Construct API

Advanced Simulation Technology inc.

500 A Huntmar Park Drive • Herndon, Virginia 20170 U.S.A.
Tel. (703)471-2104 • Fax. (703)471-2108
www.asti-usa.com

Contents

The Construct API allows developers to dynamically add sounds and voiced radio transmissions to a simulation environment. HTTP clients are able to interact with Construct by making standard HTTP requests to the Voisus server. Simulation entities, interactions, and sounds are represented as JSON resources that can be manipulated by clients to create highly customized audio and radio environments.

The Construct API is part of the larger Voisus Server API. Visit the Voisus Server API Documentation[1] first for an overview of the API including information on REST, HTTP, and JSON.

Developers have access to all Construct functionality using the HTTP API:

- Create entities representing vehicles or human characters
- Attach to entities on the simulation network
- Trigger voice transmissions on simulated radios
- Script sequences of radio calls between entities
- Add realistic sound effects to radios
- Position entity speech in 3D game environments
- Create face-to-face conversations between game avatars and human players
- Synthesize variable speech using text-to-speech (TTS)
- Add automated speech recognition and intelligent behaviors to entities

1 Application Examples

- To create a radio interaction between two Entities, create two Entity resources and an Interaction resource that specifies the speech for each radio transmission.
- Trigger dynamic, on-the-fly radio calls from Entities by creating new Utterance resources.
- Add jet engine noise to radio calls from aircraft by setting the `background` attribute on an Utterance.
- Receive speech recognition events from Entities using the streaming status feed.

2 Scenarios

Most Construct resources are contained within a Voisus Scenario. This includes resources for Entities, Interactions, Sounds, Radio Effects, and Language Models. When the Scenario runs, Entities are spawned and Interactions start running. Scenarios themselves are also created, deleted, and run using the API. Consult the Voisus Server API Documentation[2] for more information on scenario management.

[1] voisus_server_api.pdf
[2] voisus_server_api.pdf

2.1 Entities

api/scenarios/scenario_id/entities/

Construct Entities are used to add speech and radio capabilities to objects in simulation environments. These Entities may correspond and be synchronized with Entities in external simulation systems, or they may modeled solely by this Voisus Server.

Entities are created using either the web interface or this HTTP API. The Entity resource shown below is stored in a Voisus Scenario and will cause a runtime Entity to be created and destroyed each time the Scenario is installed and uninstalled, respectively. A separate section of the HTTP API is available for interacting with the runtime representation of an Entity, including retrieving speech recognition results and triggering text-to-speech transmissions.

When creating an Entity in a Scenario, there are a number of optional parameters and only a small number of required parameters. Some parameters are links to other Scenario resources, including `net`, `domain`, `language_model`, and `radio_effects`. In these cases, the parameter value is the URI of the referenced resource.

To create an Entity with basic radio functionality, the `net` and `domain` properties must be set. A TTS voice will be chosen randomly if one is not assigned.

The JSON below represents an Entity named "Broadsword 11".

Variable	Definition
`domain`	Link to a Radio domain resource (controls DIS exercise)
`net`	Link to Net resource for to enable radio communication
`behavior`	Behavior that will control the entity (optional)
`language_model`	Language model for speech recognition (optional)
`radio_effects`	Link to a Radio Effect resource (optional)
`aiml`	Name of Artificial Intelligence Markup Language (AIML) file to use (optional)
`parser`	Name of speech parser to use for Natural Language Understanding (NLU)
`attributes`	Arbitrary attributes for use in speech and behaviors
`listen`	Enables/disables listening with speech recognition
`radio_enabled`	Enables/disables radio communications
`earshot_enabled`	Enables/disables face-to-face communication in 3D environments
`marking`	String used for DIS entity-attach (optional)
`pitch_shift`	Pitch shift amount; 0.9 low, 1 normal, 1.1 high
`tts_voice`	Text-to-speech (TTS) voice
`tts_rate`	Speech rate (1–9)
`tts_volume`	Speech volume (1–9)
`breakpoints`	Breakpoints for inspecting Entity behavior execution
`default_position`	Initial (X,Y,Z) position of the Entity in geocentric coordinates

2.2 Interactions

api/scenarios/<scenario_id>/interactions/

Interactions cause one or more Entities to speak. Each Interaction specifies a list of speech actions to execute, one after the other. As an example, a "IED Attack" interaction might involve a series of radio calls between the attacked vehicle and command personnel.

The JSON below represents an interaction named "Blackjack Under Attack" with two TTS radio calls from different Entities. Since `enabled` is true and `time` is 0, the interaction will start immediately. There will be a three second delay between radio calls because `action_delay` is 3.

Variable	Definition
`action_delay`	Default delay between actions (seconds)
`enabled`	Enables/disables the interaction
`variables`	Arbitrary variables; used during speech synthesis
`holdoff`	Delay between loops (seconds)
`limit`	Number of times the interaction should run (0=forever)
`time`	Interaction start time (seconds)
`time_type`	Enumeration controlling the start time ["rel", "utc"]
`vbs2_trigger`	Optional VBS scripting expression to trigger this interaction
`actions`	List of actions to be executed (see table below)

The `time_type` property determines the meaning of the specified start `time`. A relative time type of "rel" means the `time` is relative to when the scenario begins or when the Interaction is created. The UTC time type of "utc" means the `time` is a wall clock time, specified in seconds since the Unix epoch. If an interaction is created with a UTC time in the past, the interaction will not execute. The Voisus server system clock is used when determining when an Interaction should begin. Confirm this clock is synchronized with external simulation system clocks to ensure proper time-alignment of simulation events.

Actions

Each action in an Interaction has the following format. The action `type` defines whether TTS or prerecorded speech will be used. In either case, the `background` parameter can be used to add environmental ambiance or sound effects like gunfire into the speech stream.

Variable	Definition
`entity`	Link to an Entity resource (required)
`type`	Starting offset in sound file (samples)
`predelay`	Delay before this action executes (seconds)
`text`	Action type enumeration ["tts", "sound"]
`sound`	Link to a Sound resource
`background`	Link to a Sound resource to play in the background

If a `type` of "tts" is specified, the `sound` property is ignored. If `text` is specified for a "sound" action, the text is understood to be a transcript of the sound file, and is shown as such on the Construct status webpage.

2.3 Sounds

api/scenarios/<scenario_id>/sounds/

Sound resources are used to play a sound effect or recorded speech in Construct. Each Sound selects a sound file by name and a segment within the file to play. Sounds may be reused across multiple Entities and Interactions. Sounds are separate from, but reference, sound files. Upload sound files to reference using the web interface.

The JSON below represents a looping engine sound named "Diesel Engine". The `play_count` value of 0 means the sound should loop indefinitely.

Variable	Definition
`file`	Sound file name (e.g. "explosion.wav")
`offset`	Starting offset in sound file (samples)
`length`	Length of section to play; 0=all (samples)
`gain`	Playback gain; 1.0 means no volume adjustment
`play_count`	Number of times to play when triggered (0=forever)
`play_all`	If true, Sound will always play to the end before stopping
`random`	If true, each playback begins at a random position
`loop_start`	Start position of loop [0–1] (0=beginning, 1=end)
`loop_end`	End position of loop [0–1] (0=beginning, 1=end)
`loop_delay`	Delay between loops (seconds)
`transcript`	Optional text transcript for a sound file containing speech. Transcript is shown when an Entity "speaks" using this Sound.

In many cases setting only the `file` property and using the defaults for the other properties will yield the expected results. The Sound defaults to playing the entire sound file once at normal volume.

2.4 Radio Effects

api/scenarios/<scenario_id>/radio_effects/

Radio Effects customize the sound of Entity radio transmissions by adding distortion, noise, and band-limiting audio effects. Link a Radio Effect to one or more Entities by setting the `radio_effects` property in an Entity to the Radio Effect "self" URI.

The JSON below represents a Radio Effect named "High Distortion".

2.5 Language Models

api/scenarios/<scenario_id>/language_models/

Variable	Definition
`distortion_mode`	Switches between distortion, overdrive, and none (0=none, 1=overdrive, 2=distortion)
`distortion_mix`	Adjusts the ratio of distorted to clean signal to be transmitted (0=clean only, 1=distorted only)
`distortion_gain`	Gain applied to the signal before entering the distortion effect. Higher gains yield a more distorted sound.
`limiter_threshold`	Threshold in decibels (dB) that limits the peak signal strength. More negative values yield quieter transmissions.
`noise_gain`	Gain controlling the volume of noise added to the transmission
`gain`	Gain applied to the signal just before transmission
`noise_color`	Type of added noise: "white", "pink", or "brown"
`hp_freq`	Highpass filter frequency in Hertz (Hz)
`lp_freq`	Lowpass filter frequency in Hertz (Hz)

Language Model resources select a speech recognition model and specify custom parameters for the recognition engine. To use a Language Model, set the `language_model` property of an Entity to the Language Model "self" URI.

Statistical language models or grammars must be pre-installed on the Voisus server for Speech Recognition resources to be functional. Installed models are referenced by name in this resource. In the example below, "atc11" is the name of a model provided by ASTi. Currently installed models are listed in the model dropdown on the Language Models webpage. Contact ASTi for more information on available language models.

The JSON below represents an ASR configuration named "ATC-ASR".

Variable	Definition
`asr_model`	Name of the language model on disk
`grammar`	Grammar text in BNF format
`parameters`	Newline-separated low-level parameters for the recognition engine
`root`	Name of the grammar rule to enable at start
`shared`	Boolean controlling whether a single instance of this language model should be shared across all entities to conserve memory

When `asr_model` points to a statistical language model, the `grammar` and `root` variables are ignored.

Shared language models have one important limitation: only one entity may use it to process speech at a time. If another entity starts listening at the same time with the same shared Language

Model, the speech it hears will not be transcribed with speech recognition.

3 Construct Settings

api/construct/settings/

These settings affect Construct behavior overall and are not Scenario specific. The settings resource can only be modified (it cannot be created or deleted). Issue a HTTP PUT request to the settings URI to update settings.

Variable	Definition
`log_audio`	Boolean controlling whether speech recognition audio is logged
`log_events`	Boolean controlling whether speech events are logged
`dm`	Optional network settings for a Discovery Machine behavior engine
`vbs2`	Optional settings for interfacing with a VBS instance
`tts_substitutions`	Optional text transformations for use during TTS

Frequently these settings are easily managed through the web interface, but they are also available in the API for dynamic modification from external systems.

We recommend enabling both `log_audio` and `log_events` in order to capture the complete history of events on this Construct instance for later review and tuning.

4 Natural Language Parsers

api/construct/parsers/

This collection resource lists all available Natural Language Parsers installed on the Voisus Server. Parser plugins provided by ASTi will be shown here. This list may be empty if no plugins are installed.

This resource is read-only and is meant to support the selection of a parser in a Construct Entity. The `parser` Entity attribute should be set to the `name` value of a parser in this collection.

5 Artificial Intelligence Markup Language (AIML)

api/construct/aiml/

AIML describes voice interactions and natural language processing capabilities for a Construct Entity. AIML resources created here are referenced by ID with an Entity's `aiml` attribute. The `text` attribute of the AIML resource should contain the complete XML specification of the AIML behavior.

6 Runtime Entities

api/construct/entities/

Variable	Definition
`name`	Name of the AIML definition
`text`	XML formatted AIML specification

Construct Entities that are actively running exist here in the runtime Entity API. Runtime Entities are separate from, but related to, Entity resources defined in the Scenario. The key distinction is that a Scenario and the Entity resources within can be edited without the Scenario currently running. When a Scenario runs, its Entities are created and become accessible through this API endpoint. Alternatively, if an application calls for Entities that exist only temporarily, and should not be saved in the Scenario, this endpoint is used to manage those Entities directly.

Runtime entities are synchronized with their corresponding Scenario resource, if one exists.

The attributes shown below track the current runtime Entity state and are updated dynamically as the system runs. For example, the `speaking` value updates when the Entity speaking state changes.

This runtime Entity state is also available via the Construct status streaming API endpoint to eliminate the need to repeatedly poll for this information. See below for documentation on the streaming endpoint.

Note that references to other resources are in the form of IDs or names here instead of by URI like in the Scenario resources.

7 Utterances

api/construct/utterances/

An utterance is a single speech event for a Construct Entity. Entity speech can be triggered indirectly from Interactions and Behaviors, or directly from a remote system via this Utterances API endpoint.

This endpoint is only useful when a Scenario is actively running.

An HTTP client may POST to /api/construct/utterances/to create a new utterance. The client must provide a JSON object with an `entity` parameter specifying the ID of the entity that should speak. Either `text` or `sound` should be provided, depending on whether the client wishes to use TTS or a prerecorded speech file, respectively. The server response includes the `self` URI which can be polled for status, including the `error` and `complete` flags.

8 Streaming Status and Events

api/construct/status/

The Construct status endpoint is useful when a client wishes to receive continuous state updates and event notifications from Construct Entities. An HTTP GET request on this API endpoint will block until the running Scenario is stopped. While the Scenario is running, JSON messages, separated by newline characters, will be sent from the server. Due to the blocking behavior of this endpoint, clients may wish to spawn a separate thread to issue this request in order to not block the client application's main thread.

When a Scenario is running and a client makes a request, a connected message will be sent:

Variable	Definition
id	Entity ID
domain	Name of the Entity's selected DIS exercise
net	Net ID
net_name	Net name from the Commplan
language_model	Language Model ID
language_model_name	Language Model name
behavior	Behavior ID
behavior_name	Behavior name
behavior_state	Enumeration: ['inactive', 'active', 'error', 'finished']
behavior_node	Last executed behavior node
blackboard	URI of Entity blackboard containing current state variables
aiml	AIML definition ID
aiml_name	AIML name
parser	Selected natural language parser name
state	Enumeration: ['initializing', 'active']
listen	true if the Entity is actively listening for speech
speaking	true if the Entity is currently speaking
speech_state	Enumeration: ['idle', 'waiting', 'ptt', 'speaking', 'finished']
speech_text	Text of the current utterance, if there is one
asr_state	Enumeration: ['inactive', 'initializing', 'active', 'error']
radio_enabled	Radio enabled/disabled state
earshot_enabled	Earshot enabled/disabled state
pitch_shift	Voice pitch shift value
tts_voice	Selected TTS voice
tts_rate	Current TTS speech rate (1–9)
tts_volume	Current TTS speech volume (1–9)
marking	Marking field string used to link with an Entity on the network
attributes	Arbitrary key:value pairs for use in behaviors and speech
position	Current geocentric position
default_position	Initial geocentric position
breakpoints	List of currently set behavior breakpoints
breakpoint	Active breakpoint information
radio_state	tx_active and rx_active state for the radio
earshot_state	tx_active and rx_active state for 3D communications

Variable	Definition
`id`	Utterance ID
`entity`	URI or ID of the Entity that should speak
`text`	Speech text for TTS
`sound`	Sound ID for prerecorded speech playback
`background`	Sound ID for a background sound during the tranmission
`complete`	Status returned from the server; true when utterance speech has finished
`error`	Error status returned from the server

All messages from the server will include a `type` which is either "connected", "update", "delete", or "event" depending on the situation. Clients should ignore received messages that have an unknown type.

8.1 Entity Update

An update message notifies the client that an Entity was created or updated.

8.2 Entity Delete

A delete message notifies the client that an Entity was deleted.

8.3 Speech Event

The speech event type notifies the client that an Entity has started speaking.

8.4 Speech-Finished Event

The speech-finished event type notifies the client that an Entity has stopped speaking.

8.5 Speech Recognition Event

The message event type notifies the client of a speech recognition event.

Variable	Definition
`text`	Recognized speech text after optional substitutions
`original_text`	Raw recognized speech from the recognition engine
`confidence`	Speech recognition confidence level (0–100)
`recording`	URI for recording of recognized speech
`meaning`	Extracted speech meaning from parser or AIML
`net_id`	ID of radio net that received the speech
`net_name`	Name of the radio net
`max_sample`	Peak sample value in the received speech
`voice`	true if this was a voice message (as opposed to a text message)
`time`	Unix timestamp when speech was recognized