Starling 1.4 Reference Manual

Document Version: 1
Document Number: RM-120005-01
Publication Date: 2021-06-25

Revision History

Version	Date	Description
1	2021-06-25	Initial version for Starling v1.4 release

Terms and Abbreviations

Term	Definition
6 DoF	6 Degrees of Freedom
ANTEX	Antenna Exchange Format
ARP	Antenna Reference Point
CORS	Continuously Operated Reference Station
CRS	Cloud Reference Station
DR	Dead Reckoning
EHPE	Estimated Horizontal Position Error
IGS	International GNSS Service
IMU	Inertial Measurement Unit
NMEA	National Marine Electronics Association
NTRIP	Networked Transport of RTCM via Internet Protocol
OSR	Observation State Representation
PVAT	Position, Velocity, Attitude and Time
PVT	Position, Velocity and Time
RTCM	Radio Technical Commission for Maritime Services
SBP	Swift Binary Protocol
Skylark	Swift Navigation Cloud Correction Service
SSR	State Space Representation
Starling	Swift Navigation Positioning Engine
VRP	Vehicle Reference Point
YAML	Yet Another Mark-up Language

Reference Documents

ID	Revision	Publication Date	Title
[ANT]	1.4	2010-09-15	ANTEX: The Antenna Exchange Format
[ASM]	Rev 4	May 2020	ASM330LHH, Automotive 6-axis inertial module: 3D accelerometer and 3D gyroscope, Datasheet (Doc ID 031239)
[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]	3.4.7	2021-03-11	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. For more sophisticated usage scenarios it may be necessary to use an alternate run-time environment or interact directly with the Starling™ Library API instead.

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.

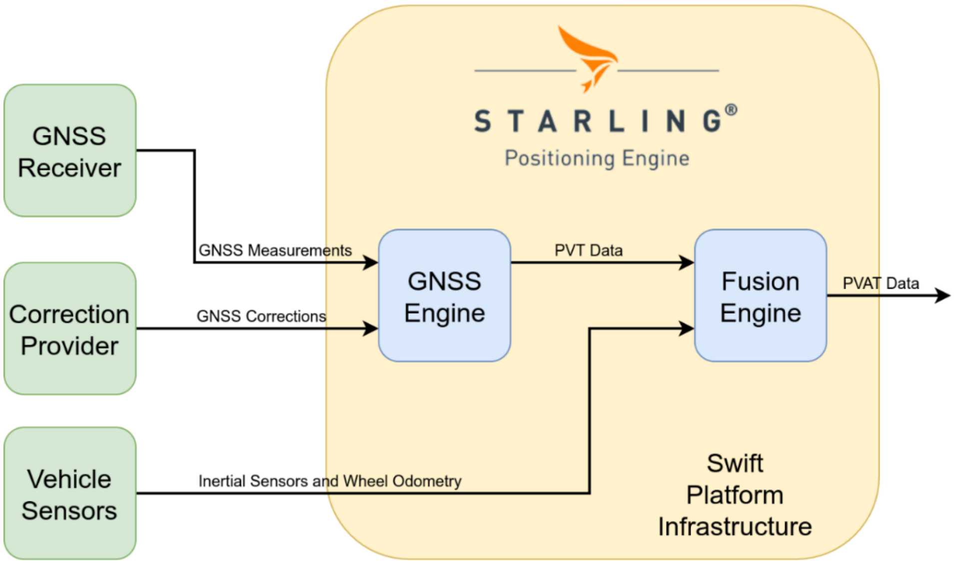

Figure 1: Starling Overview

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

2.1. GNSS Engine

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. Certain use cases (such as post-processing) may also combine measurement and correction data into a single stream.

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

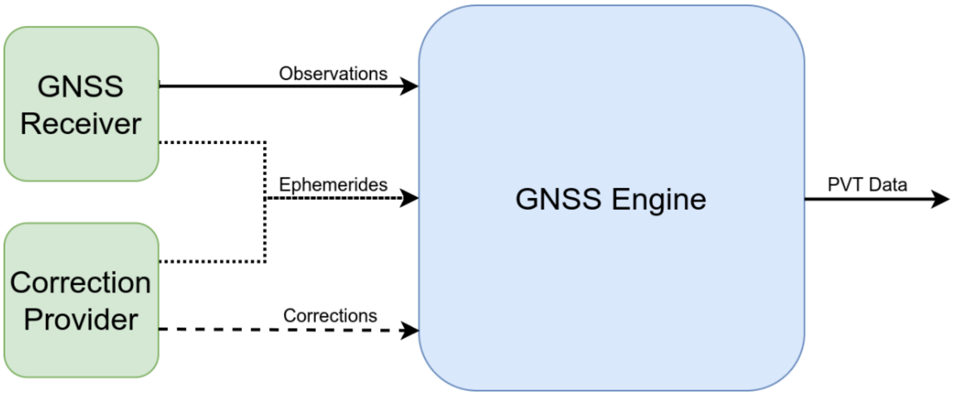

Figure 2: GNSS Engine Inputs and Outputs

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 rate of 10 Hz will result in an outgoing PVT rate of 10 Hz.

2.1.1. Correction Data Formats

The GNSS Engine is capable of receiving correction data in either Observation State Representation (OSR) or State Space Representation (SSR) formats. A full list of the messages required for OSR corrections can be found in Appendix A.1.2.

SSR corrections are supported when the following conditions are met:

1. A valid antenna database file is found in the location specified by the `external-data-path` parameter (see Section 3.3.2). The antenna database must be named `igs14.atx` and should correspond to the `igs14_2132.atx` file available from IGS, i.e. phase centre corrections for satellite and receiver antennas in the IGS14 terrestrial reference frame from GPS week 2132.
2. Starling is configured to receive corrections from a Skylark SSR mountpoint.

Note: It is not necessary to configure outgoing GGA sentences when using SSR corrections since it is a broadcast-only technology.

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. This data flow is depicted in Figure 3.

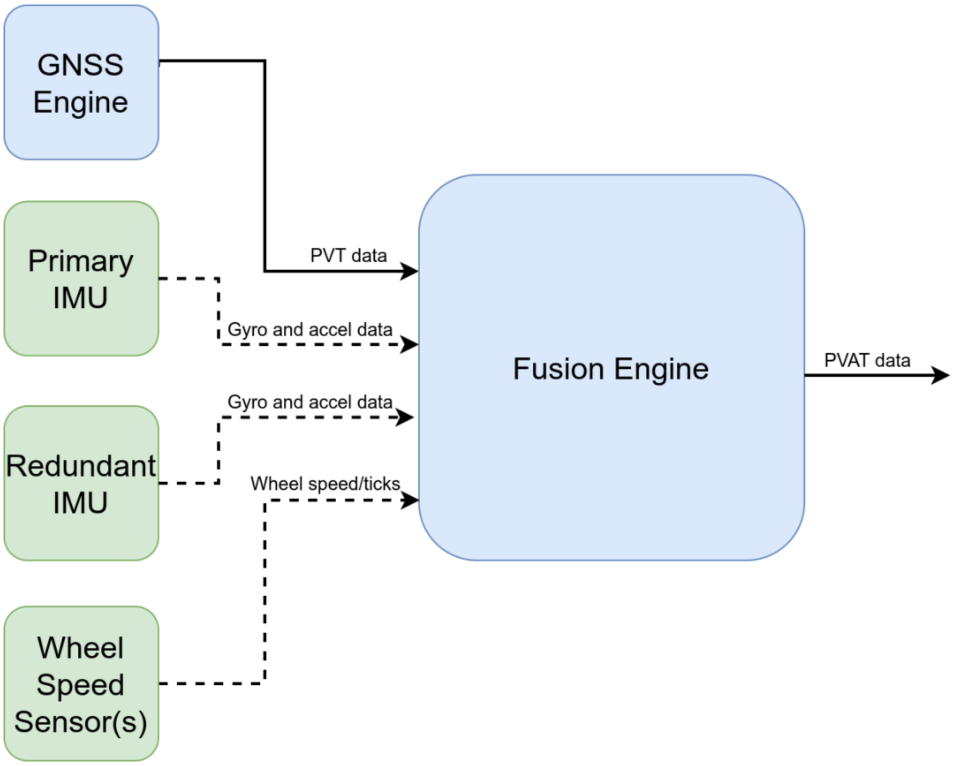

Figure 3: Fusion Engine Inputs and Outputs

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 (using MSG_ODOMETRY) or as a number of wheel ticks (using MSG_WHEELTICKS).

The output rate of the Fusion Engine can be configured using the `solution-rate` parameter (see Section 3.3.4.10). IMU data should be provided at least twice as fast as the Fusion Engine output rate.

2.2.1. Alignment Process

The Fusion Engine needs to perform an alignment procedure before it can provide positioning assistance to the system. The alignment process will begin once the following conditions are met:

1. The position standard deviation reported by the GNSS Engine is less than 30 m
2. The velocity standard deviation reported by the GNSS Engine is less than 1 m/s
3. Straight-line movement occurs at a speed above 5 m/s (20 km/h, or 12 MPH)

Alignment should typically complete within a distance of 20 to 50 m if sky visibility remains good during the initialisation phase.

2.2.2. Fast Start

The *Fast Start* feature allows the state of the Fusion Engine to be persistently stored while Starling is not active. This enables the Fusion Engine to enter an aligned state immediately after Starling is started without needing to follow the alignment process outlined above. Fast Start mode activates when the following conditions are met:

1. Valid alignment data is found in the *seed file* specified by the `fast-start-path` parameter (see Section 3.3.4.9) at Starling start-up
2. The EHPE read from the previously stored alignment data is less than 10 m
3. The speed read from the previously stored alignment data is less than 1 m/s

These thresholds may be adjusted using the parameters listed in Section 3.3.4.8. If the `periodic-storage-interval` parameter is specified then the seed file will be written to disk at the specified rate. If it is not specified then the seed file will only be written to disk when Starling is terminated gracefully (e.g. via SIGTERM on Linux targets).

2.3. Runtime Environment

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

- Binary files (see Section 3.3.7.2)
- Serial ports (see Section 3.3.7.3)
- Standard I/O streams (see Section 3.3.7.4)
- TCP client with optional NTRIP client (see Section 3.3.7.5)
- TCP servers (see Section 3.3.7.6)

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

- `ixcom`: Binary protocol supported by products from iMAR Navigation & Control; can be used for IMU and wheeltick input only.
- `nmea`: NMEA 0183 ASCII sentences as defined in [NMEA]; can be used for position output only.
- `ntrip`: Networked Transport of RTCM via Internet Protocol; can be used for correction input only.
- `rtcm`: Version 3.2 of the RTCM standard as defined in [RTCM3]; can be used for GNSS measurement and correction input only.
- `sbp`: Swift Binary Protocol as defined in [SBP]; can be used for all inputs and outputs.
- `ubx`: u-blox UBX Protocol (v27.11 or higher) as defined in [F9P]; can be used for GNSS and IMU input only.

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

3. Usage

Starling is a command-line application which accepts a number of optional arguments. An example invocation is as follows:

```
./starling-v1.4.0-x86_64 --config starling.yaml --log stdout
```

Depending upon the platform, the Starling executable may require an active license to operate (see Section 3.2 for more information). If a license file is required then an example invocation may be as follows:

```
./starling-v1.4.0-x86_64 --config starling.yaml --license license.lic
  --activation_key activation-key.txt --license license.lic --log stdout
```

The runtime configuration for Starling is provided via a YAML file. The format of this file is described in Section 3.3. The location of this file is provided to the Starling executable using the `--config` parameter.

Depending upon the configuration, Starling may terminate after completion or wait indefinitely for further input. In either case the program can be terminated by pressing Ctrl-C on the keyboard or sending SIGTERM to the process.

3.1. Command Line Parameters

Starling command line options may be specified using one or two dashes before the option name and may be specified in any order.

3.1.1. Mandatory

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

Option	Type	Default	Description
`--config filename`	string	N/A	Path to configuration YAML file

3.1.2. Optional

The following optional command line parameters are supported:

Option	Type	Default	Description
`--activation_key filename`	string	N/A	Path to activation key text file (may be mandatory, depending on platform)
`--license filename`	string	N/A	Path to license text file (may be mandatory, depending on platform)
`--log name`	string	N/A	Direct logging output to file. Accepts any valid path or the special values `stdout`/`stderr`
`--log_level 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
`--record name`	string	N/A	Record all Starling input and output data in a separate directory
`--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.

`--version`	N/A	N/A	Output Starling version to standard output and exit

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

3.2. License Activation

Versions of the Starling executable which are intended for open platforms require an active license to operate. The license activation process must be performed only once during the first execution of Starling. In order to activate a licence, the user must obtain a so-called *guard file* and an activation code from Swift Navigation. The device hosting Starling must have access to the Internet to perform the activation procedure.

The license activation procedure is as follows:

1. Obtain a guard file and activation code from Swift Navigation by submitting a support request ticket on the [Swift Navigation Support Portal]. Note that the *Support Request* button can be found at the bottom of the web page.
2. Ensure that the hosting platform has access to the Internet.
3. Ensure that the `starling` binary and `starling-guard.json` files are in the same directory.
4. Write the activation code (a 16 digit number in the form XXXX-XXXX-XXXX-XXXX) to a text file, e.g. `activation-key.txt`.
5. Launch the Starling application with the `--activation_key` and `--license` parameters. The `--activation_key` parameter should specify the location of the file created at step 4, and the `--license` parameter should specify an output file where the generated license should be written to (following initial activation) or read from (in subsequent executions of Starling).

Upon successful activation, a license file will be written to the location specified by the `--license` parameter. In case of error, an error message will be written to the output log.

3.3. YAML Configuration File

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

```
---
name: <configuration name> {string}
combined-rover-input: <availability of additional sensor data> {bool}
solution-frequency: <solution frequency in Hz> {float}
external-data-path: <path for read-only GNSS data> {string}
```

```
gnss:
  type: <type name> {enum}
  correction-age-max: <maximum age of corrections> {unsigned int}
  rover:
    protocol: <protocol name> {enum}
    <endpoint> {object}
  corrections:
    protocol: <protocol name> {enum}
    <endpoint> {object}
    ntrip-mount-point: <NTRIP mount point> {string}
    ntrip-username: <NTRIP username> {string}
    ntrip-password: <NTRIP password> {string}
    ntrip-gpgga-period: <GPGGA rate> {unsigned int}
fusion:
  imu:
    protocol: <protocol name> {enum}
    <endpoint> {object}
  wheel-odometry:
    protocol: <protocol name> {enum}
    <endpoint> {object}
  odometry-mode: <odometry mode> {enum}
  antenna-leverarm-meters-sensorframe:
    x: <x coordinate> {float}
    y: <y coordinate> {float}
    z: <z coordinate> {float}
    deviation: <uncertainty of antenna leverarm measurement> {float}
  wheelspeed-leverarm-meters-sensorframe:
    x: <x coordinate> {float}
    y: <y coordinate> {float}
    z: <z coordinate> {float}
    deviation: <uncertainty of wheelspeed leverarm measurement> {float}
  rotation-sensor-vehicle-degrees:
    x: <x axis rotation> {float}
    y: <y axis rotation> {float}
    z: <z axis rotation> {float}
    deviation: <uncertainty of misalignment measurement> {float}
  vrp-leverarm-meters-sensorframe:
    x: <x coordinate> {float}
    y: <y coordinate> {float}
    z: <z coordinate> {float}
    deviation: <uncertainty of VRP coordinates in meters> {float}
    enable-transformation: <transformation to VRP> {bool}
  fast-start:
    ehpe-limit-meters: <EHPE limit> {float}
    speed-limit-meters-per-second: <speed limit> {float}
    periodic-storage-interval: <storage interval> {duration}
  fast-start-path: <file path> {string}
  solution-rate: <fusion engine output rate in Hz> {double}
  tuning-profile: <profile name> {enum}
periodic:
  heartbeat-period: <time between heartbeats> {duration}
  version-period: <time between version log messages> {duration}
outputs:
  - protocol: <protocol name> {enum}
    name: <name of output> {string}
    mask: <mask>
    <endpoint> {object}
  - protocol: <protocol name> {enum}
    name: <name of output> {string}
    mask: <mask>
    <endpoint> {object}
  ...
```

There are five 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 optional `periodic` section which defines the configuration for periodic system status messages
5. An `outputs` section which defines the destination(s) 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.

3.3.1. Data Types

Each parameter in the YAML configuration file has a corresponding *data type*. The data types supported by Starling are as follows:

- *bool*: A Boolean value may be true or false. These fields are case-insensitive, meaning that the values `TRUE`, `True` and `true` are equivalent, as are the values `FALSE`, `False` and `false`.
- *string*: A free-form text field which may contain spaces.
- *integer*: A whole number with no decimal point.
- *float*: A number consisting of an integer which is optionally followed by a decimal point and additional precision digits.
- *enum*: An enumeration value which may only equal one of a number of predefined valid values.
- *duration*: An integer followed by a unit, used for specifying lengths of time. See Section 3.3.1.1 for more information about durations.
- *object*: Object types are used when specifying endpoints. See Section 3.3.7 for more information about endpoints.

3.3.1.1. Duration Types

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.

3.3.2. Global Settings

name

Type	string
Required	Yes
Default	N/A
Description	Free-form text field describing the configuration file. The text field may contain spaces.
Example	`name: Starling config for Car 1 with Skylark`

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.
Example	`combined-rover-input: true`

solution-frequency

Type	float
Units	Hertz
Required	Yes, if an `nmea` or `ntrip` endpoint is specified in the configuration
Default	N/A
Description	Nominal rate of incoming GNSS observations. Note that this value is only used for computing NMEA output rates – it does not define the position output rate.
Example	`solution-frequency: 10`

external-data-path

Type	string
Required	No
Default	N/A

Description	Filesystem directory to search for read-only GNSS data.
Example	`external-data-path: /home/swiftnav/starling/`

3.3.3. GNSS

The `gnss` section is mandatory and defines the configuration of the GNSS Engine. This section includes the (mandatory) `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 the weighting model applied to observations. Valid values are `piksi-multi`, `L1L2`, `L1L5`, `android-L1L5`, `teseoV-L1L2`, `teseoV-L1L5`, `ublox-F9` and `ublox-M8L`. The `L1L2` scenario can also be used for L1-only operation.
Example	`type: L1L2`

correction-age-max

Type	unsigned int
Units	seconds
Required	No
Default	30
Description	Number of seconds that the GNSS Engine will attempt to compute a corrected position using the last received correction data. If new correction data is not received within this interval then Starling will report a standalone (i.e. uncorrected) position.
Example:	`correction-age-max: 60`

3.3.3.1. Rover

The `rover` section requires a mandatory endpoint (see Section 3.3.7) to define the source of measurement data.

protocol

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

3.3.3.2. Corrections

The `corrections` section is optional. If it is specified then it requires a mandatory endpoint (see Section 3.3.7) to define the source of correction data.

protocol

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

ntrip-mount-point

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

ntrip-username

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

Example	`ntrip-username: demo202106`

ntrip-password

Type	string
Required	No
Default	N/A
Description	Password for the NTRIP caster. If not specified then no authorisation procedure is attempted when a new NTRIP connection is established.
Example	`ntrip-password: g3m43TVf3`

ntrip-gpgga-period

Type	unsigned int
Required	No
Default	0
Description	Rate at which GGA sentences are sent to NTRIP caster. Only used when the `ntrip` protocol is specified for corrections. This value is a multiplier for the `solution-frequency` period. For example, if `solution-frequency` is set to `10`, then setting `ntrip-gpgga-period` to `2` would result in GGA sentences being sent at 5 Hz. A value of `0` means that GGA sentences will not be sent to the NTRIP caster.
Example	`ntrip-gpgga-period: 10`

3.3.4. Fusion

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

3.3.4.1. IMU Input

The `imu` section is optional and specifies the source of incoming IMU data. It may only be defined when `combined-rover-input` mode is disabled. If this section is specified then it requires a mandatory 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 `ixcom`, `sbp` and `ubx`.
Example	`protocol: sbp`

3.3.4.2. Wheel Odometry Input

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 `ixcom`, `sbp` and `ubx`.
Example	`protocol: sbp`

3.3.4.3. Wheel Odometry Mode

odometry-mode

Type	enum
Required	No
Default	`wheelticks`
Description	Configure usage of wheel odometry input. Valid values are `wheelticks` (use wheeltick inputs only), `speed-sensor` (use speed sensor inputs only), `motion-detection-only` (only use odometry inputs to aid motion detection) and `do-not-use` (ignore all odometry inputs).
Example	`odometry-mode: wheelticks`

3.3.4.4. Antenna Lever Arm

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

x

Type	float
Units	metres

Required	No
Default	0.0
Description	Antenna lever arm X axis distance.
Example	`x: 0.97`

y

Type	float
Units	metres
Required	No
Default	0.0
Description	Antenna lever arm Y axis distance.
Example	`y: 0.11`

z

Type	float
Units	metres
Required	No
Default	0.0
Description	Antenna lever arm Z axis distance.
Example	`z: -0.56`

deviation

Type	float
Units	metres
Required	No
Default	1.0
Description	Standard deviation of antenna lever arm measurement. Must be greater than 0. This value should overestimate the actual expected error.
Example	`deviation: 0.05`

3.3.4.5. Wheel Speed Lever Arm

The optional `wheelspeed-leverarm-meters-sensorframe` section specifies the offset vector from the IMU reference point to the point where the wheel makes contact with the ground. These values should be specified as accurately as possible in order to achieve the highest possible dead reckoning performance.

x

Type	float
Units	metres
Required	No
Default	0.0
Description	Wheel speed lever arm X coordinate.
Example	`x: 0.10`

y

Type	float
Units	metres
Required	No
Default	0.0
Description	Wheel speed lever arm Y coordinate.
Example	`y: 0.25`

z

Type	float
Units	metres
Required	No
Default	0.0
Description	Wheel speed lever arm Z coordinate.
Example	`z: -0.8`

deviation

Type	float
Units	metres
Required	No
Default	1.0
Description	Standard deviation of wheel speed lever arm measurement. Must be greater than 0. This value should overestimate the actual expected error.
Example	`deviation: 0.05`

3.3.4.6. IMU Sensor Rotation

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

x

Type	float
Units	degrees
Required	No
Default	0.0
Description	Rotation around the X axis.
Example	`z: 90.0`

y

Type	float
Units	degrees
Required	No
Default	0.0
Description	Rotation around the Y axis.
Example	`y: 0.0`

z

Type	float

Units	degrees
Required	No
Default	0.0
Description	Rotation around the Z axis.
Example	`z: 180.0`

deviation

Type	float
Units	degrees
Required	No
Default	10.0
Description	Standard deviation of misalignment measurement. Must be greater than 0. This value should overestimate the actual expected error.
Example	`deviation: 3`

3.3.4.7. 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 will be relative to this point once the Fusion Engine is successfully aligned. When the Fusion Engine is not aligned (or the `enable-transform` parameter is set to `false`) then the output position will be at the antenna phase centre.

x

Type	float
Units	metres
Required	No
Default	0.0
Description	Vehicle Reference Point lever arm X coordinate.
Example	`x: 1.0`

y

Type	float

Units	metres
Required	No
Default	0.0
Description	Vehicle Reference Point lever arm Y coordinate.
Example	`y: 1.3`

z

Type	float
Units	metres
Required	No
Default	0.0
Description	Vehicle Reference Point lever arm Z coordinate.
Example	`z: 0.2`

deviation

Type	float
Units	metres
Required	No
Default	0.01
Description	Standard deviation of Vehicle Reference Point coordinates. Must be greater than 0.
Example	`deviation: 0.05`

enable-transform

Type	bool
Required	No
Default	false
Description	Enable transformation of PVAT output to Vehicle Reference Point.
Example	`enable-transform: true`

3.3.4.8. Fast Start

The optional `fast-start` section allows custom thresholds to be specified for activation of Fast Start (see Section 2.2.2 for more details).

ehpe-limit-meters

Type	float
Units	metres
Required	No
Default	10.0
Description	EHPE threshold at shutdown for Fast Start.
Example	`ehpe-limit-meters: 8.0`

speed-limit-meters-per-second

Type	float
Units	metres per second
Required	No
Default	1.0
Description	Speed threshold at shutdown for Fast Start.
Example	`speed-limit-meters-per-second: 1.5`

periodic-storage-interval

Type	duration
Required	No
Default	Disabled (`0s`)
Description	Periodically write Fast Start information to disk at the specified interval. A value of 0 will disable the periodic storage feature. Allowed range is ≥ 1 second.
Example	`periodic-storage-interval: 10s`

3.3.4.9. Fast Start Path

fast-start-path

Type	string

Required	No
Default	N/A
Description	Path of file to use for persistently storing Fusion Engine alignment state across shutdown and restart of Starling. If no path is specified then the Fast Start feature is disabled.
Example	`fast-start-path: /home/swiftnav/starling/fast-start.dat`

3.3.4.10. Solution Rate

solution-rate

Type	double
Unit	Hertz
Required	No
Default	10
Description	Rate at which the Fusion Engine should generate output. Note that the Fusion Engine output rate must not exceed half of the rate of the incoming IMU data. Care should be taken to ensure that the additional CPU load incurred by increasing the Fusion Engine output rate does not exceed the capabilities of the target platform.
Example	`solution-rate: 20`

3.3.4.11. Vehicle Profiles

tuning-profile

Type	enum
Required	No
Default	N/A
Description	Allows internal fusion engine parameters to be configured according to a pre-defined scenario. Valid values are `car-a`, `car-b` and `car-c`.
Example	`tuning-profile: car-c`

3.3.5. Periodic

The `periodic` section is optional and can be used to configure Starling to send output messages containing metadata at fixed intervals.

heartbeat-period

Type	duration
Required	No
Default	`1s`
Description	Send MSG_HEARTBEAT messages with the specified period. This setting is specified as a duration (with value and unit, see Section 3.3.1.1). A zero duration means that MSG_HEARTBEAT messages will not be sent.
Example	`heartbeat-period: 10s`

version-period

Type	duration
Required	No
Default	`60s`
Description	Send Starling version information in MSG_LOG messages with the specified period. This setting is specified as a duration (with value and unit, see Section 3.3.1.1). A zero duration means that version information will not be sent.
Example	`version-period: 10s`

3.3.6. Outputs

The `outputs` section is mandatory and defines the destination(s) for Starling output. The `outputs` section is specified as a sequence of output items, where each output item consists of a mandatory protocol and endpoint (see Section 3.3.7) with an optional name and mask. A maximum of 5 output items may be specified.

This section may also be named `output` for backwards compatibility purposes. In this case only a single output item may be specified.

protocol

Type	enum
Required	Yes
Default	N/A
Description	Protocol for Starling output. Valid values are `nmea` and `sbp`.
Example	`protocol: sbp`

name

Type	string
Required	No
Default	N/A
Description	Free-form text field describing the output. This name will be used in any log messages pertaining to the respective output. Names must be unique. The following names are reserved: • `Rover OBS` • `Corrections` • `NTRIP client` • `PVT engine` • `Fusion engine` • `IMU` • `Wheel odometry` • `Logging` • `Periodic`
Example	`name: SBP output`

mask

Type	enum (or array of enums)
Required	No
Default	N/A
Description	Omit input streams from a given output. Valid values are `gnss` (suppress output from GNSS Engine), `fusion` (suppress output from Fusion Engine), `logging` (suppress MSG_LOG messages), `heartbeat` (suppress MSG_HEARTBEAT) and `input-forwarding` (suppress forwarding of Rover, IMU, NTRIP, Wheel Odometry inputs).
Example	`mask: logging` `mask: [logging,heartbeat,input-forwarding]`

3.3.7. 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 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.

3.3.7.1. Optional Parameters

buffer-size

Type	unsigned int
Units	bytes
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.
Example	`buffer-size: 128`

3.3.7.1.1. SBP Configuration

If the protocol for an endpoint is set to `sbp` then an optional `sbp` section may be used to specify options which are specific to the Swift Binary Protocol.

enabled-messages

Type	list
Required	No
Default	All
Description	List of messages to output through the endpoint. If not specified then all SBP messages are output through the endpoint. Specifying an empty list will result in a validation error.
Example	Enable only MSG_GPS_TIME, MSG_POS_LLH and MSG_VEL_NED messages: `enabled-messages: [258,522,526]`

3.3.7.1.2. NMEA

If the protocol for an endpoint is set to `nmea` then an optional `nmea` section may be used to 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	unsigned int
Required	No
Default	0

Description	Period at which GPGGA sentences will be sent to the output. This value is a multiplier for the `solution-frequency` period. For example, if `solution-frequency` is set to `10`, then setting `gpgga-period` to `2` would result in GGA sentences being sent at 5 Hz.
Example	Output GGA sentences at the `solution-frequency` rate: `gpgga-period: 1`

gpgll-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPGLL sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `gpgll-period` to `2` would result in GLL sentences being sent at 5 Hz.
Example	Output GLL sentences at the `solution-frequency` rate divided by 10: `gpgll-period: 10`

gpgsa-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPGSA sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `gpgsa-period` to `2` would result in GSA sentences being sent at 5 Hz.
Example	Output GSA sentences at the `solution-frequency` rate: `gpgsa-period: 1`

gpgst-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPGST sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`,

	then setting `gpgst-period` to `2` would result in GST sentences being sent at 5 Hz.
Example	Output GST sentences at the `solution-frequency` rate: `gpgst-period: 1`

gpgsv-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPGSV sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `gpgsv-period` to `2` would result in GSV sentences being sent at 5 Hz.
Example	Output GSV sentences at the `solution-frequency` rate divided by 10: `gpgga-period: 10`

gprmc-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPRMC sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `gprmc-period` to `2` would result in RMC sentences being sent at 5 Hz.
Example	Output RMC sentences at the `solution-frequency` rate: `gprmc-period: 1`

gpvtg-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPVTG sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `gpvtg-period` to `2` would result in VTG sentences being sent at 5 Hz.

Example	Output VTG sentences at the `solution-frequency` rate: `gpvtg-period: 1`

gpzda-period

Type	unsigned int
Required	No
Default	0
Description	Period at which GPZDA sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `gpzda-period` to `2` would result in ZDA sentences being sent at 5 Hz.
Example	Output ZDA sentences at the `solution-frequency` rate divided by 10: `gpzda-period: 10`

pubx-period

Type	unsigned int
Required	No
Default	0
Description	Period at which PUBX,00 sentences will be sent to the output. Units are a multiple of `solution-frequency`. For example, if `solution-frequency` is set to `10`, then setting `pubx-period` to `2` would result in PUBX,00 sentences being sent at 5 Hz.
Example	Output PUBX,00 sentences at the `solution-frequency` rate: `pubx-period: 1`

3.3.7.2. 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.

Example	`path: /home/swiftnav/logs/starling-log.sbp`

3.3.7.3. 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.
Example	`identifier: /dev/ttyS0`

baud-rate

Type	unsigned int
Units	bits per second
Required	Yes
Default	N/A
Description	Baud rate for the serial device.
Example	`baud-rate: 115200`

byte-size

Type	int
Required	Yes
Default	N/A
Description	Number of data bits for the serial device. Valid values are `7` and `8`.
Example	`byte-size: 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).
Example	`parity: N`

stop-bits

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

flow-control

Type	bool
Required	No
Default	false
Description	Enable hardware handshaking for the serial device.
Example	`flow-control: false`

3.3.7.4. 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`.
Example	`stdstream-type: stdout`

3.3.7.5. 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.
Example	`host: eu.skylark.swiftnav.com`

port

Type	unsigned int
Required	Yes
Default	N/A
Description	TCP port number on remote server. Range [0 to 65535].
Example	`port: 55555`

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`.
Example	`ip-version: ipv4`

connect-timeout

Type	duration
Required	No
Default	Indefinitely (`0s`)
Description	Maximum time that Starling will wait before giving up on a TCP connection attempt. This setting is specified as a duration (with value and unit, see Section 3.3.1.1). A zero duration means that it will wait indefinitely.
Example	`connect-timeout: 10s`

3.3.7.5.1. 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.
Example	`enable: true`

idle

Type	duration
Required	Yes
Default	N/A
Description	Length of time that a connection remains idle before TCP starts sending keep alive probes. This setting is specified as a duration (with value and unit, see Section 3.3.1.1).
Example	`idle: 10s`

interval

Type	duration
Required	Yes
Default	N/A
Description	Length of time between individual keep alive probes. This setting is specified as a duration (with value and unit, see Section 3.3.1.1).
Example	`interval: 15s`

retries

Type	unsigned int
Required	Yes
Default	N/A

Description	Maximum number of keep alive probes that TCP should send before dropping the connection.
Example	`retries: 100`

3.3.7.6. TCP Server Endpoint

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

port

Type	unsigned int
Required	Yes
Default	N/A
Description	Local port number to listen on for incoming TCP connections.
Example	`port: 55555`

max-conns

Type	unsigned int
Required	Yes
Default	N/A
Description	Maximum number of client connections to support.
Example	`max-conns: 5`

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`.
Example	`ip-version: ipv4`

3.3.7.6.1. 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.
Example	`enable: true`

idle

Type	duration
Required	Yes
Default	N/A
Description	Length of time that a connection remains idle before TCP starts sending keep alive probes. This setting is specified as a duration (with value and unit, see Section 3.3.1.1).
Example	`idle: 20s`

interval

Type	duration
Required	Yes
Default	N/A
Description	Length of time between individual keep alive probes. This setting is specified as a duration (with value and unit, see Section 3.3.1.1).
Example	`interval: 10s`

retries

Type	unsigned int
Required	Yes
Default	N/A
Description	Maximum number of keep alive probes that TCP should send before dropping the connection.
Example	`retries: 100`

3.3.7.7. 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.
Example	`host: 192.168.0.29`

port

Type	unsigned int
Required	Yes
Default	N/A
Description	UDP port number on the remote server.
Example	`port: 55101`

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`.
Example	`ip-version: ipv4`

Appendix A. Supported Messages

A.1. GNSS Engine Inputs

The GNSS Engine requires measurements from a GNSS 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 non-SSR 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 the following 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 ~ 1077	GPS
1084 ~ 1087	GLONASS
1094 ~ 1097	Galileo
1124 ~ 1127	BeiDou
1114 ~ 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 & 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 ~ 1077	GPS
1084 ~ 1087	GLONASS
1094 ~ 1097	Galileo
1124 ~ 1127	BeiDou
1114 ~ 1117	QZSS

One of the following messages is required to provide OSR correction input in SBP format:

Message ID	Description
74	MSG_OBS
1600	MSG_OSR

All of the following messages are required if SSR correction input is provided in SBP format:

Message ID	Description
1501	MSG_SSR_ORBIT_CLOCK
1505	MSG_SSR_CODE_BIASES
1510	MSG_SSR_PHASE_BIASES
1526	MSG_TILE_DEFINITION

1531	MSG_SSR_STEC_CORRECTION
1532	MSG_GRIDED_CORRECTION

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 & 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	MSG_IMU_RAW
2305	MSG_IMU_AUX
2307	MSG_ODOMETRY
2308	MSG_WHEELTICK
65287	MSG_GNSS_TIME_OFFSET

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

A.3. SBP Output Messages

The output messages which can be generated by Starling are listed in the tables below. The output rate of the messages originating from the GNSS Engine depends upon the incoming rate of observations, whereas the Fusion Engine will generate output at the rate specified by the `solution-rate` parameter (see Section 3.3.4.10).

A.3.1. Best Position Output Messages

The *Best Position* messages provide the most accurate position with the highest availability from all available subsystems. These messages originate from the Fusion Engine whenever it is enabled. If the Fusion Engine is not enabled then these messages will originate from the GNSS Engine.

Message ID (Decimal)	Message ID (Hex)	Type
258	102	MSG_GPS_TIME
259	103	MSG_UTC_TIME
521	209	MSG_POS_ECEF
522	20A	MSG_POS_LLH
525	20D	MSG_VEL_ECEF
526	20E	MSG_VEL_NED
529	211	MSG_POS_LLH_COV
530	212	MSG_VEL_NED_COV
532	214	MSG_POS_ECEF_COV
533	215	MSG_VEL_ECEF_COV
545	221	MSG_ORIENT_EULER
65283	FF03	MSG_INS_STATUS
65286	FF06	MSG_INS_UPDATES
65290	FF0A	MSG_GROUP_META
65294	FF0E	MSG_SOLN_META

A.3.2. GNSS-Only Output Messages

The following messages are produced by the GNSS Engine.

Message ID (Decimal)	Message ID (Hex)	Type
260	104	MSG_GPS_TIME_GNSS
261	105	MSG_UTC_TIME GNSS
520	208	MSG_DOPS
523	20B	MSG_BASELINE_ECEF
524	20C	MSG_BASELINE_NED

528	210	MSG_AGE_CORRECTIONS
534	216	MSG_PROTECTION_LEVEL
553	229	MSG_POS_ECEF_GNSS
554	22A	MSG_POS_LLH_GNSS
557	22D	MSG_VEL_ECEF_GNSS
558	22E	MSG_VEL_NED_GNSS
561	231	MSG_POS_LLH_COV_GNSS
562	232	MSG_VEL_NED_COV_GNSS
564	234	MSG_POS_ECEF_COV_GNSS
565	235	MSG_VEL_ECEF_COV_GNSS
65282	FF02	MSG_DGNSS_STATUS

A.3.3. Additional Status Output Messages

The messages below may originate from any subsystem and be sent with any sender ID.

Message ID (Decimal)	**Message ID (Hex)**	**Type**
1025	401	MSG_LOG
65534	FFFE	MSG_STATUS_REPORT
65535	FFFF	MSG_HEARTBEAT

A.3.4. Sender IDs

The SBP Sender ID field is used to identify the source of data. The following values are used:

Sender ID	**Data Source**
0	Forwarded GNSS Corrections
789	Fusion Engine
4096	GNSS Engine

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.

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. All standard messages are sent with the `GP` Talker ID.

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
PUBX,00	Lat/Lon Position Data (u-blox Proprietary)
RMC	Recommended Minimum Specific GNSS Data
VTG	Course Over Ground & Ground Speed
ZDA	Time & Date

See [NMEA] for further information about the NMEA 0183 protocol and the contents of these sentences. Additional information regarding the PUBX,00 sentence can be found in [F9P].

Appendix B. Example YAML Configuration File

The following YAML file will configure Starling as follows:

- Receive L1 and L2 GNSS measurements in RTCM v3 format from 192.168.1.101, port 52302 at 1 Hz
- Apply the `teseoV-L1L2` weighting model to incoming GNSS measurements
- Receive OSR corrections in RTCM v3 format via NTRIP from the European Skylark endpoint (`eu.skylark.swiftnav.com`, port 2101)
- Send GGA sentences to Skylark at 1 Hz
- Read IMU data from `/dev/imu` in SBP format
- Use a lever arm of (0.25 m, 0.939 m, 1.14 m) from the IMU to the antenna phase centre
- Rotate the raw IMU data by 90 degrees around Z followed by a 180 degree rotation around X
- Output PVAT data in SBP format via a TCP server listening on port 55555
- Output GGA, GSV, RMC, VTG and ZDA NMEA sentences via a TCP server listening on port 55556

```yaml
---
name: Example Starling configuration file
solution-frequency: 1
gnss:
  type: teseoV-L1L2
  rover:
    protocol: rtcm
    type: tcp-client
    host: 192.168.1.101
    port: 52302
  corrections:
    protocol: ntrip
    type: tcp-client
    host: eu.skylark.swiftnav.com
    port: 2101
    ntrip-username: test
    ntrip-password: test
    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
outputs:
  # SBP output on port 55555
  - protocol: sbp
    type: tcp-server
```

```
    port: 55555
    max-conns: 10
    keep-alive:
      enable: true
      idle: 1m
      interval: 10s
      retries: 6
  # NMEA output on port 55556
  - protocol: nmea
    type: tcp-server
    port: 55556
    max-conns: 5
    keep-alive:
      enable: true
      idle: 1m
      interval: 10s
      retries: 6
    nmea:
      gpgga-period: 1
      gpgsv-period: 1
      gprmc-period: 1
      gpvtg-period: 1
      gpzda-period: 10
```

Appendix C. Starling Support Files

One or more of the following files may be required by Starling depending upon the specific usage scenario:

File	Example Name	Description
Starling executable	`starling-v1.4.0-x86_64`	Starling binary executable compiled for the relevant target platform.
YAML configuration	`starling.yaml`	Configuration file containing user-specified settings. File location is specified using the `--config` command line parameter.
Antenna database	`igs14.atx`	Phase centre corrections for satellite and receiver antennas. File location is specified using the `external-data-path` parameter in the YAML file. Only required when SSR corrections are in use.
License configuration	`starling-guard.json`	Supporting file for the Starling license manager. File name must exactly match `starling-guard.json`.
Activation key	`activation-key.txt`	File containing the activation key for Starling (a 16 digit number in the form XXXX-XXXX-XXXX-XXXX). File location is specified using the `--activation_key` command line parameter.
License data	`license.lic`	File containing license information following license activation. File location is specified using the `--license` command line parameter.

Appendix D. Reference Frames

D.1. Vehicle Reference Frame

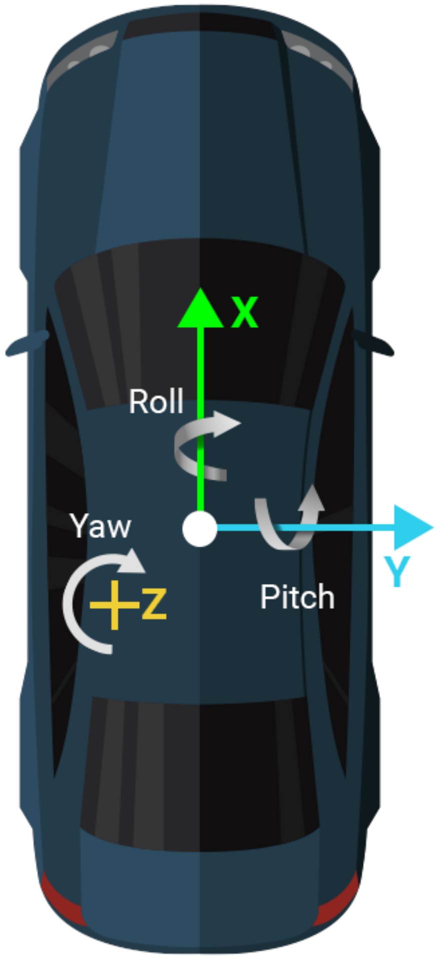

Figure 4: Vehicle Reference Frame

The diagram above shows the *Vehicle Reference Frame* used by Starling, namely:

- X axis pointing in direction of travel
- Y axis pointing right (when facing in the direction of travel)
- Z axis pointing down (when facing in the direction of travel)

All rotations are clockwise when viewed from the origin along the positive direction of the corresponding axis.

D.2. Sensor Reference Frame

The *Sensor Frame* is the reference frame of the IMU as defined by the manufacturer. For example, the image below shows the Sensor Frame for the STMicroelectronics ASM330LHH (as described in [ASM]).

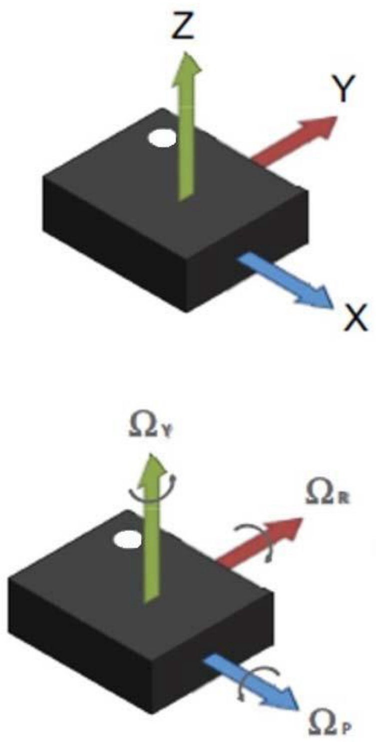

Figure 5: Sensor Reference Frame

Appendix E. Common Orientations

This table can be used to determine the Starling rotation settings for most common orientations. Angles are in degrees.

X: Forward **Y: Left** **Z: Up** Z: 0.0 Y: 0.0 X: 180.0	**X: Right** **Y: Forward** **Z: Up** Z: 90.0 Y: 0.0 X: 180.0	**X: Backward** **Y: Right** **Z: Up** Z: 180.0 Y: 0.0 X: 180.0	**X: Left** **Y: Backward** **Z: Up** Z: -90.0 Y: 0.0 X: 180.0
X: Forward **Y: Right** **Z: Down** Z: 0.0 Y: 0.0 X: 0.0	**X: Right** **Y: Backward** **Z: Down** Z: -90.0 Y: 0.0 X: 0.0	**X: Backward** **Y: Left** **Z: Down** Z: 180.0 Y: 0.0 X: 0.0	**X: Left** **Y: Forward** **Z: Down** Z: 90.0 Y: 0.0 X: 0.0
X: Forward **Y: Up** **Z: Right** Z: 0.0 Y: 0.0 X: 90.0	**X: Right** **Y: Up** **Z: Backward** Z: 0.0 Y: 90.0 X: 90.0	**X: Backward** **Y: Up** **Z: Left** Z: 180.0 Y: 0.0 X: -90.0	**X: Left** **Y: Up** **Z: Forward** Z: 0.0 Y: -90.0 X: 90.0
X: Forward **Y: Down** **Z: Left** Z: 0.0 Y: 0.0 X: -90.0	**X: Right** **Y: Down** **Z: Forward** Z: 0.0 Y: -90.0 X: -90.0	**X: Backward** **Y: Down** **Z: Right** Z: 180.0 Y: 0.0 X: 90.0	**X: Left** **Y: Down** **Z: Backward** Z: 0.0 Y: 90.0 X: -90.0
X: Up **Y: Left** **Z: Backward** Z: 0.0 Y: 90.0 X:-180.0	**X: Up** **Y: Forward** **Z: Left** Z: 90.0 Y: 0.0 X: -90.0	**X: Up** **Y: Right** **Z: Forward** Z: 0.0 Y: -90.0 X: 0.0	**X: Up** **Y: Backward** **Z: Right** Z: -90.0 Y: 0.0 X: 90.0
X: Down **Y: Left** **Z: Forward** Z: 0.0 Y: -90.0 X: 180.0	**X: Down** **Y: Forward** **Z: Right** Z: 90.0 Y: 0.0 X: 90.0	**X: Down** **Y: Right** **Z: Backward** Z: 0.0 Y: 90.0 X: 0.0	**X: Down** **Y: Backward** **Z: Left** Z: -90.0 Y: 0.0 X: -90.0