# Swift Navigation Binary Protocol

**Protocol Specification 0.49**

## Contents



# 1 Overview

The Swift Navigation Binary Protocol (SBP) is a fast, simple, and minimal binary protocol for communicating with Swift devices. It is the native binary protocol used by the Piksi GPS receiver to transmit solutions, observations, status, and debugging messages, as well as receive messages from the host operating system, such as differential corrections and the almanac. As such, it is an important interface with your Piksi receiver and the primary integration method with other systems.

This document provides a specification of SBP framing and the payload structures of the messages currently used with Swift devices. SBP client libraries in a variety of programming languages are available at http://docs.swiftnav.com/wiki/SwiftNav_Binary_Protocol.

# 2 Message Framing Structure

SBP consists of two pieces:

- an over-the-wire message framing format
- structured payload definitions

As of Version 0.49 , the frame consists of a 6-byte binary header section, a variable-sized payload field, and a 16-bit CRC value. All multibyte values are ordered in **little-endian** format. SBP uses the CCITT CRC16 (XMODEM implementation) for error detection[1].

| Offset (bytes) | Size (bytes) | Name | Description |
|---|---|---|---|
| 0 | 1 | Preamble | Denotes the start of frame transmission. Always 0x55. |
| 1 | 2 | Message Type | Identifies the payload contents. |
| 3 | 2 | Sender | A unique identifier of the sender. On the Piksi, this is set to the 2 least significant bytes of the device serial number. A stream of SBP messages may also included sender IDs for forwarded messages. |
| 5 | 1 | Length | Length (bytes) of the Payload field. |
| 6 | $N$ | Payload | Binary message contents. |
| $N+6$ | 2 | CRC | Cyclic Redundancy Check of the frame's binary data from the Message Type up to the end of Payload (does not include the Preamble). |
| | $N+8$ | | Total Frame Length |

Table 2.0.1: Swift Binary Protocol message structure. $N$ denotes a variable-length size.

Swift devices, such as the Piksi, also support the standard NMEA-0183 protocol for single-point position solutions. Observations transmitted via SBP can also be converted into RINEX. Note that NMEA doesn't define a standardized message string for RTK solutions. To make it possible to achieve RTK accuracy with legacy host hardware or software that can only read NMEA, recent firmware versions implement a "pseudo-absolute" mode.

[1] CCITT 16-bit CRC Implementation uses parameters used by XMODEM, i.e. the polynomial: $x^{16}+x^{12}+x^5+1$. For more details, please see the implementation at https://github.com/swift-nav/libsbp/blob/master/c/src/edc.c#L59. See also *A Painless Guide to CRC Error Detection Algorithms* at http://www.ross.net/crc/download/crc_v3.txt

# 3 Basic Formats and Payload Structure

The binary payload of an SBP message decodes into structured data based on the message type defined in the header. SBP uses several primitive numerical and collection types for defining payload contents.

| Name | Size (bytes) | Description |
|---|---|---|
| s8 | 1 | Signed 8-bit integer |
| s16 | 2 | Signed 16-bit integer |
| s32 | 4 | Signed 32-bit integer |
| s64 | 8 | Signed 64-bit integer |
| u8 | 1 | Unsigned 8-bit integer |
| u16 | 2 | Unsigned 16-bit integer |
| u32 | 4 | Unsigned 32-bit integer |
| u64 | 8 | Unsigned 64-bit integer |
| float | 4 | Single-precision float (IEEE-754) |
| double | 8 | Double-precision float (IEEE-754) |
| array | — | Fixed or variable length array of any fill type |
| string | — | Fixed or variable length string (NULL padded/terminated) |
| bitfield | — | A primitive type, typically a u8, can encode boolean and enumerated status flags. |

Table 3.0.2: SBP primitive types

#### Example Message

As an example, consider this framed series of bytes read from a serial port:

```
55 02 02 cc 04 14 70 3d d0 18 cf ef ff ff ef e8 ff ff f0 18 00 00 00 00 05 00 43 94
```

This byte array decodes into a `MSG_BASELINE_ECEF` (see pg. 9), that reports the baseline position solution of the rover receiver relative to the base station receiver in Earth Centered Earth Fixed (ECEF) coordinates. The segments of this byte array and its contents breakdown as follows:

| Field Name | Type | Value | Bytestring Segment |
|---|---|---|---|
| Preamble | u8 | 0x55 | 55 |
| Message Type | u16 | `MSG_BASELINE_ECEF` | 02 02 |
| Sender | u16 | 1228 | cc 04 |
| Length | u8 | 20 | 14 |
| Payload | | — | 70 3d d0 18 cf ef ff ff ef e8 ff ff f0 18 00 00 00 00 05 00 |
| `MSG_BASELINE_ECEF` | | | |
| .tow | u32 | 416300400 msec | 70 3d d0 18 |
| .x | s32 | −4145 mm | cf ef ff ff |
| .y | s32 | −5905 mm | ef e8 ff ff |
| .z | s32 | 6384 mm | f0 18 00 00 |
| .accuracy | u16 | 0 | 00 00 |
| .nsats | u8 | 5 | 05 |
| .flags | u8 | 0 | 00 |
| CRC | u16 | 0x9443 | 43 94 |

Table 3.0.3: SBP breakdown for `MSG_BASELINE_ECEF`

# 4 Message Types

Packages define a logical collection of SBP messages. The contents and layout of messages in packages marked **stable** are unlikely to change in the future. **Draft** messages *will change with future development* and are detailed purely for *informational purposes only*. Many draft messages are implementation-defined, and some collections, such as the acquisition package, are used for internal development.

| Package | Msg ID | Name | Size (bytes) | Description |
|---|---|---|---|---|
| **Stable** | | | | |
| **Draft** | | | | |

Table 4.0.5: SBP message types

# 5 Stable Message Definitions

# 6 Draft Message Definitions