SILVIA Server
Overview and Installation

Contents

SILVIA: Introduction and Components

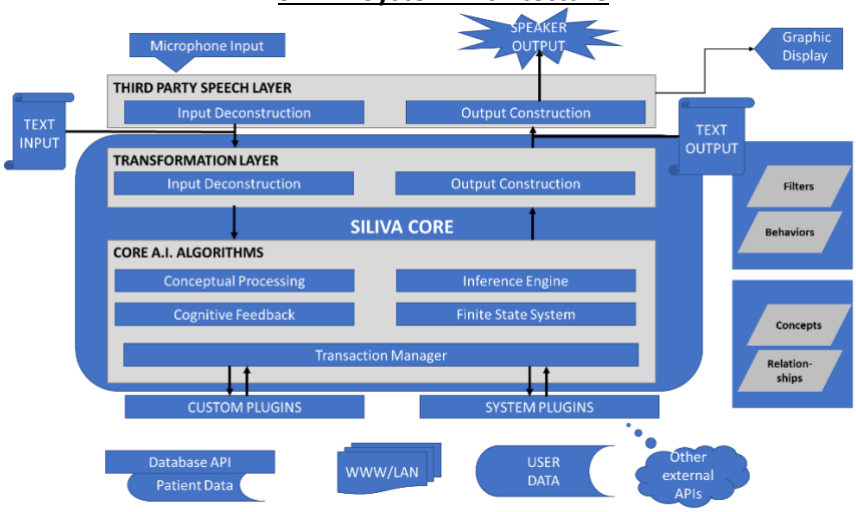
The SILVIA (**S**ymbolically **I**solated **L**inguistically **V**ariable **I**ntelligence **A**lgorithms) platform enables applications to interpret and respond to an end user's speech or text through AI technology.

Components of SILVIA include:

- **SILVIA Core:** a high-performance artificial intelligence runtime engine, configurable for client, server, or mobile and embedded devices;
- **SILVIA Voice:** a modular add-on component for integrating ASR and TTS voice components with any SILVIA application;
- **SILVIA Studio:** a graphical system for an application's SILVIA content and behavior development, including integrated scripting;
- **SILVIA API:** the programmer's interface for creating new SILVIA applications and integrating SILVIA with other applications and data; and,
- **SILVIA Server**: a configurable system for automated management of a number of SILVIA Cores on one or more networked servers.

This document will concentrate on an overview of the SILVIA Server and its installation.

SILVIA System Architecture



Server Overview

Attributes

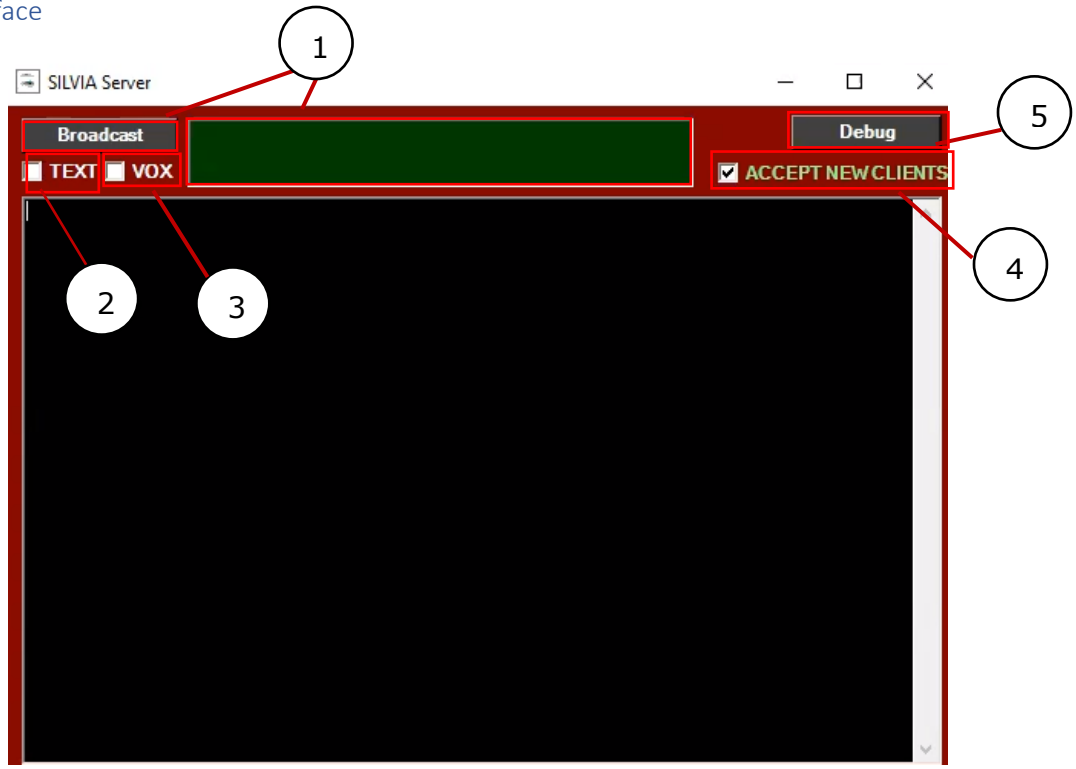
The SILVIA Server is the brains of the SILVIA system. It is responsible for deploying the SILVIA-enabled application to end-users on the web or within an organization. Attributes include:

- **Flexibility**: The SILVIA Server is deployable on-premise or in the cloud with AWS, Azure and more. In addition, the SILVIA Server can be sized to host multiple applications and user session data on a single server or across larger cloud deployments.
- **Compatibility**: The SILVIA Server can automatically manage and optimize third-party Text-To-Speech systems output for efficient voice generation and deployment.
- **Accessibility**: SILVIA's APIs and connectivity are exposed through TCP/IP and RESTful interfaces so applications can easily access the SILVIA Server from a browser or a dedicated client

application and more.

- **High performance**: The SILVIA Server can manage internal processing of over 400 transactions per second on a single instance. In addition, the SILVIA Server uses only 1MB of RAM per active connected user, even with unique per-user experiences.

Server Interface



1. **Broadcast**: Ability to send a "broadcast" message to all users of the client (For example, a SILVIA systems upgrade message to all users of a SILVIA enabled chatbot). To send a broadcast message, type in the green section and select 'Broadcast'. The message will be sent immediately to users.
2. **Text**: Enables text interactions.
3. **VOX**: Enables voice interactions.
4. **Accepts New Clients**: Allows new clients to come on to the installed SILVIA platform.
5. **Debug**: Allows detailed messages on SILVIA server Interactions.

System Requirements

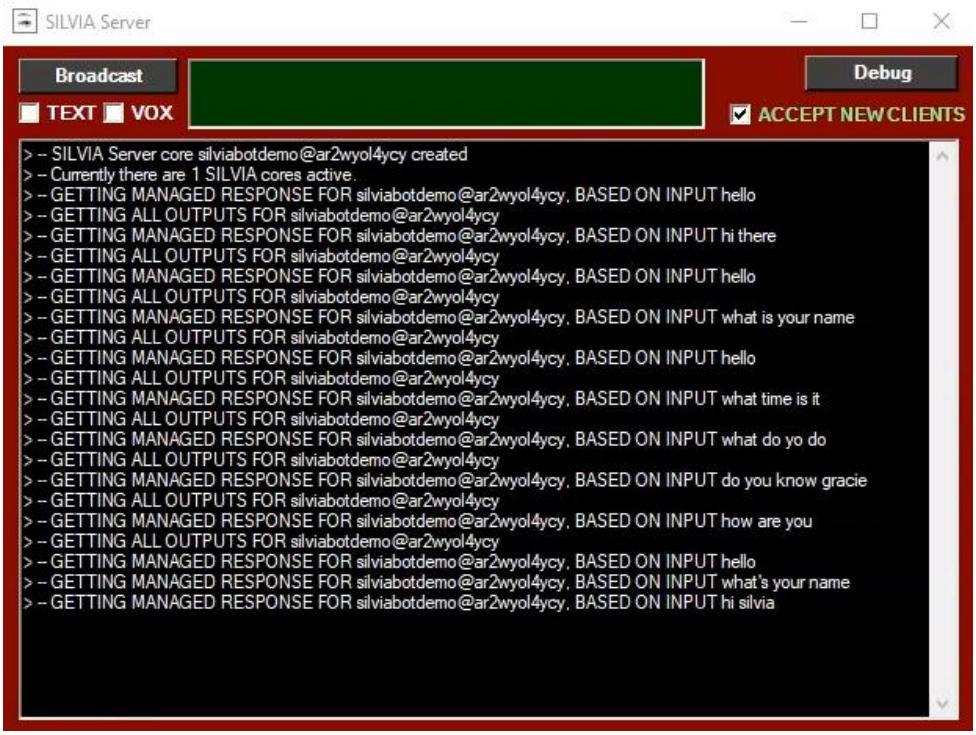
Given an existing client/server network of computers supporting TCP/IP:

- OS Requirements: Windows 10 or Windows Server 2019.
- Optionally, server or cloud-based speech recognition and text to speech technologies.

Installation of the Server

Cognitive Code provides a zipped installation folder that has all the elements needed to install and run the SILVIA Server.

1. Unzip and save the install folder provided by Cognitive Code.
2. Optional: change port, debug and log settings in the "SILVIAServerRest.bat" file:
    a. Port: the port default is 10870. This can be changed to an alternate port.
    b. Debug: the debug feature enables a message at every server interaction (See below). The debug default is "false". Set to "true" to turn on debug functionality. Turning on debug is recommended for best practice.
    c. Logfile: Logfile generates logs stored in the "consolelogs" folder. The logfile default is "false". Set it to "true" to generate log files.



Installation of SILVIA Rest Server

After installing the Silvia Server, complete the following to install and test the Silvia rest server:

1. Confirm you have a valid silviaclient.license file.
2. Ensure to set the silviaserver.exe compatibility check to run as Administrator.
3. Launch server with silviaserverrest.bat. (Note: the forms interface is part of the server application, closing this will also kill the web service. You do not need to manually kill the silviaserver.exe process in the task manager).
4. Test the server: cut, paste and modify the example calls below to test[1]. Results for each call will appear in the browser in an easy-to-read format.

[1] Note—In all REST calls, the id as the last parameter to identify each interaction is Optional. -1 will be returned if no integer id is supplied.

Tests for SILVIA Rest Server		
Test	**Pathway**	**Example**
Software version check	getsoftwareversion	localhost:10870/silviaserver/getsoftwareversion
Create the core for the identified user Note: the file parameter is the silvia .sil file that will be loaded for that user	Core/Create/user/file/id	localhost:10870/silviaserver/Core/Create/<user>/silviaandroiddemo.sil/0
Release (destroy) the core for the identified user	Core/Release/user/id	localhost:10870/silviaserver/Core/Release/<user>/1
Check: core exists for the identified user	Core/Exists/user/id	localhost:10870/silviaserver/Core/Exists/<user>/2
Current core count - reserved for admin use	Core/Count/user/id	localhost:10870/silviaserver/Core/Count/admin/3
Send input to named core on server for the identified user	IO/setinputmanaged (Does not return an AI response); call io/gettext (shown below)to get response(s) after each call to io/setinputmanaged	localhost:10870/silviaserver/IO/setinputmanaged/<user>/hello there/5 localhost:10870/silviaserver/IO/setinputmanaged/<user>/my name is leslie/6 localhost:10870/silviaserver/IO/setinputmanaged/<user>/who am i/7
Send an unmanaged input for the identified user Note: This is a shortcut to quick interactions and can return an immediate response.	Not officially supported	localhost:10870/silviaserver/IO/setinput/<user>/how are you today/8
Retrieve top of text output stack for the identified user	IO/gettext/user/id (keep calling IO/gettext until no result)	localhost:10870/silviaserver/IO/gettext/<user>/4
Retrieve top of voice output stack for the identified user; feed the result text, if any, into a tts engine	IO/getvoice/user/id (Keep calling io/getvoice until no result)	localhost:10870/silviaserver/IO/getvoice/<user>/9

Tests for Silvia Rest Server (Cont.)		
Test	**Pathway**	**Example**
Retrieve top of application message stack for the identified user	io/getmessage/user/id (keep calling io/getmessage until no result)	localhost:10870/silviaserver/IO/getmessage/<user>/11
Retrieve top of data output stack for the identified user Note: when behaviors, absorbers, or exuders are triggered through Interactions, the values, if any, contained in those elements' "data" fields are returned.	IO/getdata/user/id (keep calling io/getdata until no result)	localhost:10870/silviaserver/IO/getdata/<user>/14
Retrieve top of diagnostic output stack for the identified user	IO/getdiagnostic/user/id (keep calling io/getdiagnostic until no result)	localhost:10870/silviaserver/IO/getdiagnostic/<user>/12
Retrieve all pending text/voice/message/data/diagnostic outputs for the identified user Notes: • Unlike the other io/get methods, you will not need to keep calling until no result. • Use once per update loop to retrieve all pending messages on the stack at once.	IO/getall/user/id	localhost:10870/silviaserver/IO/getall/<user>/15

Appendix

Other Flags available for startup command in batch file

NOTE: COMMAND FLAGS MAY BE IN ANY ORDER

-debug
Enable or disable verbose "debug" output in the SILVIA Server console.
Debug console data is streamed and saved in the "consolelogs" subfolder.

argument: boolean (true/false), default: false
example: -debug true

-audiopath
Set the absolute or relative path for TTS audio file output.
Enables file-based audio output from TTS.
argument: string, default: none
example: -audiopath C:\testaudio\hereitis

-logfile

Enable or disable logging of conversational interactions.
Per-user conversational data is streamed and saved in the "logs" subfolder.

argument: boolean (true/false), default: false
example: -logfile true

-fps
Set the viseme frames-per-second for output to text files.
Each audio file will have an associated .txt file with single-space
separated viseme numbers.

argument: integer, default: 15
example: -fps 20

-voicecount
Set the maximum number of active TTS voices.
Multiple voices may be created and used for optimization of TTS output.

argument: integer, default: 1
example: -voicecount 5

-maxcores

Set the maximum number of concurrent users for this instance of SILVIA Server.
Appropriate for performance tuning and load balancing.

argument: integer, default: unlimited
example: -maxcores 10000

-timeout
Set the maximum number of minutes that a client-side user may remain inactive.
SILVIA Server will automatically disconnect from inactive users.

argument: integer, default: 60
example: -timeout 5

-name
Set the name of the TCP Channel.
argument: string, default: SILVIATCP
example: -name mySILVIAServer

-tcpport
Set the port number for communication with this instance of SILVIA Server.

argument: integer, default: 8674
example: -tcpport 2001

-ttsremote
Set and enable one or more port numbers of SILVIA Server instances on the same network.
These remote SILVIA Servers may be used as dedicated TTS Engines.
The remote TTS Servers will be driven by this SILVIA Server instance.

argument (count): integer, default: 0
arguments (ports): integer, defaults: none
example: -ttsremote 4 2002 2003 2004 2005

-ttsremotemode
If ttsremote is enabled, sets the method of request distribution amongst multiple TTS servers.
"dynamic": remote SILVIA TTS servers are accessed based on availability
"static": remote SILVIA TTS servers are accessed sequentially

argument (mode): string, default: dynamic
example: -ttsremotemode static

SILVIA Server TCP/IP Data Access Method Calls

Method: GetBehaviorData

- Retrieves a piece of SILVIA brain data from a particular SILVIA Server Core. This data is variable, and specified by the datatype field and one or more parameter fields.

SEND TO TCP/IP PORT:
SILVIA.server getbehaviordata -id 42 -user "leslie" -dataType absorberCount -parameters 14
RECEIVE FROM TCP/IP PORT:
SILVIA.server getbehaviordata -id 42 -success true -result 3 -eom
POSSIBLE ERROR MESSAGE EXAMPLE
SILVIA.server getbehaviordata -id 42 -success false -error "No data returned." -eom

Method: SetBehaviorData

- Sets (updates) a specified piece of brain data to a particular SILVIA Server Core. This data is variable, and specified by the datatype field and one or more parameter fields. In most cases, the datatype and parameters are identical to GetBehaviorData messages, with the addition of the -data field, used to specify what is written to the target field in the brain data.

SEND TO TCP/IP PORT:
SILVIA.server setbehaviordata id -43 -user "leslie" -dataType absorberText -parameters 14 2 -data "This is my absorber text"
RECEIVE FROM TCP/IP PORT:
SILVIA.server setbehaviordata -id 43 -success true -result true -eom
POSSIBLE ERROR MESSAGE EXAMPLE
SILVIA.server setbehaviordata -id 43 -success false -error "No data set." -eom

Method: AddBehaviorData

- Adds a specified piece of brain data to a particular SILVIA Server Core. This data is variable, and specified by the datatype field and one or more parameter fields. There are currently only three variations necessary for this method: the addition (creation) of a behavior, the addition of an absorber to a particular behavior, and the addition of an exuder to a particular behavior. When successful, the returned message will include the integer ID of the newly created element as a result.

SEND TO TCP/IP PORT:
SILVIA.server addbehaviordata -id 49 -user "leslie" -dataType behavior -parameters mygroup myname
RECEIVE FROM TCP/IP PORT:
SILVIA.server getbehaviordata -id 49 -success true -result 1027 -eom
POSSIBLE ERROR MESSAGE EXAMPLE
SILVIA.server getbehaviordata -id 49 -success false -error "No data added." -eom

GETTING DATA: SAMPLES

int BehaviorCount()
SENT: SILVIA.Server GetBehaviorData -id 264 -user "leslie" -dataType count
RECV: SILVIA.server getbehaviordata -id 264 -success true -result "1085" -eom

int ID(String group, String name)
SENT: SILVIA.Server GetBehaviorData -id 265 -user "leslie" -dataType id -parameters mybehaviorgroup mybehaviorname
RECV: SILVIA.server getbehaviordata -id 265 -success true -result "1022" -eom

int[] IDList(string group0, string name0, string group1 string name1, ...)
SENT: SILVIA.Server GetBehaviorData -id 265 -user "leslie" -dataType idlist -parameters mygroup myname0 mygroup myname2
RECV: SILVIA.server getbehaviordata -id 265 -success true -result "1022 1024" -eom
string Group()
SENT: SILVIA.Server GetBehaviorData -id 364 -user "leslie" -dataType group -parameters 1022
RECV: SILVIA.server getbehaviordata -id 364 -success true -result "mybehaviorgroup" -eom
string SubGroup()
SENT: SILVIA.Server GetBehaviorData -id 365 -user "leslie" -dataType subgroup -parameters 1022
RECV: SILVIA.server getbehaviordata -id 365 -success true -result "mybehaviorsubgroup" -eom
string Name()
SENT: SILVIA.Server GetBehaviorData -id 366 -user "leslie" -dataType name -parameters 1022
RECV: SILVIA.server getbehaviordata -id 366 -success true -result "mybehaviorname" -eom

int AbsorberCount(int behaviorID)
SENT: SILVIA.Server GetBehaviorData -id 266 -user "leslie" -dataType absorberCount -parameters 14
RECV: SILVIA.server getbehaviordata -id 266 -success true -result "4" -eom

int ExuderCount(int behaviorID)
SENT: SILVIA.Server GetBehaviorData -id 267 -user "leslie" -dataType exuderCount -parameters 14
RECV: SILVIA.server getbehaviordata -id 267 -success true -result "4" -eom

string AbsorberText(int behaviorID, int absorberID)
SENT: SILVIA.Server GetBehaviorData -id 268 -user "leslie" -dataType absorberText -parameters 14 3
RECV: SILVIA.server getbehaviordata -id 268 -success true -result "here is my absorber" -eom

int ExuderText(int behaviorID, int exuderID)
SENT: SILVIA.Server GetBehaviorData -id 269 -user "leslie" -dataType exuderText -parameters 14 2
RECV: SILVIA.server getbehaviordata -id 269 -success true -result "here is my exuder" -eom

string BehaviorData(int behaviorID)
SENT: SILVIA.Server GetBehaviorData -id 299 -user "leslie" -dataType behaviorData -parameters 14
RECV: SILVIA.server getbehaviordata -id 299 -success true -result "here is my arbitrary attached behavior data" -eom

string AbsorberData(int behaviorID, absorberID)

SENT: SILVIA.Server GetBehaviorData -id 300 -user "leslie" -dataType absorberData -parameters 14 2
RECV: SILVIA.server getbehaviordata -id 300 -success true -result "here is my arbitrary attached absorber data" -eom

string ExuderData(int behaviorID, exuderID)
SENT: SILVIA.Server GetBehaviorData -id 301 -user "leslie" -dataType exuderData -parameters 14 1
RECV: SILVIA.server getbehaviordata -id 301 -success true -result "here is my arbitrary attached exuder data" -eom

string AllAbsorberText(int behaviorID)
SENT: SILVIA.Server GetBehaviorData -id 270 -user "leslie" -dataType allAbsorberText -parameters 14
RECV: SILVIA.server getbehaviordata -id 270 -success true -result "here is my absorber -eol here is another absorber -eol here is a third absorber -eol" -eom

int AllExuderText(int behaviorID)
SENT: SILVIA.Server GetBehaviorData -id 271 -user "leslie" -dataType allExuderText -parameters 14
RECV: SILVIA.server getbehaviordata -id 271 -success true -result "here is my exuder -eol here is my second exuder -eol here is my third exuder -eol" -eom

SETTING DATA: SAMPLES

Group(int behaviorID, string data)
SENT: SILVIA.Server SetBehaviorData -id 464 -user "leslie" -dataType group -parameters 1022 -data mynewgroup
RECV: SILVIA.server setbehaviordata -id 464 -success true -eom

SubGroup(int behaviorID, string data)
SENT: SILVIA.Server SetBehaviorData -id 465 -user "leslie" -dataType subgroup -parameters 1022 -data mynewsubgroup
RECV: SILVIA.server setbehaviordata -id 465 -success true -eom

Name(int behaviorID, string data)
SENT: SILVIA.Server SetBehaviorData -id 466 -user "leslie" -dataType name -parameters 1022 -data mynewname
RECV: SILVIA.server setbehaviordata -id 466 -success true -eom

AbsorberText(int behaviorID, int absorberID, string data)
SENT: SILVIA.Server SetBehaviorData -id 468 -user "leslie" -dataType absorberText -parameters 14 3 -data "this is an updated absorber"
RECV: SILVIA.server setbehaviordata -id 468 -success true -eom

ExuderText(int behaviorID, int exuderID, string data)
SENT: SILVIA.Server SetBehaviorData -id 469 -user "leslie" -dataType exuderText -parameters 14 2 -data "This is an updated exuder."

RECV: SILVIA.server setbehaviordata -id 469 -success true -eom

BehaviorData(int behaviorID, string data)
SENT: SILVIA.Server SetBehaviorData -id 470 -user "leslie" -dataType behaviorData -parameters 14 -data "This is some arbitrary data."
RECV: SILVIA.server setbehaviordata -id 470 -success true -eom

AbsorberData(int behaviorID, int absorberID, string data)
SENT: SILVIA.Server SetBehaviorData -id 471 -user "leslie" -dataType absorberData -parameters 14 2 -data "This is some arbitrary absorber data."
RECV: SILVIA.server setbehaviordata -id 471 -success true -eom

ExuderData(int behaviorID, int exuderID, string data)
SENT: SILVIA.Server SetBehaviorData -id 472 -user "leslie" -dataType exuderData -parameters 14 1 -data "This is some arbitrary exuder data."
RECV: SILVIA.server setbehaviordata -id 472 -success true -eom

ADDING DATA: SAMPLES

int Behavior(string group, string name)
SENT: SILVIA.Server AddBehaviorData -id 564 -user "leslie" -dataType behavior -data mynewname mynewgroup
RECV: SILVIA.server addbehaviordata -id 564 -success true -result "1138" -eom

int Absorber(int behaviorID, string data)
SENT: SILVIA.Server AddBehaviorData -id 568 -user "leslie" -dataType absorber -parameters 14 -data "this is a new absorber"
RECV: SILVIA.server addbehaviordata -id 568 -success true -result "5" -eom

int Exuder(int behaviorID, string data)
SENT: SILVIA.Server AddBehaviorData -id 569 -user "leslie" -dataType exuder -parameters 14 -data "This is a new exuder."
RECV: SILVIA.server addbehaviordata -id 569 -success true -result "3" -eom