Starling 1.2 Reference Manual

Version: 1.2.3
RM-120001-01
2020-11-12

Revision History

Version	Date	Description
0.9	2020-04-11	Initial version
0.91	2020-05-22	Incorporate documentation updates from *Sensorfusion configuration*
0.92	2020-05-30	Incorporate documentation updates from Swift platform documentation
1.1.11	2020-06-18	Updates for v1.1.11 release
1.1.14	2020-07-24	Minor updates based on internal review
1.2.0	2020-09-16	Update for v1.2 release
1.2.1	2020-10-15	Minor updates for Starling v1.2.27.
1.2.2	2020-10-28	Minor updates for Starling v1.2.30.
1.2.3	2020-11-12	Odometry mode config entry added, Starling v1.2.31.

Terms and Abbreviations

Term	Definition
6 DoF	6 Degrees of Freedom
ARP	Antenna Reference Point
DR	Dead Reckoning
IMU	Inertial Measurement Unit
NMEA	National Marine Electronics Association
NTRIP	Networked Transport of RTCM via Internet Protocol
PVAT	Position, Velocity, Attitude and Time
PVT	Position, Velocity and Time
RTCM	Radio Technical Commission for Maritime Services
SBP	Swift Binary Protocol
VRP	Vehicle Reference Point
YAML	Yet Another Mark-up Language

Reference Documents

ID	Revision	Publication Date	Title
[F9P]	R08	2020-05-28	u-blox F9 high precision GNSS receiver, Interface Description
[NMEA]	4.11	November 2018	NMEA 0183 Standard for Interfacing Marine Electronic Devices
[RTCM3]	Version 3 + Amendments 1 & 2	2013-11-07	RTCM Standard 10403.2, Differential GNSS (Global Navigation Satellite Systems) Services - Version 3
[SBP]	2.6.3	2019-06-10	Swift Navigation Binary Protocol, Protocol Specification

Table of Contents

1. Introduction

The Starling™ Positioning Engine is Swift Navigation's next-generation precise positioning library which is designed for automotive and autonomous vehicle applications. The Starling executable implements a run-time environment for running the Starling Positioning Engine in common usage scenarios.

2. Overview

The Starling Positioning Engine is internally comprised of two main subsystems, referred to as *engines*.

- The *GNSS Engine*, which ingests GNSS measurements to compute a Position, Velocity and Time (PVT) solution. An optional source of correction data (e.g. the Skylark™ cloud correction service) can be used to increase the accuracy and integrity of the output solution.
- The *Fusion Engine*, which is responsible for ingesting the output from the GNSS Engine along with data from external sensors (such as IMUs and Wheel Odometry) to compute position and attitude in GNSS-denied environments.

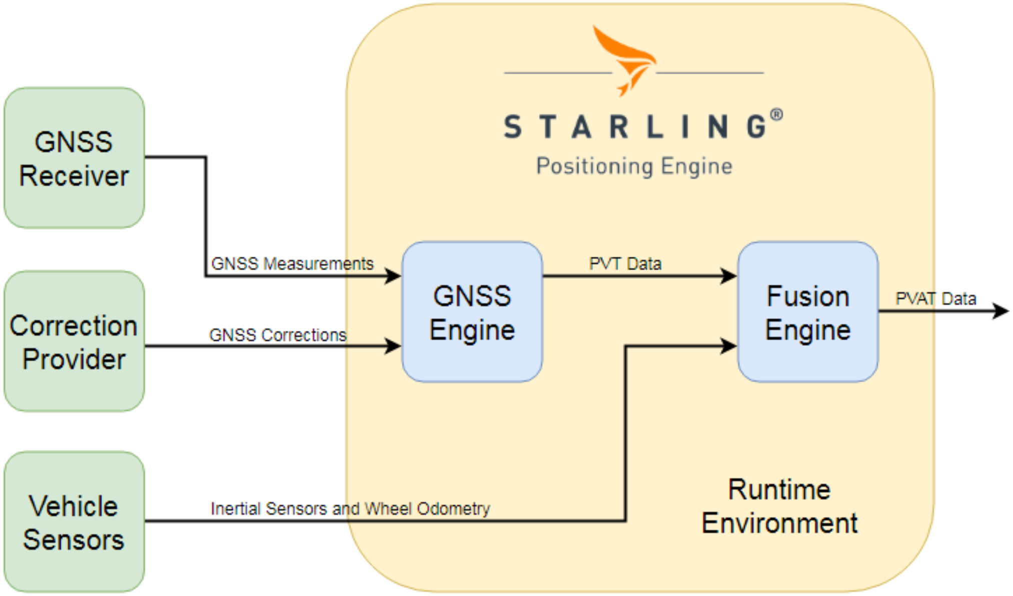

Figure 1: Starling Overview

These two engines are hosted in a runtime environment which is responsible for tasks such as message I/O, protocol conversion and interaction with the host platform.

2.1. GNSS Engine

The measurement data consists of both observations and ephemerides. Depending upon the use case, observation and ephemeris data may be provided separately or in a single stream. Ephemeris data may also be provided in the correction data stream, as is the case with Skylark™.

The following diagram provides a high-level overview of the inputs and outputs used by the GNSS Engine:

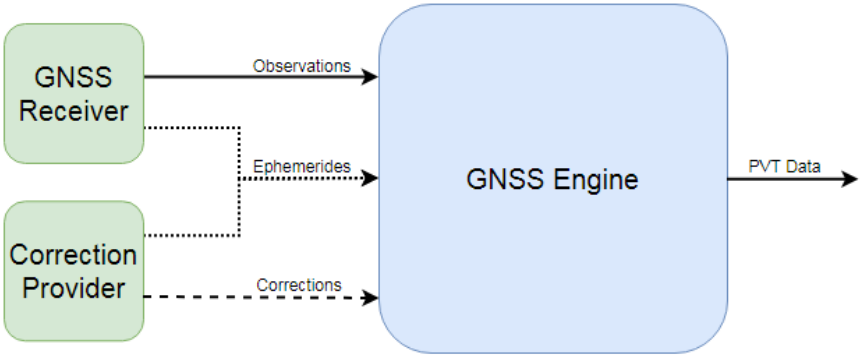

Figure 2: GNSS Engine Inputs and Outputs

Incoming data (i.e. observations, ephemerides and corrections) can be provided in RTCM v3, SBP or UBX formats. Output data can be generated in either SBP or NMEA 0183 formats. See Appendix A for a more detailed description of the input and output messages supported by Starling.

The GNSS engine generates position output for every incoming observation epoch, i.e. an incoming observation at rate of 10 Hz will result in an outgoing PVT rate of 10 Hz. Epochs without observations (e.g. due to GNSS signals obstructions) will result in no PVT output being generated.

2.2. Fusion Engine

The Fusion Engine combines the output from the GNSS Engine with data received from one or more vehicle sensors in order to compute a PVAT solution. The data flow is depicted in Figure 3.

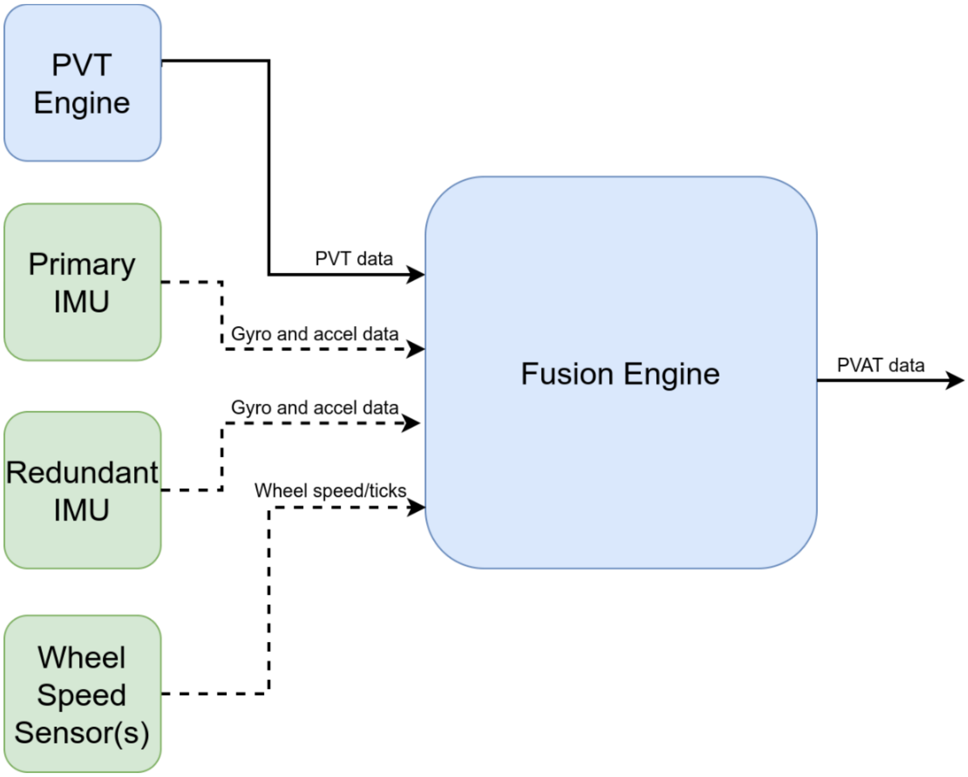

Figure 3: Fusion Engine Inputs and Outputs

The minimum vehicle sensor input required by the Fusion Engine is a single 6 DoF IMU. If data from a second IMU is provided, then this data will be cross-checked against the information from the primary sensor in order to detect sensor faults. Note that this is a mandatory requirement for ASIL-rated use cases.

The optional Wheel Odometry input can be used to constrain error growth when operating in pure Dead Reckoning mode, e.g. when driving through tunnels. Wheel Odometry input may be provided as either an on-ground speed value or as a number of wheel ticks (in which case the correct tire size must also be provided).

The Fusion Engine outputs data at a fixed rate once the initial filter alignment process is completed. This rate is specified in Starling configuration file. IMU data should be provided at least twice as fast as the Fusion Engine output rate, meaning that the minimum rate for IMU data is 20 Hz.

2.2.1. Alignment Process

The Fusion Engine needs to perform an alignment procedure before it can provide position aiding to the system.The Fusion Engine will begin its alignment process once the GNSS has achieved a minimum accuracy threshold (position standard deviation < 30 m, velocity standard deviation < 1 m/s) and a straight-line movement at > 5 m/s (20 km/h, 12 MPH) occurs. Alignment should typically complete within 20 - 50 m distance if sky visibility remains good.

2.3. Runtime Environment

The Swift runtime Platform Infrastructure supports a variety of I/O methods for incoming and outgoing data, namely:

- Binary files (see Section 3.2.5.2)
- Serial ports (see Section 3.2.5.3)
- Standard I/O streams (see Section 3.2.5.4)
- TCP client with optional NTRIP client (see Section 3.2.5.5)
- TCP servers (see Section 3.2.5.6)

These are referred to as *endpoints*. Each endpoint can be configured to use a given *protocol*. The supported protocols are as follows:

- `nmea`: NMEA 0183 ASCII sentences as defined in [NMEA]
- `ntrip`: Networked Transport of RTCM via Internet Protocol (for corrections only)
- `rtcm`: Version 3.2 of the RTCM standard as defined in [RTCM3]
- `sbp`: Swift Binary Protocol as defined in [SBP]
- `ubx`: u-blox UBX Protocol (v27.11 or higher) as defined in [F9P] (input only)

Additional information about the specific messages supported for each protocol can be found in Appendix A.

3. Usage

On typical platforms (like Linux PC) Starling can be launched from a terminal, example:

```
./starling-v1.2.31-x86 --config starling-cfg.yaml --log stdout
```

To stop Starling press Ctrl-C on the keyboard or send SIGTERM to the process.

The runtime configuration for Starling is provided via a YAML text file. The location of this file is provided to the starling executable using the `--config` parameter.

3.1. Command Line Parameters

3.1.1. Mandatory

The following command line parameter must be provided when invoking Starling:

Option	Type	Description
`--config filename`	string	Path and name of the configuration YAML file

3.1.2. Optional

The following optional command line parameters can be used to facilitate debugging:

Option	Type	Default	Description
`--log name`	string	`""`	Direct logging output to file. Accepts any valid path or the special values `stdout`/`stderr`
`--log_level`	string	`"warning"`	Filter log messages based on severity level. Log messages will only be output if they have a severity which is equal to or greater than the specified level. Valid options (in descending order of criticality) are `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `info`, `debug`, or the numerical enum value used by the starling configuration structures

--output_version	bool	False	Output version information to the log during execution
--verify_config	bool	False	Only validate whether the specified configuration file is formatted correctly and exit with success or failure based upon the parsing result. Note that initialisation of `starling` may still fail based on input values.
--record name	string	“”	Record all Starling input and output data in a separate directory. Upon each start a new directory with name and date, time will be created. Large files will be created. Use with caution.
--version			Outputs Starling version to stdout and quits

The `--log` parameter is of particular importance since the argument `--log stderr` can be supplied to diagnose any failures which may occur at startup (e.g. inability to listen on the specified TCP port).

3.2. YAML Configuration File

The Starling configuration is specified using a YAML text file which is formatted as follows:

```
---
name: <configuration name> {string}
 [combined-rover-input: <availability of additional sensor data> {bool}]
[solution-frequency: <solution frequency in Hertz> {float}]
gnss:
  type: <L1L2|L1L5|piksi-multi|LG69T-AP>
  rover:
    protocol: <nmea|rtcm|sbp|ubx>
    <endpoint> {object}
  [corrections:
    protocol: <nmea|ntrip|rtcm|sbp|ubx>
    <endpoint> {object}
    [ntrip-mount-point: <NTRIP mount point> {string}
    [ntrip-username: <NTRIP username> {string}]
    [ntrip-password: <NTRIP password> {string}]
    ntrip-gpgga-period: <GPGGA rate solution-frequency units> {uint}]]
[fusion:
  [imu:
    protocol: <nmea|rtcm|sbp|ubx>
    <endpoint> {object}]
  [wheel-odometry:
    protocol: <nmea|rtcm|sbp|ubx>
    <endpoint> {object}]
  [antenna-leverarm-meters-sensorframe:
    [x: <x coordinate> {double}]
    [y: <y coordinate> {double}]
    [z: <z coordinate> {double}]
    [deviation: <uncertainty of antenna leverarm measurement> {double}]]
  [wheelspeed-leverarm-meters-sensorframe:
    [x: <x coordinate> {double}]
    [y: <y coordinate> {double}]
    [z: <z coordinate> {double}]
    [deviation: <uncertainty of wheelspeed leverarm measurement> {double}]]
  [rotation-sensor-vehicle-degrees:
    [x: <x axis rotation> {double}]
    [y: <y axis rotation> {double}]
    [z: <z axis rotation> {double}]
    [deviation: <uncertainty of misalignment measurement> {double}]]
  [vrp-leverarm-meters-sensorframe:
    [x: <x coordinate> {double}]
    [y: <y coordinate> {double}]
    [z: <z coordinate> {double}]
    [deviation: <uncertainty of VRP coordinates in meters> {double}]
    [enable-transformation: <transformation to VRP> {bool}]]]
output:
  protocol: <nmea|rtcm|sbp>
  <endpoint> {object}
```

There are four high-level categories of settings which can be specified:

1. Global settings which are defined in the root section
2. A `gnss` section which defines the configuration of the GNSS Engine
3. An optional `fusion` section which defines the configuration of the Fusion Engine
4. An `output` section which defines the destination for Starling output

The supported options for each section are described in the remainder of this chapter. Appendix B lists an example YAML file which can be used for demonstration and testing purposes.

3.2.1. Global Settings

name

Type	string
Required	Yes
Default	N/A

Description	Text field containing string describing the configuration file. Spaces in the name are acceptable.

combined-rover-input

Type	bool
Required	No
Default	false
Description	Indicates that the GNSS input stream contains additional sensor data that may be used for sensor fusion.

solution-frequency

Type	float
Required	Yes, if `nmea` or `ntrip` endpoint is specified in the configuration
Default	N/A
Description	Solution frequency in Hz.

3.2.2. GNSS

The `gnss` section is mandatory and defines the configuration of the GNSS Engine. This section includes the `rover` and (optional) `corrections` subsections which are used to specify the input sources for measurement and correction data respectively.

type

Type	enum
Required	Yes
Default	N/A
Description	Configure the GNSS Engine for a pre-defined scenario. This includes, but is not limited to, configuration of the expected frequency ranges for incoming measurements and correction data. Valid values are `L1L2`, `L1L5, piksi-multi, LG69T-AP`. `L1L2` scenario can also be used for L1-only operation.

3.2.2.1. Rover

The `rover` section requires a mandatory endpoint (see Section 3.2.5) to define the source of measurement data.

protocol

Type	enum
Required	Yes
Default	N/A
Description	Protocol of incoming measurement data. Valid values are `nmea`, `rtcm`, `sbp` and `ubx`.

3.2.2.2. Corrections

The `corrections` section is optional. If it is specified then it requires a mandatory endpoint (see Section 3.2.5) to define the source of correction data.

protocol

Type	enum
Required	Yes
Default	N/A
Description	Protocol of incoming correction data. Valid values are `nmea`, `ntrip`, `rtcm`, `sbp` and `ubx`.

ntrip-mount-point

Type	string
Required	Yes, if `ntrip` protocol is specified for corrections
Default	N/A
Description	Mount point for NTRIP caster.

ntrip-username

Type	string
Required	No
Default	N/A
Description	User name for NTRIP caster. If not specified then no authorization procedure is attempted when a new NTRIP connection is established.

ntrip-password

Type	string
Required	No
Default	N/A
Description	Password for NTRIP caster. If not specified then no authorization procedure is attempted when a new NTRIP connection is established.

ntrip-gpgga-period

Type	uint
Required	Yes, if `ntrip` protocol is specified for corrections
Default	N/A
Description	Rate at which GGA sentences are sent to NTRIP caster. Set to zero to disable GGA output. Units are a multiple of solution period.

3.2.3. Fusion

The `fusion` section defines the configuration of the Fusion Engine. All subsections within this section are optional.

3.2.3.1. IMU

The `imu` section is not needed when the `combined-rover-input` option is enabled. Otherwise it is mandatory to specify an endpoint to define the source of IMU data.

protocol

Type	enum
Required	Yes
Default	N/A
Description	Protocol of incoming IMU data. Valid values are `nmea`, `rtcm`, `sbp` and `ubx`.

3.2.3.2. Wheel Odometry

The `wheel-odometry` section is optional. If it is specified then it requires a mandatory endpoint to define the source of wheel odometry data.

protocol

Type	Enum
Required	Yes
Default	N/A
Description	Protocol of incoming wheel odometry data. Valid values are `nmea`, `rtcm`, `sbp` and `ubx`.

3.2.3.3. Antenna Lever Arm

The optional `antenna-leverarm-meters-sensorframe` section specifies the offset vector from the IMU reference point to the GNSS antenna phase center. These values should be as accurate as possible in order to achieve the highest possible inertial fusion performance.

x

Type	double
Required	No
Default	0.0
Description	Antenna lever arm X axis distance (metres).

y

Type	double
Required	No
Default	0.0
Description	Antenna lever arm Y axis distance (metres).

z

Type	double
Required	No
Default	0.0
Description	Antenna lever arm Z axis distance (metres).

deviation

Type	double
Required	No
Default	1.0
Description	Standard deviation of antenna lever arm measurement (metres). Must be greater than 0. This value should overestimate the actual expected error.

3.2.3.4. Wheel Speed Lever Arm

The optional `wheelspeed-leverarm-meters-sensorframe` section specifies the offset vector from the centre of navigation (i.e. accelerometer intersection point) to the wheel contact point. These values should be as accurate as possible in order to achieve the highest possible dead reckoning performance.

x

Type	double
Required	No
Default	0.0
Description	Wheel speed lever arm X coordinate (metres).

y

Type	double
Required	No
Default	0.0
Description	Wheel speed lever arm Y coordinate (metres).

z

Type	double
Required	No
Default	0.0
Description	Wheel speed lever arm Z coordinate (metres).

deviation

Type	double
Required	No
Default	1.0
Description	Standard deviation of wheel speed lever arm measurement (metres). Must be greater than 0. This value should overestimate the actual expected error.

3.2.3.5. Rotation Sensor

The optional `rotation-sensor-vehicle-degrees` section specifies the intrinsic rotation sequence from the sensor to the vehicle frame of reference. The rotations are applied in ZYX order.

x

Type	double
Required	No
Default	0.0
Description	Rotation around the X axis (degrees).

y

Type	double
Required	No
Default	0.0
Description	Rotation around the Y axis (degrees).

z

Type	double
Required	No
Default	0.0
Description	Rotation around the Z axis (degrees).

deviation

Type	double
Required	No
Default	10.0
Description	Standard deviation of misalignment measurement (degrees). Must be greater than 0. This value should overestimate the actual expected error.

3.2.3.6. Vehicle Reference Point

The optional `vrp-leverarm-meters-sensorframe` section allows a custom reference point to be specified in the vehicle frame of reference. If specified, the PVAT output from the Fusion Engine will be relative to this point. If not specified then the antenna phase centre will be used.

x

Type	double
Required	No
Default	0.0
Description	Vehicle Reference Point lever arm X coordinate (metres).

y

Type	double
Required	No
Default	0.0
Description	Vehicle Reference Point lever arm Y coordinate (metres).

z

Type	double
Required	No
Default	0.0
Description	Vehicle Reference Point lever arm Z coordinate (metres).

deviation

Type	double
Required	No
Default	0.01
Description	Standard deviation of Vehicle Reference Point coordinates (metres). Must be greater than 0.

enable-transform

Type	bool
Required	No
Default	false
Description	Enable transformation of PVAT output to Vehicle Reference Point.

3.2.3.7. Odometry Mode

odometry-mode

Type	String
Required	No
Default	wheelticks
Description	Sets up the source and usage of odometry information inside of the fusion engine. Allowed values: `wheelticks, speed-sensor, motion-detection-only, do-not-use`

3.2.3.8. Tuning Profile

tuning-profile

Type	String
Required	No
Default	default

Description	Enables custom sensor fusion profile. Allowed values: `default`, `car-a`, `car-b`, `car-c`.

3.2.4. Output

The `output` section is mandatory and defines the destination for Starling output. It requires a mandatory endpoint (see Section 3.2.5) to define the destination of the data.

protocol

Type	enum
Required	Yes
Default	N/A
Description	Protocol for Starling output. Valid values are `nmea`, `rtcm`, `sbp`.

3.2.5. Endpoints

An endpoint object represents an input or output device which is used to communicate data between Starling and the host platform. An endpoint object has a mandatory `type` field which must be set to one of the following values: `file`, `serial`, `stdstream`, `tcp-client`, `tcp-server` or `udp`. The `type` field must be followed by a number of additional fields which vary depending upon the endpoint type. An endpoint definition may also contain additional optional parameters which do not depend on the endpoint type.

3.2.5.1. Optional Parameters

buffer-size

Type	uint
Required	No
Default	0
Description	The maximum frame size to use for the given protocol. A value of `0` means that a protocol-specific default value will be used.

3.2.5.1.1. NMEA

These parameters are only valid for the NMEA protocol and specify the rate at which different NMEA 0183 sentences will be sent to the output. All output sentences use the `GP` talker identifier.

gpgga-period

Type	uint
Required	No
Default	0
Description	Period at which GPGGA sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gpgll-period

Type	uint
Required	No
Default	0
Description	Period at which GPGLL sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gpgsa-period

Type	uint
Required	No
Default	0
Description	Period at which GPGSA sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gpgst-period

Type	uint
Required	No
Default	0
Description	Period at which GPGST sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gpgsv-period

Type	uint
Required	No

Default	0
Description	Period at which GPGSV sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gphdt-period

Type	uint
Required	No
Default	0
Description	Period at which GPHDT sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gprmc-period

Type	uint
Required	No
Default	0
Description	Period at which GPRMC sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gpvtg-period

Type	uint
Required	No
Default	0
Description	Period at which GPVTG sentences will be sent to the output. Units are a multiple of `solution-frequency`.

gpzda-period

Type	uint
Required	No
Default	0

Description	Period at which GPZDA sentences will be sent to the output. Units are a multiple of `solution-frequency`.

3.2.5.2. File Endpoint

A `file` endpoint represents an abstract file as supported by the host platform. Note that this may be a device path on targets which represent devices as files (e.g. UNIX-type platforms).

path

Type	string
Required	Yes
Default	N/A
Description	A file path which is meaningful to the underlying platform.

3.2.5.3. Serial Endpoint

A `serial` endpoint represents a serial endpoint on the host platform (e.g. UART or TTY device).

identifier

Type	string
Required	Yes
Default	N/A
Description	A serial device identifier which is meaningful to the underlying platform, e.g. /dev/tty0.

baud-rate

Type	uint32
Required	Yes
Default	N/A
Description	Baud rate for the serial device.

byte-size

Type	int

Required	Yes
Default	N/A
Description	Number of data bits for the serial device. Valid values are `7` and `8`.

parity

Type	enum
Required	Yes
Default	N/A
Description	Parity configuration for the serial device. Valid values are `N` (none), `O` (odd), and `E` (even).

stop-bits

Type	int
Required	Yes
Default	N/A
Description	Number of stop bits for the serial device. Valid values are `1` and `2`.

flow-control

Type	bool
Required	No
Default	false
Description	Enable hardware handshaking for the serial device.

3.2.5.4. Standard Stream Endpoint

A `stdstream` endpoint represents a standard input, standard output or standard error stream on the host platform.

stdstream-type

Type	enum

Required	Yes
Default	N/A
Description	Stream to use. Valid options are `stdin`, `stdout` and `stderr`.

3.2.5.5. TCP Client Endpoint

A `tcp-client` endpoint represents a TCP socket on the host platform.

host

Type	string
Required	Yes
Default	N/A
Description	Remote server hostname or IP address.

port

Type	uint16
Required	Yes
Default	N/A
Description	TCP port number on remote server.

ip-version

Type	enum
Required	No
Default	any
Description	Specifies the IP version to use when setting up a TCP connection. Valid values are `any`, `ipv4` and `ipv6`.

connect-timeout

Type	duration
Required	No
Default	0

Description	Maximum time that Starling will wait before giving up on a TCP connection attempt. A value of `0` means that it will wait indefinitely.

3.2.5.5.1. TCP Keep-Alive

This optional subsection describes a set of options for TCP keep-alive functionality.

enable

Type	bool
Required	No
Default	false
Description	Enable TCP keep-alive functionality.

idle

Type	duration
Required	Yes
Default	N/A
Description	Length of time that a connection remains idle before TCP starts sending keep alive probes.

interval

Type	duration
Required	Yes
Default	N/A
Description	Length of time between individual keep alive probes.

retries

Type	uint16
Required	Yes
Default	N/A
Description	Maximum number of keep alive probes that TCP should send before dropping the connection.

3.2.5.6. TCP Server Endpoint

A `tcp-server` endpoint represents a TCP server on the host platform.

port

Type	uint16
Required	Yes
Default	N/A
Description	Local port number to listen on for incoming TCP connections.

max-conns

Type	uint
Required	Yes
Default	N/A
Description	Maximum number of client connections to support.

ip-version

Type	enum
Required	No
Default	any
Description	Specifies the IP version to use when setting up a TCP connection. Valid values are `any`, `ipv4` and `ipv6`.

3.2.5.6.1. TCP Keep-Alive

This optional subsection describes a set of options for TCP keep-alive functionality.

enable

Type	bool
Required	No
Default	false
Description	Enable TCP keep-alive functionality.

idle

Type	duration
Required	Yes
Default	N/A
Description	Length of time that a connection remains idle before TCP starts sending keep alive probes.

interval

Type	duration
Required	Yes
Default	N/A
Description	Length of time between individual keep alive probes.

retries

Type	uint16
Required	Yes
Default	N/A
Description	Maximum number of keep alive probes that TCP should send before dropping the connection.

3.2.5.7. UDP Endpoint

A `udp` endpoint represents a UDP socket endpoint on the host platform.

host

Type	string
Required	Yes
Default	N/A
Description	Remote server hostname or IP address.

port

Type	uint16

Required	Yes
Default	N/A
Description	UDP port number on the remote server.

ip-version

Type	enum
Required	Yes
Default	N/A
Description	Specifies the IP version to use when sending UDP data. Valid values are `ipv4` and `ipv6`.

3.2.6. Durations

Fields of type `duration` are specified using the format `<int64><ns|us|ms|s|m|h>`.

The `int64` is the numerical value whereas the `<ns|us|ms|s|m|h>` portion represents the unit. Therefore, an entry of `10ns` indicates 10 nanoseconds and `5m` corresponds to 5 minutes.

Appendix A. Supported Messages

A.1. GNSS Engine Inputs

The GNSS Engine requires GNSS measurements from a Measurement Engine to compute a PVT solution. More precisely, it requires:

- Pseudorange, phase range and CNR observables from the satellites
- Ephemeris data (optional)
- GLONASS L1 and L2 code phase biases (if using GLONASS)

If a correction service is used to improve the accuracy, the GNSS Engine will use:

- Pseudorange, phase range and CNR observables from the satellites
- Reference station location
- Ephemeris data (optional)
- GLONASS L1 and L2 code phase biases (if using GLONASS)

For the inputs, the GNSS Engine supports any one of three protocols:

1. RTCM v3.2. See [RTCM3] for further details.
2. Swift Binary Protocol (SBP). See [SBP] for further details.
3. u-blox UBX Protocol (v27.11 or higher). See [F9P] for further details.

A.1.1. Rover Measurements

A.1.1.1. Pseudorange, Phase Range and CNR Observables

The following messages are required if measurement input is provided in RTCM v3.2 format:

Message ID	Description
1074 to 1077	GPS
1084 to 1087	GLONASS
1094 to 1097	Galileo
1124 to 1127	BeiDou
1114 to 1117	QZSS

The following message is required if measurement input is provided in SBP format:

Message ID	Description
74	MSG_OBS

The following message is required if measurement input is provided in UBX format:

Message ID	Description
0x02 0x15	UBX-RXM-RAWX

A.1.1.2. Ephemeris Data

The following messages are required if measurement input is provided in RTCM v3.2 format and the Measurement Engine is to act as the source of ephemerides:

Message ID	Description
1019	GPS
1020	GLONASS
1045 and 1046	Galileo
1042	BeiDou
1044	QZSS

The following messages are required if measurement input is provided in SBP format and the Measurement Engine is to act as the source of ephemerides:

Message ID	Description
138	MSG_EPHEMERIS_GPS
139	MSG_EPHEMERIS_GLO
141	MSG_EPHEMERIS_GAL
137	MSG_EPHEMERIS_BDS
142	MSG_EPHEMERIS_QZSS

The following messages are required if measurement input is provided in UBX format and the Measurement Engine is to act as the source of ephemerides:

Message ID	Description
0x02 0x13	UBX-RXM-SFRBX

A.1.1.3. GLONASS L1 and L2 Code Phase Biases

The following message is required if measurement input is provided in RTCM v3.2 format:

Message ID	Description
1230	GLONASS L1 and L2 Code-Phase Biases

The following message is required if measurement input is provided in SBP format:

Message ID	Description
117	MSG_GLO_BIASES

A.1.2. Corrections

A.1.2.1. Pseudorange, Phase Range and CNR Observables

The following messages are required if correction input is provided in RTCM v3.2 format:

Message ID	Description
1074 to 1077	GPS
1084 to 1087	GLONASS
1094 to 1097	Galileo
1124 to 1127	BeiDou
1114 to 1117	QZSS

The following message is required if correction input is provided in SBP format:

Message ID	Description
74	MSG_OBS

A.1.2.2. Reference Station Location

The following messages are required if correction input is provided in RTCM v3.2 format:

Message ID	Description
1005	Stationary RTK reference station ARP
1006	Stationary RTK reference station ARP with antenna height

The following message is required if correction input is provided in SBP format:

Message ID	Description
72	MSG_BASE_POS_ECEF

A.1.2.3. Ephemeris Data

The following messages are required if correction input is provided in RTCM v3.2 format and the correction provider is to act as the source of ephemerides:

Message ID	Description
1019	GPS
1020	GLONASS
1045 and 1046	Galileo
1042	BeiDou
1044	QZSS

The following messages are required if correction input is provided in SBP format and the correction provider is to act as the source of ephemerides:

Message ID	Description
138	MSG_EPHEMERIS_GPS
139	MSG_EPHEMERIS_GLO
141	MSG_EPHEMERIS_GAL
137	MSG_EPHEMERIS_BDS
142	MSG_EPHEMERIS_QZSS

A.1.2.4. GLONASS L1 and L2 Code Phase Biases

The following message is required if correction input is provided in RTCM v3.2 format:

Message ID	Description
1230	GLONASS L1 and L2 Code-Phase Biases

The following message is required if correction input is provided in SBP format:

Message ID	Description
117	MSG_GLO_BIASES

A.2. Fusion Engine Inputs

The following SBP messages are used to provide vehicle sensor input to the Fusion Engine:

Message ID	Description
2304	SBP_MSG_IMU_RAW
2305	SBP_MSG_IMU_AUX
2307	SBP_MSG_ODOMETRY
2308	SBP_MSG_WHEELTICK

See [SBP] for further information about the SBP protocol and the contents of these messages.

A.3. SBP Output Messages

The table below lists the outputs generated by the PVT and Fusion Engines. Note that messages from other sender IDs may be present in the output depending upon the specifics of the executing platform and input sensor configuration. For example, if correction data is provided then it will be forwarded with sender ID 0.

The output rate of the messages originating from the GNSS Engine depends upon the incoming rate of observations.

The SBP messages with ID (decimal) 258, 259, 522, 529, 521, 532, 526, 530, 525, 533, 545, 65283, 65286, and 65295 belong to the “Best Position” group and contain the best possible output from all available subsystems.

Sender ID	Source	Message ID (dec)	Message ID (hex)	Type
4096	GNSS Engine	260	104	GPS_TIME_GNSS
4096	GNSS Engine	261	105	UTC_TIME GNSS
4096	GNSS Engine	520	208	DOPS
4096	GNSS Engine	523	20B	BASELINE_ECEF
4096	GNSS Engine	524	20C	BASELINE_NED
4096	GNSS Engine	528	210	AGE_CORRECTIONS
4096	GNSS Engine	534	216	PROTECTION_LEVEL

4096	GNSS Engine	553	229	POS_ECEF_GNSS
4096	GNSS Engine	554	22A	POS_LLH_GNSS
4096	GNSS Engine	557	22D	VEL_ECEF_GNSS
4096	GNSS Engine	558	22E	VEL_NED_GNSS
4096	GNSS Engine	561	231	POS_LLH_COV_GNSS
4096	GNSS Engine	562	232	VEL_NED_COV_GNSS
4096	GNSS Engine	564	234	POS_ECEF_COV_GNSS
4096	GNSS Engine	565	235	VEL_ECEF_COV_GNSS
4096	GNSS Engine	1025	401	LOG
4096	GNSS Engine	65282	FF02	DGNSS_STATUS
789	Fusion Engine	258	102	GPS_TIME
789	Fusion Engine	259	103	UTC_TIME
789	Fusion Engine	521	209	POS_ECEF
789	Fusion Engine	522	20A	POS_LLH
789	Fusion Engine	525	20D	VEL_ECEF
789	Fusion Engine	526	20E	VEL_NED
789	Fusion Engine	529	211	POS_LLH_COV
789	Fusion Engine	530	212	VEL_NED_COV
789	Fusion Engine	532	214	POS_ECEF_COV
789	Fusion Engine	533	215	VEL_ECEF_COV
789	Fusion Engine	545	221	ORIENT_EULER
789	Fusion Engine	65283	FF03	INS_STATUS
789	Fusion Engine	65286	FF06	INS_UPDATES
789	Fusion Engine	65290	FF0A	GROUP_META
789	Fusion Engine	65295	FF0E	SOLN_META

See [SBP] for further information about the SBP protocol and the contents of these messages.

A.4. NMEA Output Messages

The following messages are supported for the NMEA output mode:

Sentence Formatter	**Description**
GGA	Global Positioning System Fix Data
GLL	Geographic Position – Latitude/Longitude
GSA	GNSS DOP and Active Satellites
GST	GNSS Pseudorange Error Statistics
GSV	GNSS Satellites In View
HDT	Heading, True
RMC	Recommended Minimum Specific GNSS Data
VTG	Course Over Ground & Ground Speed
ZDA	Time and Date

See [NMEA] for further information about the NMEA 0183 protocol and the contents of these sentences.

Appendix B. Example YAML Configuration File

The following YAML file will configure Starling as follows:

- Receive measurements in RTCM3 format from 192.168.1.101, port 52302
- Receive corrections in RTCM v3 format from Skylark EU (eu.skylark.swiftnav.com, port 2101) via NTRIP
- Send GGA sentences to Skylark at 1 Hz
- Read IMU data from `/dev/imu` in SBP format
- Use a lever arm of (0.25, 0.939, 1.14) from the IMU to the antenna phase centre
- Rotate the raw IMU data by 90 degrees around Z followed by a 180 degrees rotation around X
- Output PVT data in SBP format via a TCP server running on port 55555

```
---
name: Example YAML configuration file
solution-frequency: 1
gnss:
  type: L1L2
  rover:
    protocol: rtcm
    type: tcp-client
    host: 192.168.1.101
    port: 52302
  corrections:
    protocol: ntrip
    type: tcp-client
    host: eu.swiftnav.com
    port: 2101
    ntrip-username: <username>
    ntrip-password: <password>
    ntrip-mount-point: OSR
    ntrip-gpgga-period: 1
fusion:
  imu:
    protocol: sbp
    type: file
    path: /dev/imu
  antenna-leverarm-meters-sensorframe:
    x: 0.25
    y: 0.939
    z: 1.14
    deviation: 0.01
  rotation-sensor-vehicle-degrees:
    z: 90
    y: 0
    x: 180
    deviation: 1
  wheelspeed-levearm-meters_sensorframe:
    x: 0.0
    y: 0.0
    z: 0.0
    deviation: 0.1
output:
  protocol: sbp
  type: tcp-server
  port: 55555
  max-conns: 10
```