Strive SAMMI documentation

By: Dan (TheLettuceClub)

You must install the Strive x SAMMI extension for this to work. Instructions [here](https://github.com/Krackatoa/Strive-x-Sammi).

For general instructions on how to get webhook data into SAMMI buttons, please see Krackatoa’s fine [guide for GGXrd](https://docs.google.com/document/d/1fFRs7567tCdn9gGu_4iLQM8tAB07vP8aDQeLaWrbqUg/edit). This document will only go over the various events my mod will send and what data they contain.

For non-SAMMI connections, connect to “ws://localhost:6615”, the mod will send message to you in the same format as defined below.

Important note: In order to prevent cheating, a universal 20 frame delay has been added to the mod. You can compensate for this in OBS by adding a render delay to your game capture and audio sources. We appreciate your understanding.

State update:

Abstract: collects the state of the game and players, runs every frame.

Trigger name: ggst\_stateUpdate

Fields: (indentation implies a “.”, so the tree looks like webhook.data.eventInfo.<>)

* webhook (incoming variable)
  + data
    - eventInfo
      * p1/p2 (both exist and have identical fields)
        + action: the current BBScript state name
        + health: current player health as number from 0 to 420
        + burst: amount of burst gauge, 0-1000
        + risc: amount of RISC gauge, 0-12800
        + tension: amount of tension/meter, 0-10000
        + dangerBalance: value that indicates how close a player is to triggering danger state, unknown values/thresholds
        + charaName: 3 letter name of the character
        + invuln

strikeInvuln: Boolean number, 1 is yes, 0 is no

throwInvuln: same but for throws

absInvuln: same but for everything

* + - * + inputs:

up, down, left, right, p, k, s, h, d, taunt, dash, RC, replay, record, reset, FD: all are numerical Booleans like cancels. 1=on, 0=off.

* + - * + tensionPulse: each player’s tension pulse
        + direction: which direction the player is facing. 0=left, 1=right
      * roundTimeLeft: time left in the given round
      * roundTimeLimit: total initial time in the round
      * roundCount: current round of the game (1+)
      * roundMax: max number of rounds in the game (always two more than the number of hearts on screen)
      * frameCount: the number of frames since the beginning of the round

Round Start:

Abstract: fires once on the beginning of any round/game

Trigger name: ggst\_roundStartEvent

Fields:

* webhook.data.eventInfo
  + p1Character: 3-char abbreviation of character name (broken)
  + p1OnlineName: p1’s online name, if present
  + p2Character: 3-char abbreviation of character name (broken)
  + p2OnlineName: p2’s online name, if present

Round End:

Abstract: fires once when the round ends

Trigger name: ggst\_roundEndEvent

Fields:

* webhook.data.eventInfo
  + type: type of round end. One of “Player1”, “Player2”, or “Draw”
  + cause: what caused the round end, either “TimeOut”, or “KO”
  + winnerName: uses the online name if present, empty if local
  + loserName: same as winnerName
  + frameCount: the number of frames since the beginning of the round

Hit Event:

Abstract: fires whenever either player lands a hit (only on block or hit, whiffs don’t count)

Trigger name: ggst\_hitEvent

Note: the damage related fields don’t update for grabs or supers that deal damage without using hitboxes.

Fields:

* webhook.data.eventInfo
  + hitType: the type of hit performed: normal, special or overdrive
  + chSize: the size of the counterhit, if there is one. One of: normal, small, medium, large, or RISC
    - NOTE: this only works with real counterhits, simply turning on CH in training mode isn’t enough unfortunately
  + guardType: how the attack can be blocked. One of: all, high, low, or none
  + attackLevel: the level of the attack, 1-5
  + damage: the damage the attack dealt to the other player. This is the scaled damage that should corroborate with the defender’s health difference.
  + comboCount: the current combo count
  + comboDamage: the total (scaled) damage this combo has done.
  + chipDamage: damage done on block, will be filled in on hit or block.
  + attacker: who is doing the attacking, either “Player1” or “Player2”
  + defender: who is getting hit, same values
  + attackerAction: the BBScript state name of the attack that’s happening
  + attackerChara: 3-character abbreviation of the character name
  + defenderAction: the BBS state of the hitstun
  + defenderPrevAction: the BBScript state of the defender before the current one
  + defenderChara: 3-character abbreviation of the character name
  + wallHealth: the current health of the wall. Decreasing number where 0=wallbreak (slightly broken)
  + cancels:
    - gatlingCancel, sequenceCancel, specialCancel, hitSpecialCancel, jumpCancel, hitJumpCancel, dashCancel, hitDashCancel, airDashCancel: all are the same Boolean numbers as the invuln types
  + priority: the internal “priority” value of the attack. Ranges from 1 to at least 10
  + frameCount: the number of frames since the beginning of the round

Object Created Event:

Abstract: fired when a new object like a projectile is created. Can have null values under some circumstances.

Trigger name: ggst\_objectCreatedEvent

Fields:

* webhook.data.eventinfo
  + objName: the name of the object just created
  + parent: the parent of this object. One of “player1”, “player2”, or “projectile”
  + p1/p2Action: the BBS state of both players at the time
  + sprite: what sprite the object is displaying
  + frameCount: the number of frames since the beginning of the round

Menu Timeout:

Abstract: fires whenever the game goes to the main menu, to detect mid-match quits

Trigger name: ggst\_Timeout

No Fields. This will fire a lot, basically whenever the game switches “scenes”.