Logging Data from the Console

Contents

Types of Logs

The Piksi Console has the ability to generate six types of log files. When you are running the console, with a Piksi connected via USB, by default, the baseline, position, and velocity observations are logged on your computer in a simple comma separated value (CSV) file format. Clicking the Record buttons in the Observations Tab will log the raw GPS observations on your computer in Receiver Independent Exchange Format (RINEX) 2.10 (ftp://igscb.jpl.nasa.gov/igscb/data/format/rinex210.txt) format for the Rover or the Base, depending on which button you click. If you add an argument when you run the Console, you can generate a full log of the communication between Piksi and the Console. Thus, you may generate the following log files via the Console.

Generated Automatically

The Console generates three log files automatically, all encoded in comma separated value (CSV) format:

- **Baseline**: RTK vectors, generated whenever you connect Piksi to the console and a Float or Fixed RTK solution is achieved;
- **Position**: standard GPS position, generated whenever you connect Piksi to the console; and
- **Velocity**: standard GPS-derived velocities, generated whenever you connect Piksi to the console.

Via the Console

Users can generate RINEX files by clicking the 'Record' buttons in the Observations Tab:

- **Rover**: detailed RTK data, generated whenever you click 'Record' on the Rover in the Observations Tab; and
- **Base**: detailed RTK data, generated whenever you click 'Record' on the Base in the Observations Tab.

Via the Command Line

Users can generate the full communication between Pikis and the Console as Swift Binary Protocol by adding a command line argument when they start the console:

- **Swift Binary Protocol**: detailed Swift Binary Protocol data, encoded in JavaScript Object Notation (JSON) (http://json.org/) format.

Initiating Logging

Via the Console

Once you have started the console:

- When it achieves a **single point GPS solution,** the Piksi Console automatically generates *position* and *velocity* logs.
- When it achieves a **Float RTK GPS solution,** the Piksi Console automatically generates *baseline* logs.
- When you press **Record in the Observations Tab** for either the rover or the base station, you can initiate detailed RINEX logs for observatons.

Via the Command Line

In order to initiate more detailed logging of serialized SBP, you need to pass a command line argument when you start the console. These arguments vary by operating system. On all operating systems, the "-o" PATH command line options can also optionally be specified. If a directory is passed after the -o option, json logfiles will be placed in this directory and named according to the system time. If a filename is passed after the -o option, this filename will be used for logging.

Linux

1. Pass the --log command line argument to the console.py python script:

```
$ python console.py --log
```

OSX

1. Open the terminal application.

2. Start the piksi console with the --log command line argument:

```
$ open -n /Applications/Piksi\ Console.app --args --log
```

Windows

1. Browse to the console.exe binary in the program files folder
2. Create a shortcut to this executable
3. Right click on the shortcut and choose properties
4. In the “Target:” box, add the command line argument -l (for log), as shown below in **Figure 1**.
5. In the compatibility tab, choose the “run as administrator” to allow the console to write into the program files folder, as shown below in **Figure 2**.

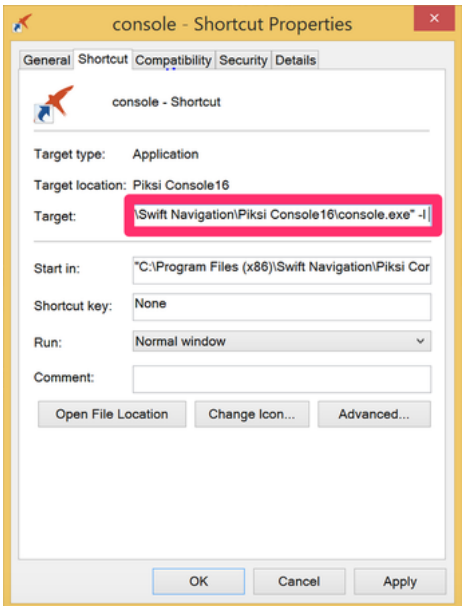


Figure 1. Command Line Argument

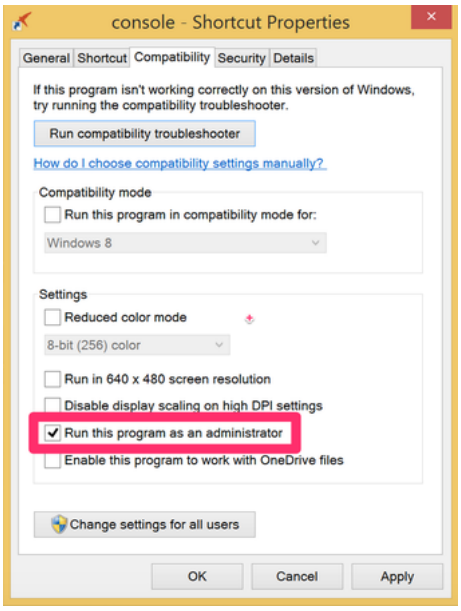


Figure 2. Run as Administrator

Log Locations

Linux

In Linux, log files are stored in the *piksi_tools* directory.

OS X

On OS X, log files are stored within the Piksi Console package contents. To access the files:

- Navigate to the */Applications* folder;
- If you are using finder, right click on *Piksi Console* and select "Show Package Contents"
- Navigate to the */Contents/MacOS* folder

The log files should be the most recently generated files in this folder.

Windows

On Windows, log files are stored within the application folder. Note that users will generally have to have administrator privileges on Windows in order for the log file generation to function correctly.

File Names

CSV Files

The time in this file name is the time that the Piksi first went into RTK mode (Fixed or Float).

> *{name}*_log_*%Y%m%d-%H%M%S*.CSV

RINEX Files

The time in this file name is the time that Record button was pressed in Piksi.

> *{name}-%Y%m%d-%H%M%S*.OBS

SBP Files

The time in this file name is the time that the console was started.

> serial-link-*%Y%m%d-%H%M%S*.log.json

Options

- Options for *{name}* for CSV files are: baseline, velocity, position
- Options for *{name}* for RINEX file are: Rover, Base
- *%Y* is the four digit year.
- *%m* is the two digit month.
- *%d* is the two digit day.
- *%H* is the two digit hour (military time).
- *%M* is the two digit minute.
- *%S* is the two digit second.

CSV files may be opened as text files or with a spreadsheet application. OBS and JSON files may be opened as text files.

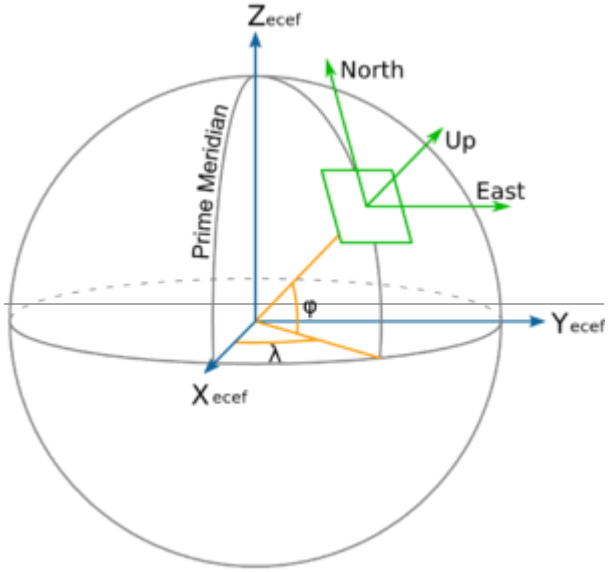
Log Descriptions

Baseline

The baseline file gives a local NED (north, east, down) coordinate representation of the vector between the base and the rover, as a CSV text file. The definition of NED is:

- **North**: The X-axis (denoted by Xn) points toward the ellipsoid north (geodetic north).
- **East**: The Y-axis (denoted by Yn) points toward the ellipsoid east (geodetic east).
- **Down**: The Z-axis (denoted by Zn) points downward along the ellipsoid normal.

The diagram to the right illustrates this representation (in green). Note that, in this diagram, the green Z-axis points up. However, the Z-axis output by Piksi points down, in order to comply with the right hand rule (http://en.wikipedia.org/wiki/Right-hand_rule). This simply means that the Z-axis Piksi output is generally a negative number: the orientation is down, so negative values note the distance from the base to the rover, in cases where the rover is above it.



If you inspect the Baseline Log file, the first few lines will look something like this...

```
2015-05-20



13:07:52.599720,-0.6840,3.8750,1.7820,4.3196,6,0x00,960
2015-05-20 13:07:52.699720,-0.6900,3.8780,1.7770,4.3212,6,0x00,960
2015-05-20 13:07:52.799720,-0.6870,3.8770,1.7850,4.3231,6,0x00,960
```

This baseline file is a CSV version of the SBP message called MSG_BASELINE_NED. This table gives an overview of the format of the baseline file, by column.

Column	Description	Example	Comments
1	Time	2015-05-20 13:07:52.599720	Full GPS time (http://www.leapsecond.com/java/gpsclock.htm) and date, down to the microsecond in this format: %Y%m%d-%H%M%S.
2	North	-0.6840	The north vector between the base and rover, in meters.
3	East	3.8750	The east vector between the base and rover, in meters.
4	Down	1.7820	The down vector between the base and rover, in meters. Note that this is the negative of the up vector.
5	Distance	4.3196	The total length of the NED vector between the base and rover, in meters.
6	Number of Satellites	6	The number of satellites that Piksi was tracking when this observation was recorded.
7	Flags	0x00	Describes which mode the RTK algorithm is in: 0x00 = float mode, 0x01 = fixed mode
8	Hypotheses	960	The number of fixed integer hypotheses that Piksi is considering at the moment.

Position

Standard GPS (single point precision) position location data. If you inspect it, the first few lines will look something like this...

```
2015-05-20 13:07:52.484720,49.8769365203,12.3375256482,566.3836,6,0
2015-05-20 13:07:52.599720,49.8769312853,12.3375093868,565.7801,6,0
2015-05-20 13:07:52.699720,49.8769305995,12.3375114272,565.7496,6,0
```

This position file is a CSV version of the SBP message called MSG_POS_LLH. This table gives an overview of the format of the position file, by column.

Column	Description	Example	Comments
1	Time	2015-05-20 13:07:37.700003	Full GPS time (http://www.leapsecond.com/java/gpsclock.htm) and date, down to the microsecond in this format: %Y%m%d-%H%M%S.
2	Latitude	49.8769348989	Latitude in decimal degrees (WGS84).
2	Longitude	12.3375303721	Longitude in decimal degrees (WGS84).
3	Height	561.7785	Height above (WGS84) ellipsoid in meters.
4	Satellites	6	The number of satellites that Piksi used in solution.
5	Flags	0x00	The current state of Piksi's RTK algorithm: 0x00 = Single Point Position, 0x01 = Fixed RTK, 0x02 = Float RTK

Velocity

The velocity file gives the NED velocities in meters/second as a CSV text. If you inspect it, the first few lines will look like this...

```
2015-05-20 13:07:52.485290,0.033000,0.024000,-0.035000,0.040804,6
2015-05-20 13:07:52.600000,0.023000,-0.012000,0.010000,0.025942,6
2015-05-20 13:07:52.700000,-0.055000,0.029000,-0.030000,0.062177,6
```

This velocity file is a CSV version of the SBP message called MSG_VEL_NED. This table gives an overview of the format of the velocity file, by column.

Column	Description	Example	Comments
1	Time	2015-05-20 13:07:37.700003	Full GPS time (http://www.leapsecond.com/java/gpsclock.htm) and date, down to the microsecond in this format: %Y%m%d-%H%M%S.
2	North Velocity	0.033000	The north vector velocity in meters per second.
2	East Velocity	0.024000	The east vector velocity in meters per second.
3	Down Velocity	-0.035000	The down vector velocity in meters per second. Note that this is the negative of the up vector velocity.
4	Total Velocity	0.040804	The north vector velocity in meters per second.
6	Satellites	6	The number of satellites that Piksi is tracking.

Rover and Base

The Rover and Base files are identically formatted. They provide detailed GPS data about each Piksi, encoded as text in RINEX 2.10 (ftp://igscb.jpl.nasa.gov/igscb/data/format/rinex210.txt) format, often used for post-processing GPS data.

Swift Binary Protocol

This file represents the Swift Binary Protocol, encoded as JSON (http://json.org), with encapsulated binary. If you inspect it, the first few lines will look like this...

```
{"timestamp": 1432839080, "data": {"sender": 1789, "msg_type": 21, "prn": 24, "cf":
749.2676391601562...
{"timestamp": 1432839080, "data": {"sender": 0, "msg_type": 69, "header": {"n_obs":
16, "t": {"wn": 1846....
{"timestamp": 1432839080, "data": {"sender": 1789, "msg_type": 21, "prn": 2, "cf":
-1248.779296875...
```

The Swift Binary Protocol is fully defined here. The file that is logged represents the full communication stream between Piksi and the Console via the serial port.
