Raven API

User Guide UG221122

Version 1.3

Protocol Version 0xFFFEFFFE

CONTENTS

Introduction

The RavenAPI is used to control Iris Dynamics 6DoF motion platforms from an application like a simulator or game. The application initiates RavenAPI messages by sending a UDP datagram containing a frame of simulator data, or a behavior command. The platform responds to every valid package it receives according to the package contents. When powered, the platform is always listening for messages, no configuration is required.

Messages begin with the protocol version identifier, followed by the message ID. The following data is context-specific. Finally, all messages conclude with an 8-bit CRC calculation.

Multiple messages can be sent in one datagram, and messages can be split between consecutive datagrams.

All numerical representations are in a signed 32-bit format and sent most-significant-byte first.

Background

Connection

API commands can be carried out via UDP datagrams. Typical IP settings are 10.0.0.2. Incoming data (from an application to the platform) uses port 9200, and outgoing data uses port 9201.

Degree of Freedom (DoF)

RavenAPI commands acceleration cueing of 6 degrees of freedom: surge, sway, heave, roll, pitch, and yaw.

DoF	Description
Surge	A forward-backward motion, such as a car accelerating. Positive surge motion is forward motion.
Sway	A side-to-side motion such as a car cornering fast (not to be confused with yaw).
Heave	An up-down motion, such as a helicopter lifting off.
Roll	A rotation about the surge axis, such as a barrel roll.
Pitch	A rotation about the sway axis, such as a nose dive.
Yaw	A rotation about the heave axis, such as spinning on an office chair.

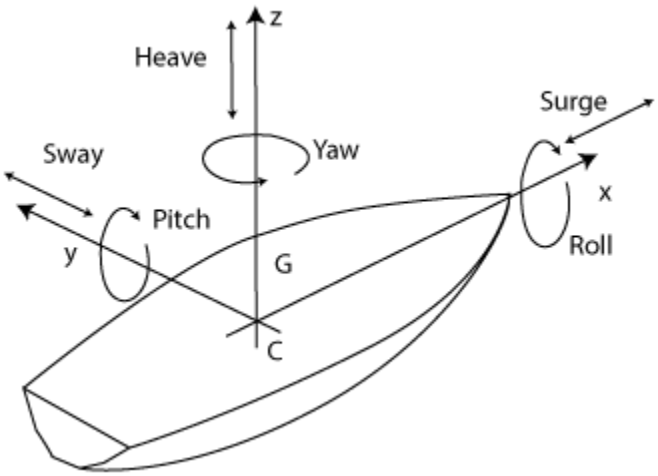

Image 1: Degrees of Freedom diagram

Frames of Reference (FoR)

The world FoR refers to a virtual fixed space.

The avatar FoR is that experienced by the simulated craft in the virtual world. It is with respect to this FoR that applications package and send data to RavenAPI.

The rider FoR is that experienced by the physical rider.

The platform FoR is with respect to the platform base. It is with respect to this FoR that the platform's position and orientation are reported from RavenAPI.

Anticipated Format

The Raven controller expects acceleration data passed to it to be in the avatar's FoR. This means that depending on how the application calculates accelerations, middleware or a plugin may be required to rotate the accelerations appropriately.

Types of Platform Control

Acceleration Cueing

The avatar's accelerations in the virtual environment are measured and reported to the platform which makes an effort to replicate those accelerations for the physical rider. Accordingly, the accelerations provided by the application to the platform should be in the avatar FoR.

The response of the platform to indicated avatar accelerations is controlled by the Raven' kinematic and washout controller.

Washout Positions

Washout positions are surge, sway, and heave translations, and roll, pitch, and yaw rotations that the platform will tend toward and settle to. The washout positions are requested with respect to the platform FoR.

These positions are measured from the center resting position when all actuators are at half of their maximum extension.

The path used to return a DoF to the washout positions is a function of the DoF's settling time setting, and the incoming accelerations.

Using non-zero settling positions for surge and sway can result in excess power consumption and actuator temperatures and is typically not recommended.

Modes of Operation Control

There are four modes of operation: Off, Level Brake, Loading and Cueing.

Off	The position controller is disengaged. Actuators that are free of errors exert damping forces. This is the default state when the GUI is disconnected and the actuators are not initialized or have errors.
Level Brake	The platform attempts to stay level while allowing itself to raise or lower depending on the net external load. This is the default state when the GUI is not connected and the actuators are initialized and error free.
Loading	Each DoF position can be commanded by the GUI or the RavenAPI by setting the washout targets. Accelerations from RavenAPI are ignored.
Cueing	Accelerations from RavenAPI are scaled according to the gain settings and reproduced by the platform.

Modes of operation can be controlled with a “Mode change request” type message.

Gravity

Forces due to gravity must be accounted for when cueing motions for the rider. By using different message types, the application can add the force of gravity to the accelerations of the avatar, or instead pass a value for the acceleration due to gravity.
When adding the force of gravity prior to the accelerations, an application should project the force of gravity onto the user's surge, sway, and heave accelerations according to the avatar's pitch and roll orientation.

Message Construction

Framing Data

Message Framing – Application to Platform

	Byte							
	1	2	3	4	5	6	...	N
From app	Version ID				Message ID		Payload	CRC

Message Framing – Platform to Application

	Byte							
	1	2	3	4	5	6	...	N
To app	Version ID				Message ID		Payload	CRC

Units

Data Format

Name	DoFs	Format	Unit	Maximums
Positions	Surge, Sway, Heave	2s compliment signed 32-bit	micrometers (um)	+- 2,000,000
Position Speeds			millimeters per second (mm/s)	+- 4,000
Position Accelerations			millimeters per second squared (mm/s^2)	+- 60,000
Rotations	Roll, Pitch, Yaw		milldegrees (mdeg)	+- 360,000
Rotation Speeds			degrees per second (deg/s)	+- 250
Rotation Acceleration			degrees per second squared (deg/s^2)	+- 4,000

Vocabulary

Application to Platform

Application to Platform Messages

Message ID	Payload Size (Words / Bytes)	Total Package Length (Bytes)	Description
5	6 / 24	31	**Acceleration only frame** transmit the accelerations of each dof (x6) from the avatar FoR. A coordinated turn in an aircraft should result in zero sway acceleration.
21	8 / 32	39	**Acceleration and attitude frame** transmit the accelerations of each dof (x6), and the attitude of the rider (roll and pitch) from the avatar FoR. Gravity (9.8 m/ss) is projected onto the surge, sway, and heave accelerations according to the roll and pitch angle.
170	6 / 24	31	**Washout targets frame** transmit the washout position of each dof (in the platform FoR). This frame can be used to implement offsets for dof positions during cueing depending on specific simulator behaviors. Also provides a way to move the platform in loading mode.
682	1 / 4	11	**Mode change request frame** request to change to a new mode of operation.
10922	0 / 0	7	**Actuator Temperature Request Frame** Requests current actuator temperature data.
10923	0 / 0	7	**Actuator + Platform Power Request Frame** Requests the power draw of all motors
65532	0 / 0	7	**All Actuators Position Request Frame** Requests current position for all actuators
65533	0 / 0	7	**All Actuators Commanded Force Request Frame** Requests current measured force for all actuators.

65534	0 / 0	7	**All Actuators Target Position Request Frame** Requests current target position for all actuators.
65535	0 / 0	7	**All DoFs Position Request Frame** Requests current position/angle for all DoFs.
3780	0 / 0	7	**Firmware Release Request Frame** Requests current Super Eagle firmware information

Application to Platform Payloads

Message ID	Words (32-bits)								
	1	2	3	4	5	6	7	8	9
5	Surge acc	Sway acc	Heave acc	Roll acc	Pitch acc	Yaw acc			
21	Surge acc	Sway acc	Heave acc	Roll acc	Pitch acc	Yaw acc	Roll angle	Pitch angle	
85	Surge acc	Sway acc	Heave acc	Roll acc	Pitch acc	Yaw acc	Roll angle	Pitch angle	Gravity
170	Surge pos	Sway pos	Heave pos	Roll ang	Pitch ang	Yaw ang			
682	Mode request								

Platform to Application

Platform to Application responses

Response ID	Payload Size (Words / Bytes)	Total Package Bytes	Description
5, 21, 85, 170, 65531	7 / 28	35	**Platform status frame** Returns the position and orientation of the platform with respect to the platform FoR, and a status word containing flags that describe the platform state.
682	1 / 4	11	**Mode change response frame** Returns the status word, which contains the mode of operation.
10922	7 / 28	35	**Temperatures request response frame** Returns the temperature of each actuator, and a status word containing flags that describe the platform state.
10923	8 / 32	39	**Power request response frame** Returns the power burn of each actuator (words 1-6), as well as the sum of all power draw (word 7), and a status word containing flags that describe the platform state.
65535	8 / 32	39	**Platform DoF request response frame** Returns the position or angle(um or millidegrees) of each DoF, a timestamp, and a status word containing flags that describe the platform state.
65534	8 / 32	39	**Target positions request response frame** Returns the target position (um) of each actuator, a timestamp, and a status word containing flags that describe the platform state.
65533	8 / 32	39	**Actuator force request response frame** Returns the commanded force (-32k to + 32k) of each actuator, a timestamp, and a status word containing flags that describe the platform state.
65532	8 / 32	39	**Actuator positions request response frame** Returns the actual position of each actuator (um), a timestamp, and a status word containing flags that describe the platform state.
3780	8 / 32	39	**Firmware version request frame** Returns the firmware version, release state, revision number, year of release, months of release, day of release, commit hash, and a status word containing flags that describe the platform state.

Platform to Application Payloads

Message ID	Words (32-bits)							
	1	2	3	4	5	6	7	8
5, 21, 85, 170, 65531	Surge pos	Sway pos	Heave pos	Roll ang	Pitch ang	Yaw ang	Status Word	
682	Status Word							
10922	Temp 0 Data	Temp 1 Data	Temp 2 Data	Temp 3 Data	Temp 4 Data	Temp 5 Data	Status Word	
10923	Motor 0 Power in Watts	Motor 1 Power in Watts	Motor 2 Power in Watts	Motor 3 Power in Watts	Motor 4 Power in Watts	Motor 5 Power in Watts	Total Power in Watts	Status Word
65535	Surge pos	Sway pos	Heave pos	Roll ang	Pitch ang	Yaw ang	Timestamp	Status Word
65534	Act 0 Target	Act 1 Target	Act 2 Target	Act 3 Target	Act 4 Target	Act 5 Target	Timestamp	Status Word
65533	Act 0 Force	Act 1 Force	Act 2 Force	Act 3 Force	Act 4 Force	Act 5 Force	Timestamp	Status Word
65532	Act 0 Pos	Act 1 Pos	Act 2 Pos	Act 3 Pos	Act 4 Pos	Act 5 Pos	Timestamp	Status Word
3780	Firmware Version	Release State	Revision Number	Release Year	Release Month	Release Day	Commit Hash	Status Word

Status Word

Status Word Composition

Bit												
Bits 31 - 12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	Motor 0 OK	Motor 1 OK	Motor 2 OK	Motor 3 OK	Motor 4 OK	Motor 5 OK	Thermal Mode		Operational Mode			

Operational Mode

Mode	Value
OFF	0
LEVEL BRAKE	1
LOADING	2
CUEING	3

Thermal Mode

Mode	Value
NORMAL	0
OVERHEAT PROTECTION	1

Actuator OK Status

Status	Value
Not OK (has errors)	0
OK (no errors)	1

CRC Class

The CRC byte of each message can be constructed using the following code:

```
/* The following CRCFast class was adapted from https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculation-C-Code and can be found in Barr, Michael. "Slow and Steady Never Lost the Race," Embedded Systems  Programming, January 2000, pp. 37-46.
Big thank you for this! */
#define CRC_POLYNOMIAL    0xD5
class CRCFast {
  public:
    static uint8_t table [256];
    static uint8_t generate (uint8_t const message[], int nBytes) {
      uint8_t data;
      uint8_t remainder = 0;
      // Divide the message by the polynomial, a byte at a time.
      for (int byte = 0; byte < nBytes; ++byte) {
          data = message[byte] ^ remainder;
          remainder = table[data] ^ (remainder << 8);
      }
      return (remainder); // The final remainder is the CRC.
    }
    static void build_table() {
      uint8_t remainder;
      for (int dividend = 0; dividend < 256; ++dividend){
        remainder = dividend;
        // Perform modulo-2 division, a bit at a time.
        for (uint8_t bit = 8; bit > 0; --bit) {
          if (remainder & 0x80)
            remainder = (remainder << 1) ^ CRC_POLYNOMIAL;
          else
            remainder = (remainder << 1);
         }
         //Store the result into the table.
         table [dividend] = remainder;
      }
    }
};
```

Revision History

Version	Date	Author	Reason
1.0	November, 2022	kc	Initial Draft
1.1	March, 2023	Kc kh	Remove depreciated frame 2730 Add frame 10923
1.2	September, 2023	Kh	Removed references to 'Mantis' (old naming) and removed deprecated frame 85 Updated formatting