STEAL Specification
Version 6.0

Dennis Peters,

Jonathan Anderson, Lihong Zhang, Pouria Shaker, Andrew Vardy

Fall 2009

2009-09-10(#894)

1 Overview

The STEAL (Simulator - TEAm Link) protocol is the means by which CTF controllers and simulators communicate. This document outlines the STEAL protocol's grammar and semantics[1].

2 Grammar

The STEAL grammar is given here as a context free grammar. In this grammar:

- $request(x)$ is a request from a controller to the server (ends with newline)

- $response(x)$ is a response from the server to a controller (ends with newline)

- `monospace` font indicates a string literal (ASCII-encoded)

- Items in braces - $\{\textbf{Something}\}_x^y$ - are repeated x to y times

- Symbol classes in square brackets can be replaced by any single character in the indicated class (e.g. $[A - Za - z]$ can be replaced by any upper- or lower-case letter)

- A pipe character (|) on the right side of a production is a logical OR (e.g. $\textbf{X} \rightarrow a \mid b$ indicates that $\textbf{X}$ may produce a or b)

- **Space** is an ASCII space character (32_{10})

The STEAL grammar is given in Figure 1.

---

[1]For information on the CTF game itself (e.g. allowable flag placement, the meaning of TCTF), consult the CTF specification.

$$
\begin{aligned}
\textbf{Start} &\rightarrow \textit{request}(\texttt{CTF2009}\ \textbf{Space Name})\ \textit{response}(\texttt{CTF2009})\ \textbf{Version} \\
\textbf{Version} &\rightarrow \textit{request}(\texttt{STEAL}\ \textbf{Space Float})\ \textit{response}(\texttt{STEAL}\ \textbf{Space Float})\ \textbf{RequestSide} \\
\textbf{RequestSide} &\rightarrow \textit{request}(\texttt{requestSide}\ \textbf{Space Int})\ \textit{response}(\texttt{onSide}\ \textbf{Space Int})\ \textbf{NewRound} \\
\textbf{NewRound} &\rightarrow \textit{request}(\texttt{getTrees})\ \textit{response}(\textbf{Trees})\ \textbf{NewRound} \\
&\quad |\ \textbf{PlaceFlag} \\
\textbf{Trees} &\rightarrow \texttt{trees}:\textbf{Space}\ \{\textbf{Point Space}\}^{20} \\
\textbf{PlaceFlag} &\rightarrow \textit{request}(\texttt{placeFlag}\ \textbf{Space Point})\ \textit{response}(\texttt{ok})\ \textbf{Play} \\
\textbf{Play} &\rightarrow \textit{request}(\texttt{getState})\ \textit{response}(\textbf{State})\ \textbf{Play} \\
&\quad |\ \textit{request}(\texttt{getState})\ \textit{response}(\textbf{RoundOver})\ \textbf{NewRound} \\
&\quad |\ \textit{request}(\texttt{getState})\ \textit{response}(\textbf{GameOver}) \\
&\quad |\ \textit{request}(\textbf{Command})\ \textit{response}(\textbf{CommandResponse})\ \textbf{Play} \\
&\quad |\ \textit{request}(\textbf{Command})\ \textit{response}(\textbf{RoundOver})\ \textbf{NewRound} \\
&\quad |\ \textit{request}(\textbf{Command})\ \textit{response}(\textbf{GameOver}) \\
\textbf{Command} &\rightarrow \texttt{spin}\ \textbf{Space Int Space Int Space Float} \\
&\quad |\ \texttt{accelerate}\ \textbf{Space Int Space Int Space Float} \\
&\quad |\ \texttt{placePlayer}\ \textbf{Space Int Space Point} \\
&\quad |\ \texttt{setTime}\ \textbf{Space Int} \\
\textbf{State} &\rightarrow \texttt{state}:\textbf{Space Int}\ \{\textbf{TeamState}\}^{2} \\
\textbf{TeamState} &\rightarrow \textbf{Space Name Space Int FlagState}\ \{\textbf{PlayerState}\}^{8} \\
\textbf{FlagState} &\rightarrow \textbf{Space}\ (\textbf{Int Space Point}) \\
\textbf{PlayerState} &\rightarrow \textbf{Space}\ (\textbf{Int Space Point Space Float Space Float}) \\
\textbf{CommandResponse} &\rightarrow \texttt{time}:\textbf{Space Int} \\
\textbf{RoundOver} &\rightarrow \texttt{roundOver}:\textbf{Space Int Space Int} \\
\textbf{GameOver} &\rightarrow \texttt{gameOver}:\textbf{Space Int} \\
\textbf{Error} &\rightarrow \texttt{error}:\textbf{Space}\ \{[A-Za-z0-9\textbf{Space}]\}^{255}_{1} \\
\textbf{Point} &\rightarrow (\textbf{Float},\textbf{Float}) \\
\textbf{Float} &\rightarrow \textbf{Int.Int}\ |\ -\textbf{Int.Int} \\
\textbf{Int} &\rightarrow \{\textbf{Digit}\}^{9}_{1} \\
\textbf{Name} &\rightarrow \texttt{red}\ |\ \texttt{blue}\ |\ \texttt{green}\ |\ \texttt{purple}\ |\ \texttt{orange}\ |\ \texttt{black} \\
\textbf{Digit} &\rightarrow [0-9] \\
\textbf{Space} &\rightarrow 32_{10}
\end{aligned}
$$

Figure 1: STEAL Grammar

This grammar may also be visualized as a Nondeterministic Finite State Automaton (NFSA), as shown in Figure 2.
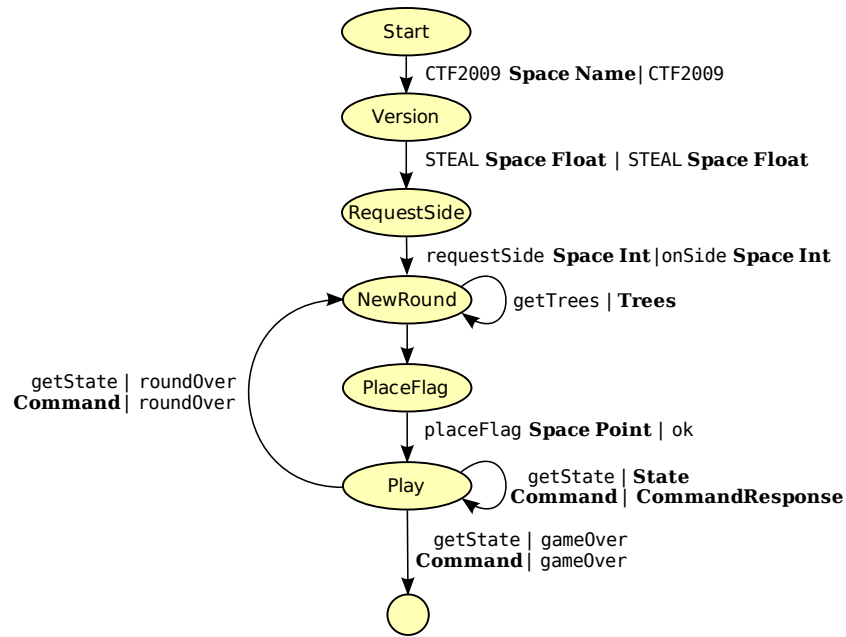
Figure 2: STEAL as an NFSA

In this case, the pipe character | is used, not as a logical OR, but as a separator between input and output.

3   Semantics

3.1   Requests

3.1.1   CTF2009

Initial handshake.
Parameter: the colour of the connected controller.

3.1.2   STEAL

Informs the simulator of the STEAL protocol version the controller is using.
Parameter: the STEAL version the client supports

3.1.3   requestSide

Requests a particular side of the field for home territory.
Parameter: the side being requested (0: left, 1: right)

3.1.4   getTrees

Requests the location of all trees on the field.

3.1.5 placeFlag

Place the team's flag.
   Parameter: where to place the flag

3.1.6 getState

Requests the state of all players and flags.

3.1.7 spin

Set a player's angular velocity.
   Parameters:

1. Player number

2. Time (in ms) the command should take effect (non-negative, 0 means "immediately")

3. New angular velocity (in degrees/s)

3.1.8 accelerate

Set a player's linear acceleration.
   Parameters:

1. Player number

2. Time (in ms) the command should take effect (non-negative, 0 means "immediately")

3. New acceleration (in m/s^2)

3.1.9 placePlayer

Place a player on the field (**test mode only**). Parameters:

1. Player number

2. Location (in m)

3.1.10 setTime

Set the game time (**test mode only**).
   Player movement is undefined: players may remain where they were before the time changed, or they may move according to their current speed and orientation.
   State changes are also undefined: collisions may or may not be detected if the time advance is large enough that players can "jump over" other objects.
   Parameters:

1. New game time (in ms)

3.2   Responses

3.2.1   CTF2009

Initial handshaking response.

3.2.2   STEAL

The version of the STEAL protocol that the simulator supports.

3.2.3   onSide

The side of the field assigned to the controller.
   Parameter: the side assigned (0: left, 1: right)

3.2.4   Trees

The location of all the trees on the field.
   Parameters: Each parameter is the location of a single tree
   If a tree has not been placed (e.g. one team has not placed their trees), its position shall be reported as (0.0, 0.0). Since (0,0) is not a legal tree placement, controllers shall understand (0,0) to mean that the tree has not been placed.

3.2.5   StateResponse

The current state of the game.
   Parameters:

1. The initial **Int** is the current game time (in ms)

   (a) If the game has not started yet (e.g. one team has not placed its trees or flag), this value shall be 0

2. The state of each team follows

3.2.6   TeamState

The state of a team.
   Parameters:

1. The name of the team

2. The team's score

3. The state of the flag (according to **FlagState**)

4. The state of the team's players (according to **PlayerState**)

3.2.7 FlagState

The state of a flag, enclosed in parentheses.
 Parameters:

1. The flag's visibility (0 for invisible, 1 for visible)

2. The flag's location - (0.0,0.0) if invisible

3.2.8 PlayerState

The state of a player, enclosed in parentheses.
 Parameters:

1. State (meanings given in Table 1)

2. Location (in m)

3. Speed (in m/s)

4. Orientation (in degrees)

Value	Meaning
0	Invisible
1	Defending
2	Attacking
3	Prisoner
4	Escapee
5	Flag Carrier
6	TCTF

Table 1: Player States

When players are reported to be Invisible, they shall be reported to have a location of (0,0) m, a speed of 0 m/s and an orientation of 0 degrees.

3.2.9 CommandResponse

A command has been received and determined to be valid.
 Parameters:

1. The current game time (in ms)

3.2.10 RoundOver

The current round of play is over (a flag, or all the players on a team, have been captured).
 Parameters:

1. The reason the round ended (0: flag captured, 1: team captured)

2. The team which did the capturing (1: the receiving controller, 0: the other controller)

3.2.11   GameOver

The game is over.

  Parameters:

1. Which team won the game (0: the receiving controller lost, 1: the receiving controller won, 2: the game was a tie)

3.2.12   Error

The Error response is sent to a simulator or controller that sends an erroneous message. Example of erroneous messages include messages which violate the STEAL grammar and messages that ask the simulator to do inappropriate things (e.g. place a flag on the wrong side of the field).