Genius Agent API (AAPI)

Table of contents

Introduction

Welcome

The Genius Agent API (AAPI) is a WebSockets-based extensible API that allows interaction with the Genius Dialler platform. Primarily designed for use by the Genius SuperScripter, it also allows easy integration for clients wishing to augment or replace SuperScripter with their own in-house CRMs, DCAs and other agent applications.

Version history

V1.0 (29/09/2017) - Initial draft

V1.1 (23/10/2017)
Added detailed transfer, manual and follow-on call flow information
Added additional error messaging

V1.2 (16/11/2017) - Added TagCall

V1.3 (29/04/2018) - Added documentation for KeepAlive

V1.4 (08/11/2019) - Added queuenumber to NowInCall

V1.5 (21/05/2020) - Added ChapterMark

V1.6 (29/11/2021) - Added uniqueCallID to NowInCall

V1.7 (11/02/2022) - Added GetWebRTC

V2.0 (02/05/2022)
Added Omnichannel options
Retired *RunTool* and *TogglePark* commands
Added Second-Factor support to authentication

V2.1 (21/11/2023)
Clarified terminology on transfer modes and options

Connection and Communication

Connecting to the API

Connection is over a standard Secure WebSockets (WSS) interface. Most modern languages support WebSockets, with 3rd party extensions and libraries commonly available to enhance and extend functionality.

The connection address to be used will be supplied by your Genius account manager. It will typically be in the form **aapi.geniusppt.com**, which should then normally be addressed in code as

wss://aapi.geniusppt.com

Important: TLS1.2 or later must be used to connect to the service.

Mechanism

Messages sent to and from the AAPI are in JSON format, and normally contain two vital components: the SessionID (SID) and a RequestID (RID).

JSON conventions are followed throughout the API: line breaks are optional, but strings must be delimited with quotes, integers and booleans are not, and arrays of values follow standard notation.

The site http://json.parser.online.fr is a useful tool for checking the formatting of JSON messages.

SessionID (SID)

The SessionID (SID) is a unique 32-bit integer that acts reference for the agents connection to the API service - each connection attempt will receive a distinct SID, even if the same agent credentials are used. During authentication, a SID is generated by the API service, and all subsequent requests to the API must use this SID.

RequestID (RID)

The RequestID (RID) is an ephemeral 32-bit integer reference number for a request to the API. As responses from the API are asynchronous, the RID is used to tie together responses with the original requests.

Message format

A request to the dialler is a JSON message containing key-value pairs; normally the SID and RID, along with a "type" key which is the command to be sent.

For example, a GoAvailable request would look like

```
{
  "type":"GoAvailable",
  "RID":468262020,
  "SID":1489758398
}
```

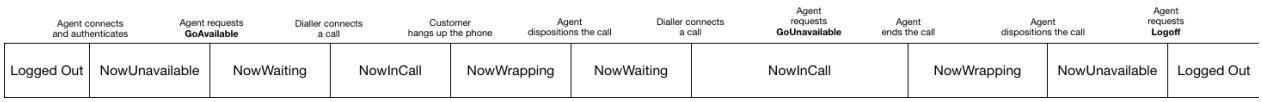
Messages may also require JSON arrays of values - for example, a Disposition message has data to be returned

```
{
  "type":"Disposition",
  "RID":659034978,
  "SID":1489758398,
  "outcome":251,
  "callbackNumber":"",
  "callbackTime":"",
  "personalCallback":false,
  "data":{
    "URN":"1234-test",
    "Client":"test",
    "Name":"test",
    "JobTitle":"test",
    "Email":"test@test.com"
        }
}
```

Dialler Basics

Standard cycle

In normal environments, Agents logging into the dialler go through a number of states during the course of their session.



- From an initial logged out state, the agent logs into the dialler. This provides the initial authentication to the service, but they are not yet connected to the dialler.
- The agent "nails up" a phone connection to their desk.
- Once this is successful, the agent is logged in but in an **Unavailable** state - they are connected but will not make or receive any automated calls.The agent can make a manual, non-automatic call at this point.
- To signal to the dialler that the agent wishes to start making and receiving automated calls, a **GoAvailable** request needs to be sent.
- This puts the agent into the pool of agents to make and receive calls, and the agent is waiting to be connected to a call. The agent will receive a **NowAvailable** event, quickly following by another event signalling that the agent is now in the state **NowWaiting**.
- When a call is connect to an agent, they are in a **NowInCall** state.
- When the call ends - either by the customer terminating the call, or the agent ending the call - the agent moves to a **NowWrapping** state.
- The agent must now disposition the call. Once the call is dispositioned, they move back to a state of **NowWaiting**.
- When the agent would like to leave this pool, they need to request that they **GoUnavailable**; this signals to the dialler that the agent will be leaving the pool as soon as it's safe. To avoid generating any unnecessary abandoned calls, this request may not be granted immediately.
- Once the agent is in an Unavailable state, they'll receive a **NowUnavailable** event. They can either rejoin the pool via **GoAvailable**, make a manual call, or log out of the dialler.

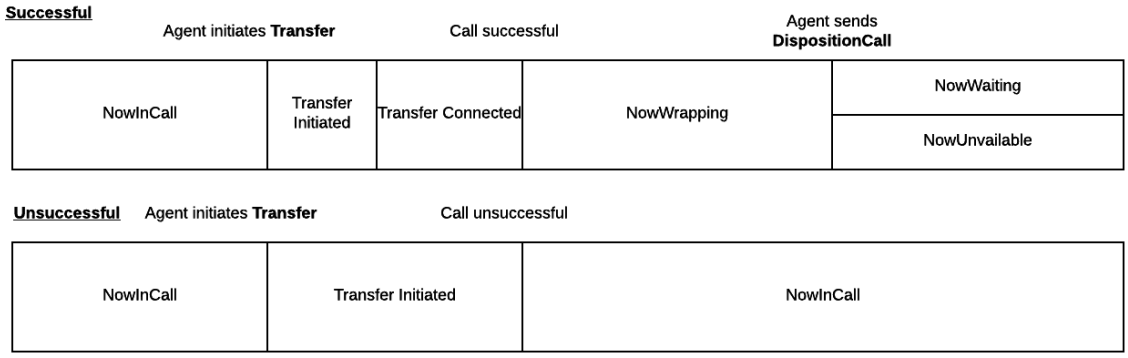
The AAPI contains all commands required to move the agents through these states, as well as events to signal when the state has changed.

Transferring calls

There are three types of transfers - Blind Without Connect, Blind With Connect, and Conference

Blind transfer without waiting for a connect

The call is simply pushed to the recipient - the call is transferred and the agent moves into wrap. For example, transferring the call to an inbound queue will result in the customer being placed into that queue and hearing "on hold" music, as if they'd went straight into that queue.



- Agent is in a call and sends a **TransferCall** request.
- The dialler then initiates the call transfer and returns a **TransferUpdate** event with TransferStatus of 3

If the transfer recipient is available and the call connects

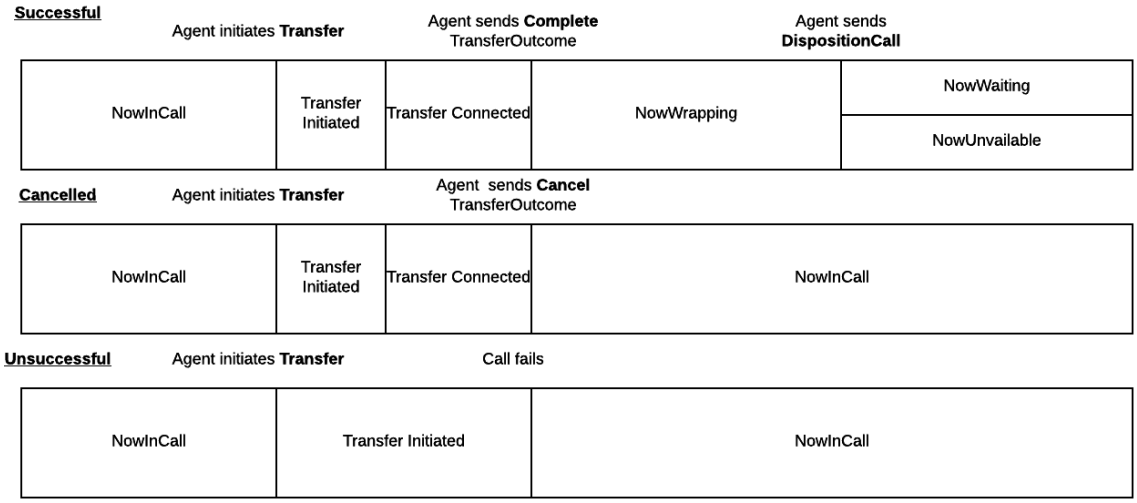
- A **TransferUpdate** event with TransferStatus of 0
- The Agent is sent a **NowWrapping** event
- The Agent then dispositions the call, and receives a **NowWaiting** event or if the initial call was a ManualDial the Agent receives a **NowUnavailable** event.

If the transfer recipient is unavailable and the call does not connect

- The agent receives a TransferUpdate event with TransferStatus of 2.
- The Agent is returned to the call

Blind transfer, waiting for connect

The call is placed on hold, and the Agent waits for the transferring party to answer - for example, if the call is being transferred into an inbound queue, the agent will wait for the call to be answered by another agent. Once the transfer recipient answers the call, the agent can complete the transfer (which moves the agent into wrap). If the agents cancels the transfer whilst waiting for a connection, or the recipient rejects the transfer, or the transfer is unsuccessful, the call returns to the agent.



The call flow for Blind With Connect transfers is similar to Blind Without Connect transfers with the addition of the Agent having the ability to **Cancel** and **Complete** transfer and also **Park** the recipient party

- Agent is in a call and sends a **TransferCall** request.
- The dialler then initiates the call transfer and returns a **TransferUpdate** event with TransferStatus of 3

If the transfer recipient is available and the call connects

- A **TransferUpdate** event with TransferStatus of 0
- The Agent is sent a **NowWrapping** event
- The Agent then dispositions the call, and receives a **NowWaiting** event or if the initial call was a ManualDial the Agent receives a **NowUnavailable** event.

If the transfer recipient is unavailable and the call does not connect

- The agent receives a TransferUpdate event with TransferStatus of 2.
- The Agent is returned to the call

When the agent completes the transfer

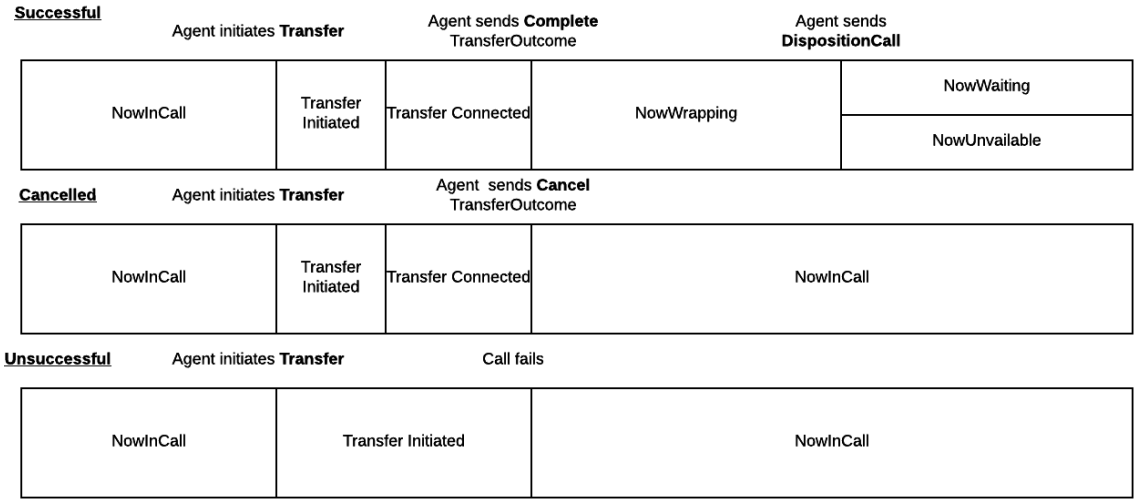
- The agent receives an OK message.
- The agent then enters a **NowWrapping** state so receives a **NowWrapping** message.
- The Agent then dispositions the call, and receives a **NowWaiting** event or if the initial call was a ManualDial the Agent receives a **NowUnavailable** event.
-

If the Agent cancels the transfer

- The Agent receives an 'OK' message back after sending TransferOutcome 'cancel';
- Agent is returned to **NowInCall** state and a **NowInCall** event is received

Conference Transfer

The call is placed on hold and the Agent is connected to the recipient. If the transfer recipient answers the call, the agent, caller and recipient are all connected together on the same call; the agent can then complete the transfer (which moves the agent into wrap). If the agents cancels the transfer, or the recipient rejects the transfer, or the transfer is unsuccessful, the call returns to the agent.



The call flow for Conference transfers is similar to Warm transfers and also allows the Agent the ability to **Cancel** and **Complete** transfer and also **Park** the recipient and caller

- Agent is in a call and sends a **TransferCall** request.
- The dialler then initiates the call transfer and returns a **TransferUpdate** event with TransferStatus of 3

If the transfer recipient is available and the call connects

- A **TransferUpdate** event with TransferStatus of 0
- The Agent is sent a **NowWrapping** event
- The Agent then dispositions the call, and receives a **NowWaiting** event or if the initial call was a ManualDial the Agent receives a **NowUnavailable** event.

If the transfer recipient is unavailable and the call does not connect

- The agent receives a **TransferUpdate** event with TransferStatus of 2.
- The Agent is returned to the call.

When the agent completes the transfer

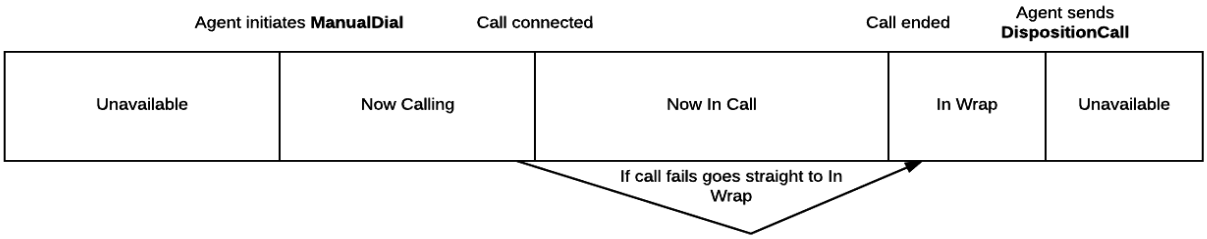
- The agent receives an OK message.
- The agent then enters a **NowWrapping** state so receives a **NowWrapping** message.
- The Agent then dispositions the call, and receives a **NowWaiting** event or if the initial call was a ManualDial the Agent receives a **NowUnavailable** event.

If the Agent cancels the transfer

- The Agent receives an 'OK' message back after sending TransferOutcome 'cancel';
- Agent is returned to **NowInCall** state and a **NowInCall** event is received

Manual call

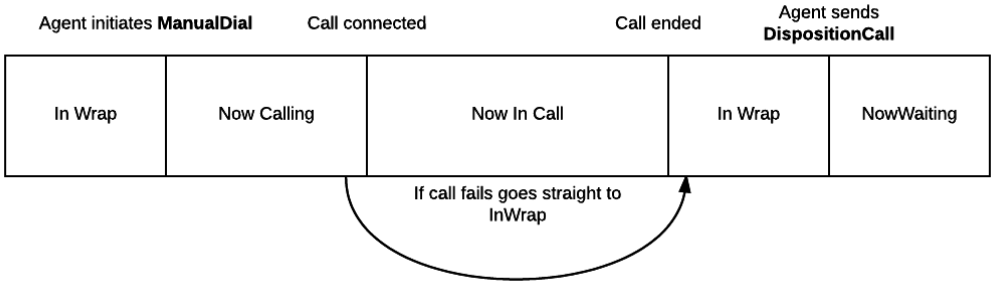
A manual call is made outside of a campaign, and is completely "ad hoc".



- From an **Unavailable** state, the Agent sends a ManualDial request
- The Dialler initiates the call and returns a NowCalling event
- Once the call is connected, the Agent is sent a NowInCall event and is in a **Talking** state.
- After the call is complete, a NowWrapping event is sent
- The Agent must now Disposition the call
- Once dispositoned, a NowUnavailable event is sent.

Follow-on calls

A follow-on call is a redial of a previously presented record, whether dialled manually, or automatically inbound or outbound.In most cases, this is to reconnect with a caller who disconnected, or has requested an immediate call on a different number.



- From an **Wrapping** state, the Agent sends a **ManualDial** request
- The Dialler initiates the call and returns a **NowCalling** event
- Once the call is connected, the Agent is sent a **NowInCall** event
- The call then continues to completion at which point the Agent is sent in a **NowWrapping** event
- The Agent must now **Disposition** the call
- After disposition, the agent will be sent a **NowWaiting** event.

Omnichannel and SessionIndex

The Genius dialler platform supports many types of communication channel: not just voice, but text and video-based conversations. There are three broad classes of session:

- LIVE - voice or video calls, where the agent is conversing in real-time with a customer directly using audio or video.
- INTERACTIVE - web-chat, WhatsApp, Facebook Messenger and other "chat" mechanisms, where the agent is conversing with a customer in real-time over a text-based medium.
- OFFLINE - email, SMS, Twitter, Facebook messages, where an agent is responding to a communication from a customer, but the customer is not necessarily waiting for an immediate response.

An agent can be handling a number of different channels at once. By necessity, the agent can only be taking one Live call at once, but they can also simultaneously be handling any number of Interactive and/or Offline calls at the same time. The number of simultaneous channels passed to the agent is controlled by the agent or campaign configuration, and can also be controlled via the API.

As one agent may be taking more than one session simultaneously, some AAPI calls require the specification of a "SessionIndex". The SessionIndex will be presented on any session-specific events being passed back from the dialler, and this same SessionIndex must be supplied on applicable commands.

By convention, Live calls are always on SessionIndex 0. There is a nominal limit of 9 concurrent sessions per agent, but this may increase in the future.

If no SessionIndex is supplied in a command, SessionIndex 0 is assumed.

API commands

The commands below are all accepted by the AAPI service. Unless explicitly stated, all parameters are mandatory - if no value is to be passed, a null, 0 or empty string should be sent.

Connecting and Authentication

The Connection and Authentication commands control the connection to the AAPI and Genius Dialler, including the initial validation of credentials and establishing of the telephone connection to the agent desk.

Authenticate

Attempts to authenticate a new agent session to the AAPI.

Request

Key	Value	Description
type	Authenticate	
RID	RequestID	
Agent	*string*	The unique agent username
Password	*string*	The agents password
extension	*string*	The DDI or extension number of the agents desk or soft phone. This can be left blank if establishing a WebRTC or offline/interactive session only.
SecondFactor	*string*	Optional - if 2FA is enabled, and the previous authentication attempt has returned an error 107 ("Needs second factor"), repeat the authentication attempt with the 6-digit code obtained from the user
DeviceID	*string*	Optional - If a DeviceID has been previously stored by a successful 2FA authentication, pass this through to the AAPI to validate if another 2FA authentication needs to be performed, or if the stored device is still valid. This is used to facilitate a "don't ask for this code for x number of days" feature on the client
StoreDevice	*integer*	Optional - The number of days to store a successful 2FA authentication against the device ID

Expected response

Key	Value	Description
Response	OK	
RID	RequestID	
SID	*integer*	The SessionID assigned to the session by the AAPI. This must be used in all subsequent requests for the session
DeviceID	*string*	When a successful 2FA authentication has been performed, a unique DeviceID will be generated. This can be used in subsequent authentications to bypass 2FA for a specified number of days.

GetTenants

Asks for a list of tenants available to this session.

Request

Key	Value	Description
type	GetTenants	
SID	SessionID	
RID	RequestID	

Expected response

Key	Value	Description
RID	RequestID	
tenants	*A JSON array of strings*	Contains the lists of tenants available to the agent

GetCampaigns

Asks for a list of campaigns available to this session within a selected tenant

Request

Key	Value	Description
type	GetTenants	
SID	SessionID	
RID	RequestID	
tenant	*string*	The name of the dialler tenant

Expected response

Key	Value	Description
RID	RequestID	
campaigns	*A JSON array of strings*	Contains the lists of campaigns available to the agent in the selected tenant

Connect

Attempts to establish a connection to the dialler, including the telephony connection to the agent phone.

Request

Key	Value	Description
type	Connect	
SID	SessionID	
RID	RequestID	
tenant	*string*	The name of the dialler tenant
campaign	*string*	The name of the initial campaign

Expected response

Key	Value	Description
RID	RequestID	
status	Connected	Agent has been successfully logged into the dialler and the telephony connection has been established

Logoff

Attempts to log the agent out of the dialler and AAPI.

Request

Key	Value	Description
type	Logoff	
SID	SessionID	
RID	RequestID	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

KeepAlive

Any connected client must periodically send a "KeepAlive" message, to notify the server that the client is still connected. We recommend a "KeepAlive" is sent at least every 30 seconds; any AAPI command received will reset the KeepAlive timer at the server side.

Request

Key	Value	Description
type	KeepAlive	
SID	SessionID	

Expected response

None

GetWebRTC

Requests a new WebRTC extension for connection to the dialler. The extension details will be unique and should be used only once. The agent extension number specified during the Authenticate will be overwritten with the extension number generated here.

Request

Key	Value	Description
type	GetWebRTC	
SID	SessionID	
RID	RequestID	
tenant	*string*	The name of the dialler tenant
campaign	*string*	The name of the initial campaign

Expected response

Key	Value	Description
RID	RequestID	
response	OK	
webrtc-extension	string	These should be supplied to the local SIP client and used to establish a WebRTC-based connection to the Genius platform.
webrtc-password	string	
webrtc-uri	string	

Availability and Call Flow

The Availability and Call Flow commands are used for signalling to the dialler whether agents should start or stop receiving automatic calls, as well as making ad-hoc or follow-on calls. Call and screen recording control is also available.

GoAvailable

Signals to the dialler that the agent wishes to join the automatic dialling call pool, and will be connected automatically to inbound and outbound calls and messages. GoAvailable can be sent whilst already in the dialling pool, if the intention is to update the availability for different session types.

Request

Key	Value	Description
type	GoAvailable	
SID	SessionID	
RID	RequestID	
live	One of three values: • -1 : go available with the existing setting for live calls • 0: go available with no live access • 1: go available with explicit live access (default)	
offline	One of these values: • -2: go available with the existing number of offline sessions • -1: go available with no offline session access • 0: go available with the configured offline session access for the campaign • *x:* go available with access to a maximum of *x* concurrent offline sessions	
interactive	One of these values: • -2: go available with the existing number of interactive sessions • -1: go available with no interactive session access • 0: go available with the configured interactive session access for the campaign • *x:* go available with access to a maximum of *x* concurrent interactive sessions	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

GoUnavailable

Signals to the dialler that the agent wishes to leave the automatic dialling call pool as soon as possible.

Request

Key	Value	Description
type	GoUnavailable	
SID	SessionID	
RID	RequestID	
breakReason	*integer*	The "reason code" why the agent wishes to go unavailable - this corresponds to a lookup table for reporting.

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

Disposition

Dispositions the call on the dialler. This ends the current call (if the agent isn't already in wrap), records the outcome of the call and returns the record to the database. Typically the agent will then move on to waiting for the next call, or move to Unavailable if a GoUnavailable has been requested.

Request

Key	Value	Description
type	Disposition	
SID	SessionID	
RID	RequestID	
outcome	*integer*	The result code for the call
callbackNumber	*string*	If a callback is to be booked, the number to use for the callback. Leave blank for no callback.
callbackTime	*string*	If a callback is to be booked, the time for the callback. Leave blank for no callback.
personalCallback	*boolean*	If a callback is to be booked, whether the callback should be attempted to be given to the original agent (true), or to any available agent (false). Set to false if no callback is being set.
data	*Key-Value pair array*	An array of values to be returned to the database. As many or as few values can be used as required.
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

StartWrap

Hangs up the current call, and starts the "wrap" process

Request

Key	Value	Description
type	StartWrap	
SID	SessionID	
RID	RequestID	
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

ManualDial

Make a manual call to a specified number. This can happen in two different scenarios:

- Redialling an already dialled record - to the previously dialled number or an alternative number. This is referred to as a "follow-on" call
- Dialling a record outside of a campaign, when the agent is in an Unavailable state.

Request

Key	Value	Description
type	ManualDial	
SID	SessionID	
RID	RequestID	
telNumber	*string*	The telephone number to be dialled

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

PauseRecording

Requests that the call (and option screen) recording is paused.

Request

Key	Value	Description
type	PauseRecording	
SID	SessionID	
RID	RequestID	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

ResumeRecording

Requests that the call (and option screen) recording is resumed.

Request

Key	Value	Description
type	ResumeRecording	
SID	SessionID	
RID	RequestID	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

ToggleHold

Toggles the agents "on hold" status

Request

Key	Value	Description
type	ToggleHold	
SID	SessionID	
RID	RequestID	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

Data

The Data commands allow database records, held at the Genius Dialler, to be queried, created and updated.

GetRecord

Retrieves a record from the dialler database

Request

Key	Value	Description
type	GetRecord	
SID	SessionID	
RID	RequestID	
URN	*string*	The Unique Reference Number of the record to be retrieve

Expected response

Key	Value	Description
RID	RequestID	
data	*Key-Value pair array*	An array of keys and values for the requested record

NewRecord

Creates a new record in the dialler database and returns the URN of the created record

Request

Key	Value	Description
type	NewRecord	
SID	SessionID	
RID	RequestID	
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
data	*Key-Value pair array*	An array of keys and values for the new record

RunQuery

Retrieves a record from the dialler database

Request

Key	Value	Description
type	RunQuery	
SID	SessionID	
RID	RequestID	
QueryNumber	*integer*	The number of the DataQuery that has been defined within the API. Queries can be created within SuperScripter Designer, or by Genius Support.
Parameters	*Key-Value pair array*	Parameter values to be passed into the query
values	*Key-Value pair array*	Data values to be passed into the query (normally used when INSERTing data)

Expected response

Key	Value	Description
RID	RequestID	
Data	*object*	Depending on the query, this may be an *integer*, a *string*, or a *Key-Value pair array*

MultiSale

Updates the "Multiple Sale" count for the current call. This is written out along with the disposition outcome for reporting.

Request

Key	Value	Description
type	MultiSale	
SID	SessionID	
RID	RequestID	
outcome	*integer*	The outcome number whose count is to be adjusted
quantity	*integer*	The amount by which the count is to be adjusted

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

Transfers

Transfer commands are used to control the transfer live calls between other agents, inbound queues or external numbers.

GetAgents

Returns a list of agents that are currently available for a call to be transferred to

Request

Key	Value	Description
type	GetAgents	
SID	SessionID	
RID	RequestID	

Expected response

Key	Value	Description
RID	RequestID	
Agents	*JSON array*	An array of agents available for transfer

TransferCall

Transfers the current call to another agent, queue or external number

Request

Key	Value	Description
type	TransferCall	
SID	SessionID	
RID	RequestID	
TransferType	*string*	The type of transfer to initiate - 0 - Blind With Connect 1 - Blind Without Connect 2 - Conference Transfer
TransferTo	*string*	The recipient transfer type - 0 - Named agent or extension number 1 - External number 2 - Inbound queue
Recipient	*string*	The number, agent name or queue number to transfer the call to
data	*Key-Value pair array*	The record data to be transferred along with the call
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

TogglePark

Toggles the Park

Request

Key	Value	Description
type	TogglePark	
SID	SessionID	
RID	RequestID	
Switch	*boolean*	true park the call, or false to recover from park
Who	*string*	"client" for the initially connected party, or "recipient" for the party receiving the transfer

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

Parking Notes

The term 'Parking' in relation to transfers is used as an alternative to 'Hold', in essence they have the same effect.

There are a couple of things to bear in mind in relation to 'Parking';

1. No Parking capability is available on 'Blind' transfers

2. On a 'Warm' transfer the recipient can be placed in 'Parked' state and retrieved from 'Parked' state, the caller remains in 'Parked' state until transfer is 'Complete' or 'Cancelled'

3. On a conference transfer the recipient and caller are both live unless the Agent selects to place either in a 'Parked' state and the Agent can place either into and retrieve them from a 'Parked' state.

All parties are removed from a 'Parked' state automatically on transfer 'Completion'.

The caller is retrieved from a 'Parked' state automatically on transfer 'Cancellation'.

TransferOutcome

Completes the call transfer

Request

Key	Value	Description
type	TransferOutcome	
SID	SessionID	
RID	RequestID	
Outcome	*string*	"cancel" to abandon the transfer, or "complete" to release the call to the transfer recipient
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
response	OK	

Miscelleneous

The AAPI has a number of miscellaneous commands that do not necessary fit within other categories.

Acknowledge

An Acknowledge must be sent in response to any Event generated by the AAPI server. This allows further synchronous events to be delivered by the AAPI without risk of events being processed out-of-order.

Request

Key	Value	Description
type	Acknowledge	
SID	SessionID	
RID	the RequestID of the event being acknowledged	

Expected response

None

SendMail

Sends a template email

Request

Key	Value	Description
type	SendMail	
SID	SessionID	
RID	RequestID	
template	*integer*	The ID of the template to be used (defined in GDMS)
recipient	*string*	The email recipient
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	

SendSMS

Sends a template SMS

Request

Key	Value	Description
type	SendSMS	
SID	SessionID	
RID	RequestID	
template	*integer*	The ID of the template to be used (defined in GDMS)
recipient	*string*	The email number
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	

Keypress

Simulates the pressing of keypad digits and sends DTMF tones on the live call.

Request

Key	Value	Description
type	Keypress	
SID	SessionID	
RID	RequestID	
keyPress	*string*	The DTMF number(s) to be simulated

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	

TagCall

Tags a call with a particular URN. This is normally used on inbound calls; this is purely for reporting purposes, and won't affect the currently dialled record.

Request

Key	Value	Description
type	TagCall	
SID	SessionID	
RID	RequestID	
URN	*string*	The URN/Reference/Account ID of the currently dialled record
sessionIndex	The SessionIndex of the call	

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	

ChapterMark

Marks a "chapter" in the call. This is used to aid playback of the call recording, by breaking the call into segments

Request

Key	Value	Description
type	ChapterMark	
SID	SessionID	
RID	RequestID	
chaptername	*string*	The name of the chapter to create

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	

GetMessageID

Retrieves an offline message with a specific ID. This includes any email attachments on the message. If attachments are not required, GetMessageHistory may be a lighterweight alternative.

Request

Key	Value	Description
type	ChapterMark	
SID	SessionID	
RID	RequestID	
messageID	*int*	The unique message ID, as contained in the record GeniusMessageID or GeniusMessageHistory data fields

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	
sender	string	The sender of the message
recipientsTo	array of string	The recipient(s) of the message
recipientsCc	array of string	Any other parties who were cc'd to this message (email only)
subject	string	The subject of the message (email only)
body	string	The body of the message; plain text for SMS, HTML for email
attachments	array of attachment	An array of attachment objects, containing: filename - the name of the file data - the binary data of the file inline - a boolean indicating this is an HTML inline attachment within an email

GetMessageHistory

Retrieves a number of offline messages, based on a list of specific message IDs.

Request

Key	Value	Description
type	ChapterMark	
SID	SessionID	
RID	RequestID	
messageList	*array of int*	An array of unique message IDs, as contained in the record GeniusMessageID or GeniusMessageHistory data fields

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	
messages	array of object	An array of message objects (see GetMessageHistory for descriptions) • sender • recipientsTo • recipientsCc • subject • body • ID

SendChatMessage

Sends a chat message

Request

Key	Value	Description
type	SendSMS	
SID	SessionID	
RID	RequestID	
message	*string*	The plain-text chat message to send
sessionIndex	*int*	The SessionIndex of the call

Expected response

Key	Value	Description
RID	RequestID	
Response	OK	

API events

As well as requests from an AAPI client to the server, there is the requirement for the AAPI to signal to the client that certain dialler events have occurred. These can, and will, happen without any request from the client. Events must be explicitly acknowledged by the client, to prevent a queue of events building at the server.

NowAvailable

Signals to the AAPI clients that the agent is now available for automatic calling, and will receive calls and messages in due course.

Request

Key	Value	Description
event	NowAvailable	
RID	RequestID	

Expected response

An Acknowledge.

NowUnavailable

Signals to the client that the agent is now unavailable for automatic calling.

Request

Key	Value	Description
event	NowUnavailable	
RID	RequestID	

Expected response

An Acknowledge.

NowWrapping

Signals to the client that the the live call has ended and the agent is now in wrap.

Request

Key	Value	Description
event	NowWrapping	
RID	RequestID	

Expected response

An Acknowledge.

NowWaiting

Signals to the client that the agent is waiting for a call

Request

Key	Value	Description
event	NowWaiting	
RID	RequestID	

Expected response

An Acknowledge.

NowRecording

Signals to the client that the call is now being recorded.

Request

Key	Value	Description
event	NowRecording	
RID	RequestID	

Expected response

An Acknowledge.

NowPaused

Signals to the client that call recording has been paused

Request

Key	Value	Description
event	NowPaused	
RID	RequestID	

Expected response

An Acknowledge.

NowInCall

Signals to the client that the agent has been connected to a call.

Request

Key	Value	Description
event	NowInCall	
RID	RequestID	
campaign	*string*	The campaign of the record being presented
calltype	*integer*	0 = Predictive Outbound 3 = Preview Outbound 4 = Progressive Outbound 5 = Inbound
tel	*string*	The telephone number used to dial the connected record, or the CLI presented by the inbound call (if available)
transfer	*boolean*	Whether the call is a transfer from another agent
sessionID	*string*	The internal name of the call session, not to be confused with the SID
queuenumber	*integer*	The internal queue number the call was connected for. For non-inbound calls, this will be 0.
data	*Key-Value pair array*	The record data
uniquecallid	*string*	A value uniquely representing the currently connected call

Expected response

An Acknowledge.

NowLoggedOff

Signals to the client that the agent is now logged out of the dialler and the AAPI.

Request

Key	Value	Description
event	NowLoggedOff	
RID	RequestID	

Expected response

An Acknowledge.

NowCalling

Status information on the progress of a manual dial

Request

Key	Value	Description
event	TransferUpdate	
RID	RequestID	

Expected response

An Acknowledge.

TransferUpdate

Status information on the progress of a call transfer

Request

Key	Value	Description
event	TransferUpdate	
RID	RequestID	
TransferStatus	*integer*	**0 -** Transfer connected. **1 -** Transfer disconnected: eg transferrer / transferee hangs up. **2 -** The telephony layer did not establish an inquiry call to the transferee **3 -** Transfer is being offered **4 -** Connected/Conferencing

Expected response

An Acknowledge.

ChatMessage

A chat message has been received

Request

Key	Value	Description
event	ChatReceived	
RID	RequestID	
sessionIndex	*integer*	The sessionIndex of the session containing the chat
sessionID	*string*	The sessionID of the session containing the chat
direction	*AgentToCustomer* *CustomerToAgent* *TransfererToTransferee* *TransfereeToTransferer*	An enumeration of who sent the message and to who
agent	*string*	The name of the agent sending or receiving the message
message	*string*	The plain-text message being sent or received

Expected response

An Acknowledge.

ChatTyping

An indication of whether the remote agent or customer is typing a message

Request

Key	Value	Description
event	ChatTyping	
RID	RequestID	
sessionIndex	*integer*	The sessionIndex of the session containing the chat
sessionID	*string*	The sessionID of the session containing the chat
direction	*AgentToCustomer* *CustomerToAgent* *TransfererToTransferee* *TransfereeToTransferer*	An enumeration of who sent the message and to who
agent	*string*	The name of the agent sending or receiving the message
typing	*boolean*	A true or false indication of whether the remote party is typing or not

Expected response

An Acknowledge.

Error Messages

Any request to the AAPI may result in an error message: rather than the standard response, the AAPI will return a message with -

Key	Value	Description
Response	ERROR	
RID	RequestID	
ErrorLevel	*string*	An arbitrary error level, for informational and categorisation only. Some clients may choose to only display errors above a certain criticality to agents.
ErrorNumber	*integer*	See below
ErrorMessage	*string*	See below

Error messages

Error Number	Error Level	Error Message	Information
99	Serious	Malformed message	The request sent to the AAPI isn't in the expected format. Please check the relevant page in the documentation for the format and expected parameters.
100	Critical	Username and/or password is not valid	The credentials supplied didn't match those stored in the AAPI database.
101	Critical	User is not authorised to log on	The account specified hasn't been given permission to log on via the AAPI
102	Critical	User is currently banned	The account specified has been blocked from using the AAPI, normally due to an abnormally high number of failed authentication requests
103	Critical	IP banned	Too many failed attempts have been made to authenticate from the IP address, and subsequent attempts are blocked
104	Critical	No password entered	An attempt was made to authenticate, but no password was specified
105	Critical	No agent name entered	An attempt was made to authenticate, but no agent name was specified
106	Critical	No agent name or password entered	An attempt was made to authenticate, but no agent

			name or password was specified
107	Critical	Requires Second Factor	2FA has been enabled for the agent; supply a valid 6-digit 2FA code or previously authenticated DeviceID
110	Critical	User has requested GETCAMPAIGNS for a tenant which they don't have access	Agent accounts are restricted to relevant dialler tenant, and an attempt was made to retrieve campaign information for a tenant to which the agent hasn't been granted access.
111	Critical	User has requested a script for a tenant and/or campaign that they don't have access to	An attempt was made to retrieve a SuperScript for a tenant to which the agent hasn't been granted access.
112	Critical	User has requested a script, but no script has been configured for the requested tenant and campaign	The AAPI was asked to send the SuperScript for a particular campaign, but no script has been assigned.
113	Serious	Campaign hasn't been started	A request was made to log in to a specific campaign, but the campaign isn't currently started.
114	Critical	Script requested but agent isn't connected	A request was made for a SuperScript, but the agent isn't actually connected to the dialler at this time.
120	Critical	User does not have permissions to access Designer	An attempt was made to open a Designer connection, but the user hasn't been granted permissions.
121	Critical	Command requires Designer login	The command being attempted is only valid for a Designer user.
150	Serious	Unable to establish a connection to the agent phone	The dialler was unable to connect to the agent via the specified telephone extension number or DDI.
151	Serious	Specified extension is busy	The dialler attempted to connect to the agent via the specified telephone extension number or DDI, but the number was busy.
160	Serious	Can't go available, agent is already available	A GoAvailable request was sent, but the agent is already in an available state.

161	Serious	Can't go unavailable, agent is already unavailable	A GoUnavailable request was sent, but the agent is already in an unavailable state
198	Serious	SID is unknown, please reauthenticate	A request was sent with an unknown SID - this may be due to inactivity on the SID without sending KeepAlives, or due to a service interruption.
199	Serious	The endpoint for the SID does not match. Please reauthenticate.	The IP endpoint used to initially authenticate the SID doesn't match the one now being used. Requests will be denied.
200	Critical	QueryID is unknown for this script	An attempt was made to run a DataQuery that doesn't exist, or isn't valid for the current script
201	Serious	Expected parameter not found in request	A DataQuery was attempted, but without a required parameter being given
202	Serious	Expected value not found in request	A DataQuery was attempted, but without a required value being given
210	Serious	SQL query failed	The DataQuery failed for an unknown reason
220	Serious	Agent has reached their request limit	Agents can be assigned limits on the number of DataQueries than can be performed, or the number of items of data that they can retrieve. This limit has been reached, and will be reset overnight.
300	Serious	Agent is not in an appropriate state to make a manual/follow-on call	An attempt was made to make a manual or follow-on call when the agent is not the Wrapping or Unavailable states.
301	Serious	Manual dial could not be made due to invalid telephone number	The manual call could not be established, as the specified phone number was invalid.
310	Serious	Transfer request contains invalid recipient value	The transfer could not be established, as the requested phone extension, external number, queue number or agent name was invalid.
311	Serious	Transfer request queue is in invalid format	The transfer could not be established, as the

			requested phone extension, external number, queue number or agent name was invalid.
320	Serious	Cannot disposition, agent is not in wrap	A Disposition was sent, but the agent is not wrapping
321	Serious	Cannot send DTMF, agent is not in a call	Sending DTMF failed, as the agent isn't currently on a live call
322	Warning	Cannot send DTMF, invalid digit(s) specified	The specified DTMF digits contained characters other than 0-9, * or #.
323	Serious	Cannot start wrap, agent is not talking	A StartWrap was sent, but the agent isn't currently on a call
324	Serious	Cannot start wrap, agent is already in wrap	A StartWrap was sent, but the agent is already in wrap
330	Serious	Cannot pause call, call is not currently being recorded or is already paused	A PauseRecording was received, but the call isn't being recorded, or may already be paused
331	Serious	Cannot resume call, call was already being recorded	A ResumeRecording was received, but the call was already being recorded
340	Warning	Cannot mark a chapter as the agent isn't talking	Chapters can only be marked when the agent is talking on a live call

Example code

C#/.Net

```
        private static WebSocket websocketConnection;
        private static ConcurrentDictionary<int, Dictionary<string, object>>
repliesReceived = new ConcurrentDictionary<int, Dictionary<string, object>>();

        public static Dictionary<string, object>
SendMessageAndWaitForResponse(Dictionary<string, object> sendList)
        {
            int RID = SendMessage(sendList);

            while (!repliesReceived.ContainsKey(RID))
            {
                Thread.Sleep(1000);
            }

            Dictionary<string, object> output;
            repliesReceived.TryRemove(RID, out output);
            if (output.ContainsKey("ErrorNumber") && (string)output["ErrorNumber"] ==
"198")
            {
                MessageBox.Show(
                    "Connection to the server has been reestablish, but we now need to
reauthenticate. Click OK to continue");

             var error = SocketClient.Authenticate(ScripterStuff.username,
ScripterStuff.password);

                while (error!=null)
                {
                    MessageBox.Show(
                   "Reauthentication failed. Try again?");
                    error = SocketClient.Authenticate(ScripterStuff.username,
ScripterStuff.password);
                }

                output= SendMessageAndWaitForResponse(sendList);
            }
            return output;
        }

        private static int SendMessage(Dictionary<string, object> sendList)
        {
            int RID = new Random().Next(0, Int32.MaxValue);
            if (sendList.ContainsKey("RID"))
            {
                sendList["RID"] = RID;

            }else sendList.Add("RID", RID);
            if (SID != 0) sendList.Add("SID", SID);

            string json = JsonConvert.SerializeObject(sendList);
            json = json.Substring(1, json.Length - 2);
            websocketConnection.Send(json);
            return RID;
        }
```

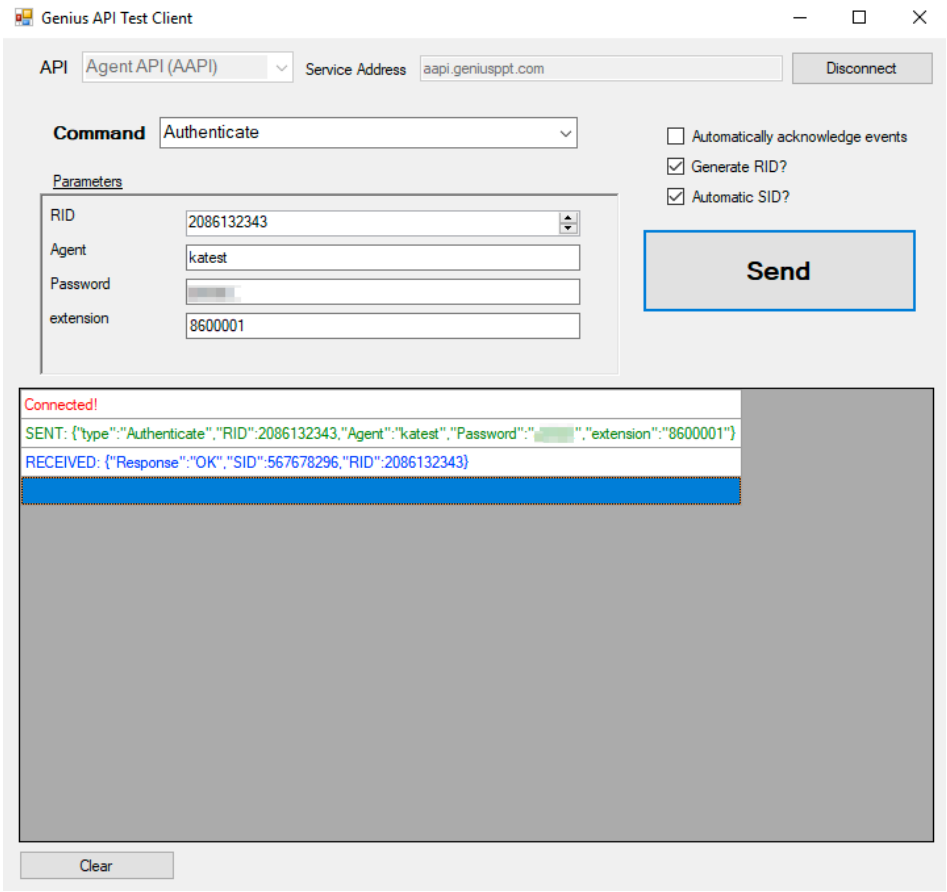
```
private static void ConnectSocket()
{
    websocketConnection = new WebSocket(server);

    websocketConnection.OnMessage += Ws_OnMessage;
    websocketConnection.OnClose += Ws_OnClose;
    websocketConnection.OnError += Ws_OnError;
    websocketConnection.Connect();
}
```

API Test Client

To assist with working with the AAPI, a Test Client application is available for download. This serves both as an example of an application using the AAPI, and as a way of testing and experimenting with AAPI commands in an easy-to-use and transparent utility.

The API Test Client is available for download at https://superscripter.geniusppt.com/APITestClient.exe

Note: Some browsers may detect this utility as a suspected virus. It's not!



All commands currently implemented by the AAPI are supported. Options are available to automatically generate the RID, use the SID provided by the AAPI service, to automatically acknowledge events as they occurs. We recommend all three options are used.