.data.addResponse("how are you today", "I am doing quite well.")
```

Parameters

Absorber	The string or variable name specifying the input template.
Exuder	The string or variable name specifying the output template.

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.setContext](#)
- [silvia.data.setReuse](#)

[silvia.data.setContext](#)

Sets the context for a previously learned response

Description

This function adds knowledge to the current SILVIA brain by setting the conversational context for the most previously learned Exuder. A variable name starting with “\$” may be used, in which case the value contained in the variable will be used instead.

Example Usage

```
success = silvia.data.setContext("movies robots gort")
```

Parameters

context	The string or variable name specifying the context for the Exuder.
---------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.addResponse](#)
- [silvia.data.setReuse](#)

[silvia.data.setReuse](#)

Sets the previously learned response as re-useable or not

Description

This function adds knowledge to the current SILVIA brain by setting the preveiously learned Exuder as re-useable or not. If re-useable, SILVIA is free to draw on the Exuder’s knowledge and syntax data from other behaviors for dynamic output construction.

Example Usage

```
success = silvia.data.setReuse(true)
```

Parameters

reuse	The boolean value enabling or disabling the re-use of the Exuder.
-------	---

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.addResponse](#)
- [silvia.data.setContext](#)

silvia.data.setBehaviorName

Sets the name for the most recently learned behavior

Description

This function adds knowledge to the current SILVIA brain by setting the name of the behavior most recently modified or learned.

Example Usage

```
success = silvia.data.setBehaviorName("my_behavior")
```

Parameters

name	The string representing the new name for the behavior. A nil will clear the name value.
------	---

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.setBehaviorGroup](#)
- [silvia.data.setBehaviorSubGroup](#)

silvia.data.setBehaviorGroup

Sets the group for the most recently learned behavior

Description

This function adds knowledge to the current SILVIA brain by setting the group of the behavior most recently modified or learned.

Example Usage

```
success = silvia.data.setBehaviorGroup("movies")
```

Parameters

group

The string representing the new g62roup for the behavior. A nil will be

ignored.

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.setBehaviorName](#)
- [silvia.data.setBehaviorSubGroup](#)

silvia.data.setBehaviorSubGroup

Sets the sub-group for the most recently learned behavior

Description

This function adds knowledge to the current SILVIA brain by setting the sub-group of the behavior most recently modified or learned. This sub-group has no functional value to the SILVIA algorithms and is only intended as an organizational tool.

Example Usage

```
success = silvia.data.setBehaviorSubGroup("greeting")
```

Parameters

subgroup	The string representing the new sub-category for the behavior. A nil will be ignored.
----------	---

Returns

The boolean success or failure of the operation.

See Also

- [silvia.data.setBehaviorName](#)
- [silvia.data.setBehaviorGroup](#)

silvia.feedback

This class, implemented as part of the SILVIA C#/LUA API, provides access to SILVIA's cognitive feedback functions.

This includes parametric control, data/file access, and invoking SILVIA's feedback functions dynamically.

Summary

silvia.feedback

This class, implemented as part of the SILVIA C#/LUA API, provides access to SILVIA's cognitive feedback functions.

FUNCTIONS

<u>silvia.feedback.setActive</u>	Enables or disables SILVIA's cognitive feedback output
<u>silvia.feedback.isActive</u>	Returns the boolean enabled/disabled status of the SILVIA cognitive feedback system
<u>silvia.feedback.clarify</u>	Forces the SILVIA cognitive feedback to generate a set number of outputs
<u>silvia.feedback.addFeedback</u>	Adds conceptual data to the feedback memory.
<u>silvia.feedback.suggestFeedback</u>	Adds conceptual data to the feedback memory as a suggestion.
<u>silvia.feedback.setInterval</u>	Sets the range in seconds of time between cognitive feedback generated outputs
<u>silvia.feedback.setThreshold</u>	Sets the conceptual trigger values for feedback utterance
<u>silvia.feedback.setProbability</u>	Sets the overall that a feedback output will be uttered
<u>silvia.feedback.pause</u>	Pauses the output of SILVIA's cognitive feedback
<u>silvia.feedback.resume</u>	Resumes the output of SILVIA's cognitive feedback
<u>silvia.feedback.search</u>	Initiates a time-based search of conversational (feedback) data for a particular user
<u>silvia.feedback.searchRemoveFromStack</u>	Removes the current utterance in the search results from the general feedback memory
<u>silvia.feedback.searchGetString</u>	Returns the current utterance in the search results
<u>silvia.feedback.searchGetTime</u>	Returns the time-stamp of the current utterance in the search results
<u>silvia.feedback.searchNext</u>	Increments to the next search result in the array Returns the time-stamp of the current utterance in the search results
<u>silvia.feedback.write</u>	Exports conversational (feedback) data of a particular user
<u>silvia.feedback.read</u>	Imports conversational (feedback) data from a file

Functions

silvia.feedback.setActive

Enables or disables SILVIA's cognitive feedback output

Description

A desirable operating mode is one where SILVIA can dynamically generate contextually relevant output, independent of direct user input. This function sets the active/inactive state of this mode. Note that as SILVIA interacts with users and uses this cognitive feedback mode, a body of feedback data, or “conversational memory” is built up over the course of these interactions. The boolean “clear” flag allows this feedback data to be reset.

Example Usage (C#)

```
_core.ApiFeedback().SetActive(true, false);
```

Example Usage (LUA)

```
silvia.feedback.setActive(true, false)
```

Parameters

enable	The boolean value that turns feedback output on or off.
clear	The boolean value that if true, clears the current feedback memory.

Returns

None.

See Also

- [silvia.feedback.isActive](#)

[silvia.feedback.isActive](#)

Returns the boolean enabled/disabled status of the SILVIA cognitive feedback system

Description

A desirable operating mode is one where SILVIA can dynamically generate contextually relevant output, independent of direct user input. This function returns the boolean status of this mode.

Example Usage (C#)

```
bool isActive = _core.ApiFeedback().IsActive();
```

Example Usage (LUA)

```
isActive = silvia.feedback.isActive()
```

Parameters

none

Returns

The boolean value of the cognitive feedback's enabled/disabled status.

See Also

- [silvia.feedback.setActive](#)

silvia.feedback.clarify

Forces the SILVIA cognitive feedback to generate a set number of outputs

Description

This function forces the invocation of the cognitive feedback system, but only for a set number of times. This is typically used to force SILVIA to follow up on a particular line of thought. For instance, a user might ask SILVIA “what do you mean?” as a generic request for more information. The script associated with the resultant Exuder could contain a call to this “clarify” function, with a count of “1”, to force a single follow-up to SILVIA’s last utterance. The default “response” time is a range between 1.5 and 2.5 seconds. To change this range, you can immediately follow up a call to this function with a call to “silvia.feedback.setInterval”. Note that for this function, and for feedback in general, it is best to have as many “reusable” Exuders in the brain data as possible.

Example Usage (C#)

```
_core.ApiFeedback().Clarify(1);
```

Example Usage (LUA)

```
silvia.feedback.clarify(1)
```

Parameters

count	The number representing the count of additional “feedback” outputs to be generated
-------	--

Returns

None.

See Also

- [silvia.feedback.setInterval](#)

silvia.feedback.addFeedback

Adds conceptual data to the feedback memory.

Description

This function converts the input string parameter to conceptual data and places it in the immediate feedback

memory for consideration. During the course of normal interactions with SILVIA, this memory is automatically manipulated and updated internally by the

SILVIA core. However, this exposed LUA function allows a trainer to more explicitly force the conversational direction of the SILVIA feedback, based on behaviors specific to the training and the application. This function might typically be followed by a call to “`silvia.feedback.clarify`”.

Example Usage (C#)

```
bool success = _core.ApiFeedback().AddFeedback("acting Broadway");  
_core.ApiFeedback().Clarify(1);
```

Example Usage (LUA)

```
success = silvia.feedback.addFeedback("acting Broadway") silvia.feedback.clarify(1)
```

Parameters

<code>feedback</code>	The string containing the concepts to be placed in immediate cognitive “memory”.
-----------------------	--

Returns

The boolean success or failure of the operation.

See Also

- [`silvia.feedback.suggestFeedback`](#)
- [`silvia.feedback.clarify`](#)

`silvia.feedback.suggestFeedback`

Adds conceptual data to the feedback memory as a suggestion.

Description

This function converts the input string parameter to conceptual data and places it in the immediate feedback memory for consideration. However, unlike the “`addFeedback`” function, the conceptual data is only used as suggestive guidance for the course of the feedback, not as a more direct forcing of the conversation. But like “`addFeedback`”, this exposed LUA function does give a trainer more explicit control over the conversational direction of SILVIA’s feedback. Such control can be based on behaviors specific to the functions of the AI brain’s training, and of the application. This function might typically be followed by a call to “`silvia.feedback.clarify`”.

Example Usage (C#)

```
bool success = _core.ApiFeedback().SuggestFeedback("robots");  
_core.ApiFeedback().Clarify(1);
```

Example Usage (LUA)

```
success = silvia.feedback.suggestFeedback("robots") silvia.feedback.clarify(1)
```

Parameters

feedback	The string containing the concepts to be placed in immediate cognitive "memory".
----------	--

Returns

The boolean success or failure of the operation.

See Also

- [silvia.feedback.addFeedback](#)
- [silvia.feedback.clarify](#)

[silvia.feedback.setInterval](#)

Sets the range in seconds of time between cognitive feedback generated outputs

Description

This function controls SILVIA's overall "chattiness", when feedback is enabled. The range is specified as a minimum and maximum. When enabled, each feedback utterance is generated n seconds after the previous feedback utterance, where n is a random value between the minimum and maximum. Note that this function applies to both the "setActive" and "clarify" forms of feedback enablement. For obvious reasons, "minimum" should be <= "maximum".

Example Usage (C#)

```
_core.ApiFeedback().SetInterval(5.0f, 10.0f);
```

Example Usage (LUA)

```
silvia.feedback.setInterval(5.0, 10.0)
```

Parameters

minimum	a number value specifying the lower range of possible feedback intervals
maximum	a number value specifying the upper range of possible feedback intervals

Returns

The boolean success or failure of the operation.

See Also

- [silvia.feedback.setActive](#)
- [silvia.feedback.clarify](#)
- [silvia.feedback.setThreshold](#)
- [silvia.feedback.setProbability](#)

silvia.feedback.setThreshold

Sets the conceptual trigger values for feedback utterance

Description

When feedback is enabled, this function controls SILVIA's conceptual threshold. In other words, the values serve as a limiter on the expression of conceptually unimportant or uninteresting thoughts. The first value is a weight, or conceptual "importance". The second value is the minimum number of concepts needed to meet or exceed the given weight.

Therefore, if a particular internal thought, generated by SILVIA, has \geq the given number of concepts that are \geq the given weight, then that thought is considered "important" enough to be considered for outward expression.

Example Usage (C#)

```
_core.ApiFeedback().SetThreshold(0.35f, 2);
```

Example Usage (LUA)

```
silvia.feedback.setThreshold(0.35, 2)
```

Parameters

threshold	a number value specifying the minimum weight required for a concept's consideration
count	a number value specifying the minimum number of concepts that must meet the weight requirement

Returns

The boolean success or failure of the operation.

See Also

- [silvia.feedback.setActive](#)

- [silvia.feedback.clarify](#)
- [silvia.feedback.setInterval](#)
- [silvia.feedback.setProbability](#)

silvia.feedback.setProbability

Sets the overall that a feedback output will be uttered

Description

When feedback is enabled, this function serves as a final gate for limiting SILVIA's feedback output. The parameter, a coefficient between 0.0 and 1.0, serves a means to control how probable it is at any given time for SILVIA to generate feedback output. For instance, one could set the feedback interval to between 2.0 and 3.0 seconds, and set 0.5 as the probability. The result would be an attempt at a feedback-generated output every two or three seconds, but an actual output generated only half of the time.

Example Usage (C#)

```
_core.ApiFeedback().SetProbability(0.75f);
```

Example Usage (LUA)

```
silvia.feedback.setProbability(0.75)
```

Parameters

probability	a coefficient specifying the overall chance that a given feedback result will be uttered.
-------------	---

Returns

The boolean success or failure of the operation.

See Also

- [silvia.feedback.setActive](#)
- [silvia.feedback.clarify](#)
- [silvia.feedback.setInterval](#)
- [silvia.feedback.setThreshold](#)

silvia.feedback.pause

Pauses the output of SILVIA's cognitive feedback

Description

When feedback is enabled, this function temporarily pauses the generation of SILVIA's

feedback output. This is different than calling “setActive” with a false value because in this

case, a subsequent call to “`silvia.feedback.resume`” will allow SILVIA to continue with the feedback process where it was last left off.

Example Usage (C#)

```
_core.ApiFeedback().Pause();
```

Example Usage (LUA)

```
silvia.feedback.pause()
```

Parameters

none

Returns

None.

See Also

- [`silvia.feedback.resume`](#)
- [`silvia.feedback.setActive`](#)
- [`silvia.feedback.clarify`](#)

`silvia.feedback.resume`

Resumes the output of SILVIA’s cognitive feedback

Description

When feedback is enabled, and feedback has been temporarily paused, this function re- engages SILVIA’s feedback output from the point of the last call to “`silvia.feedback.pause`”. This is different than calling “`setActive`” with a true value because in this case, it is resuming the existing feedback state.

Example Usage (C#)

```
_core.ApiFeedback().Resume();
```

Example Usage (LUA)

```
silvia.feedback.resume()
```

Parameters

none

Returns

None.

See Also

- [silvia.feedback.pause](#)
- [silvia.feedback.setActive](#)
- [silvia.feedback.clarify](#)

silvia.feedback.search

Initiates a time-based search of conversational (feedback) data for a particular user

Description

If there is some conversational data for the named user, it is searched for concepts, within the given time range, and a structure is set up containing the data for sequential reference. A nil values for concepts will cause the method to return all of the given user’s utterances within the given time range. A nil value for startTime causes the search to have no beginning time boundaray, and a nil value for endTime causes the search to have an ending boundary of the current data and time.

Note that the “strip” parameter allows you to remove selected concepts from consideration in the search. For instance, you can easily discard “a”, “an”, “the”, or other unimportant concepts from being required by the internal search.

Example Usages (C#)

```
bool success = _core.ApiFeedback().Search(“robert”, “rock music”, “rolling stones”, true, “3/8/2010 12:15:12”, “3/10/2010 12:15:12”);
bool success = _core.ApiFeedback().Search(“lisa”, “construction”, null, true, nil, nil);
```

Example Usages (LUA)

```
success = silvia.feedback.search(“robert”, “rock music”, “rolling stones”, true, “3/8/2010 12:15:12”, “3/10/2010 12:15:12”)
success = silvia.feedback.search(“lisa”, “construction”, null, true, nil, nil)
```

Parameters

username	a string value naming the user whose conversational data will be searched
concepts	a string value containing one or more concepts that are expected in the data
strip	a string value containing one or more concepts that should not be considered
exclude	a string value containing one or more concepts that will cause a particular utterance to be excluded from the search

startTime	a string containing a formatted date/time
endTime	a string containing a formatted date/time
requireAll	a boolean value indicating whether all concepts are required for an utterance is included
related	a boolean value indicating whether closely related concepts will be accepted as valid

Returns

A boolean success or failure of the operation.

See Also

- [silvia.feedback.searchRemoveFromStack](#)
- [silvia.feedback.searchGetString](#)
- [silvia.feedback.searchGetTime](#)
- [silvia.feedback.searchNext](#)

silvia.feedback.searchRemoveFromStack

Removes the current utterance in the search results from the general feedback memory

Description

After a call to `silvia.feedback.search`, the results are placed in an array structure that may be referenced iteratively. This method removes the current indexed utterance from the general memory pool that was returned as result of that previous search.

Example Usage (C#)

```
bool success = _core.ApiFeedback().SearchRemoveFromStack();
```

Example Usage (LUA)

```
success = silvia.feedback.searchRemoveFromStack()
```

Parameters

none

Returns

A boolean success or failure of the operation.

See Also

- [silvia.feedback.search](#)
- [silvia.feedback.searchGetString](#)
- [silvia.feedback.searchGetTime](#)
- [silvia.feedback.searchNext](#)

[silvia.feedback.searchGetString](#)

Returns the current utterance in the search results

Description

After a call to `silvia.feedback.search`, the results are placed in an array structure that may be referenced iteratively. This method returns the current indexed utterance that was a result of that previous search.

Example Usage (C#)

```
String result = _core.ApiFeedback().SearchGetString(true);
```

Example Usage (LUA)

```
result = silvia.feedback.searchGetString(true)
```

Parameters

<code>punctuated</code>	a boolean value indicating whether the result should include punctuation
-------------------------	--

Returns

A string value containing the utterance.

See Also

- [silvia.feedback.search](#)
- [silvia.feedback.searchRemoveFromStack](#)
- [silvia.feedback.searchGetTime](#)
- [silvia.feedback.searchNext](#)

[silvia.feedback.searchGetTime](#)

Returns the time-stamp of the current utterance in the search results

Description

After a call to `silvia.feedback.search`, the results are placed in an array structure that may be referenced iteratively. This method returns the date/time of the current indexed utterance that was a result of that previous search.

Example Usage (C#)

```
String result = _core.ApiFeedback().SearchGetTime();
```

Example Usage (LUA)

```
result = silvia.feedback.searchGetTime()
```

Parameters

none

Returns

A string value containing the formatted date/time of the utterance.

See Also

- [silvia.feedback.search](#)
- [silvia.feedback.searchRemoveFromStack](#)
- [silvia.feedback.searchGetString](#)
- [silvia.feedback.searchNext](#)

[silvia.feedback.searchNext](#)

Increments to the next search result in the array Returns the time-stamp of the current utterance in the search results

Description

After a call to `silvia.feedback.search`, the results are placed in an array structure that may be referenced iteratively. This method moves the index along to the next result in the array. When called in a loop along with `silvia.feedback.searchGetString`, `silvia.feedback.searchGetTime`, or `silvia.feedback.searchRemoveFromStack`, the developer can get or remove the searched results, one after another, and results may be to be placed in a table or processed as they are received. A false value is returned when there are no more search results in the array.

Example Usage (C#)

```
bool success = _core.ApiFeedback().SearchNext();
```

Example Usage (LUA)

```
success = silvia.feedback.searchNext()
```

Parameters

none

Returns

A boolean value containing the success or failure of the operation.

See Also

- [silvia.feedback.search](#)
- [silvia.feedback.searchRemoveFromStack](#)
- [silvia.feedback.searchGetString](#)
- [silvia.feedback.searchGetTime](#)

silvia.feedback.write

Exports conversational (feedback) data of a particular user

Description

If there is some conversational data for the named user, it is saved to the named file. This can then be loaded and referenced for future sessions.

Example Usage (C#)

```
bool success = _core.ApiFeedback().Write("robert", "feedback/robert.fbk");
```

Example Usage (LUA)

```
success = silvia.feedback.write("robert", "feedback/robert.fbk")
```

Parameters

username	a string value naming the user whose conversational data will be saved
filename	a string value containing the filepath/filename in which to save the data

Returns

A boolean success or failure of the operation.

See Also

- [silvia.feedback.read](#)

silvia.feedback.read

Imports conversational (feedback) data from a file

Description

If the named file of conversational data exists, it is read and added to the conversational feedback stack, with it's proper timestamp and username. This allows for the saving and loading of sessions for a particular user. Particularly useful for persistent memory between sessions.

Example Usage (C#)

```
bool success = _core.ApiFeedback().Read("feedback/robert.fbk");
```

Example Usage (LUA)

```
success = silvia.feedback.read("feedback/robert.fbk")
```

Parameters

filename	a string value containing the filepath/filename from which to load the data
----------	---

Returns

A boolean success or failure of the operation.

See Also

- [silvia.feedback.write](#)

silvia.mem

This class, implemented as part of the SILVIA C#/LUA API, provides access to memory functions that can dynamically remember, forget, or delete blocks of SILVIA knowledge. This also includes file manipulation, since loading, merging, and saving files are operations that can change SILVIA memory as well.

Summary

<u>silvia.mem</u>	This class, implemented as part of the SILVIA C#/LUA API, provides access to memory functions that can dynamically remember, forget, or delete blocks of SILVIA knowledge.
--------------------------	--

FUNCTIONS

silvia.mem.load	Loads or merges one or more SILVIA brain files into memory
---------------------------------	--

silvia.mem.save	Saves a SILVIA brain (concepts, bindings, behaviors) to a file.
---------------------------------	---

[silvia.mem.mergeText](#)

Merges plain-text expert data into the SILVIA brain

<u>silvia.mem.mergeAIML</u>	Merges AIML (AI Markup Language) data into the SILVIA brain
<u>silvia.mem.getAllGroups</u>	Returns a string containing all of the groups in memory, separated by the " " character.
<u>silvia.mem.getActiveGroups</u>	Returns a string containing only the groups in memory that are currently active.
<u>silvia.mem.groupEnable</u>	Enables one or more groups in memory.
<u>silvia.mem.groupEnableOnly</u>	Exclusively enables one or more groups in memory.
<u>silvia.mem.groupDisable</u>	Disables one or more groups in memory.
<u>silvia.mem.groupsEnabled</u>	Tests the enabled/disabled state of a group in memory.
<u>silvia.mem.groupDelete</u>	Deletes one or more groups from memory.
<u>silvia.mem.groupDeleteExcept</u>	Deletes any groups from memory not specified.

Functions

silvia.mem.load

Loads or merges one or more SILVIA brain files into memory

Description

SILVIA is able to dynamically load and merge external brain files. Merging means that data loaded is additive, so one could dynamically merge-load a brain file that "knows" all about cars in response to a query about the subject. Merging also allows the specification of more than one file, using the "|" separator. Note that if there is a behavior with the name "boot" in a brain file, and if this function's boot flag is set to true, that behavior's Exuder(s) and associated events will be invoked immediately after the file has been loaded. This allows trainers to create "startup" behaviors for their trained AI brains.

Example Usages (C#)

```
bool success = _core.ApiMem().Load("latestBrain.slv", false, true); success =
_core.ApiMem().Load("cars.slv|pets.slv|books.slv", true, false);
```

Example Usages (LUA)

```
success = silvia.mem.load("latestBrain.slv", false, true) success =
silvia.mem.load("cars.slv|pets.slv|books.slv", true, false)
```

Parameters

filename	A string containing the name(s) of the SILVIA brain file(s) to be loaded.
merge	A boolean flag to determine if the file is loaded to a fresh brain, or merged with existing data.