Blockstream Satellite

User Guide (v0.4.6)

Blockstream

July 12, 2023

Contents

1 Overview

The Blockstream Satellite network broadcasts the Bitcoin blockchain worldwide 24/7 for free, protecting against network interruptions and providing areas without reliable internet connection with the

opportunity to use Bitcoin. You can join this network by running your own Blockstream Satellite receiver node. This document guides you through all the hardware options, software components, and assembly instructions.

In summary, the process requires the five steps below:

1. Check your coverage at our Coverage Map.
2. Get the required hardware, such as a ready-to-use Satellite Kit.
3. Use the Blockstream Satellite command-line interface (CLI) to handle all the required software installations and configurations.
4. Align your satellite dish appropriately to receive the Blockstream Satellite signal.
5. Run the Bitcoin Satellite and Satellite API applications.

When checking the coverage, ensure your view of the satellite has no obstacles, such as trees or buildings. You can find the target area in the sky using the antenna pointing angles provided by our coverage map or an augmented reality app such as the Satellite Pointer (available for iOS and Android).

Once you get your receiver node up and running, there is a lot that you can do with it. You can use it as a satellite-connected Bitcoin node offering redundancy and protection from internet failures to connected peers. Alternatively, you can run it as your primary Bitcoin full node, with hybrid connectivity (internet and satellite) or only satellite connectivity. The satellite network broadcasts new blocks, transactions, and the complete block history. Hence, you can synchronize the entire blockchain from scratch using the satellite connection.

You can also send encrypted messages worldwide through the satellite network using our Satellite API while paying for each transmission through the Lightning Network. Moreover, if you run a Lightning node, you can sync it faster through Lightning gossip snapshots sent over satellite. You can even download the Bitcoin source code over satellite and bootstrap the node without touching the internet.

The remainder of this guide covers all the above steps in detail, but a quick reference guide is available if you are already familiar with the process.

If you have purchased a satellite kit, you can follow the kit-specific instructions available on Blockstream's Help Center. Otherwise, we recommend continuing on this guide and proceeding to the next section, which covers the hardware options.

1.1 Support

For additional help, you can join the **#blockstream-satellite** IRC channel on freenode or contact Blockstream Support.

2 Hardware Guide

This section explains the hardware components required to assemble a Blockstream Satellite receiver.

There are three alternatives to collect the required hardware. The first and quickest option is to purchase a ready-to-use Satellite Kit. Alternatively, you can buy the Satellite Kit Components on your own. Lastly, you can opt for a completely DIY setup, where you pick a combination of compatible parts on your own. To do so, you will need to understand the hardware requirements. This section covers the three approaches.

2.1 Satellite Kits

As mentioned earlier, the quickest alternative to gather the required parts for a Blockstream Satellite setup is by purchasing a satellite kit. Check the kits available on the Blockstream Store.

There are two main *satellite kits*:

1. Pro Kit (Standalone Receiver).
2. Satellite Base Station (Sat-IP Receiver).

The Base Station is our go-to receiver choice. Its minimalist design, simplified setup, and high performance will fit most Bitcoin users' needs.

The Blockstream Satellite Pro Kit is available for:

- Users in some areas of the Asia-Pacific region covered only by the Telstar 18V C band satellite (and not covered by the Telstar 18V Ku band beam).
- Users who want to use their own dish antennas.
- Users who wish to use larger dish antennas.

Note the Satellite Base Station is not compatible with the C band. That is, it does not work with the Telstar 18V C band beam covering the Asia-Pacific region. If you are in a location with C band coverage only (i.e., without Ku band coverage), you will need a Pro Kit or a DIY receiver option.

If you have decided to go with a satellite kit and selected the Pro Kit receiver, note you still need a satellite antenna and coaxial cables (not included). Please refer to the requirements for antennas](#satellite-antenna) and coaxial cables. After that, you can proceed to the next section, which explains the [receiver setup.

If you selected the Blockstream Satellite Base Station (again, compatible with Ku band only), you are all set. Proceed to the next section.

Otherwise, you can proceed to assemble a satellite receiver on your own. Aside from the Pro Kit and Satellite Base Station options, you can gather the required components for the Basic Kit (formerly

sold on Blockstream Store) based on a Linux USB Receiver. Refer to the list of Basic Kit components. Alternatively, you can find detailed information in this guide to put together an affordable software-defined radio (SDR) receiver with just under $100.

2.1.1 Satellite Kit Comparison

The following table summarizes the different features offered by each of the satellite receiver options:

	SDR	Basic Kit	Pro Kit	Base Station
Blockstream Kit Available			Yes	Yes
USB Interface	Yes	Yes		
Ethernet Interface			Yes	Yes
Requires LNB Power Supply	Yes			
Support for Universal LNB[1]		Yes	Yes	Yes
Dual-Satellite Capable[2]			Yes	
CPU Utilization[3]	High	Low	None	None
Multiple Host Connections[4]			Yes	Yes
Optional Rack Mountable			Yes	
Compatible with C-band[5]	Yes	Yes	Yes	
Performance[6]	Limited	Excellent	Excellent	Excellent
Budget	Low (< $100)	Medium	High (> $900)	Medium ($500)

[1] Support means that the interface provides a 22 kHz signal for switching the Universal LNB between Ku low and Ku high bands. This feature is required to use Universal LNBs when receiving from the Galaxy 18 or Eutelsat 113 satellites.

[2] The device can receive from two satellites simultaneously in areas with overlapping coverage. This feature enables greater redundancy, higher bitrate, and faster blockchain sync times.

[3] The SDR receiver is implemented in software and runs on the host computer. Hence, it uses the underlying CPU significantly. The Basic Kit receiver uses a dedicated receiver chip and only minimal resources from the host CPU. The Pro Kit and Base Station receivers are entirely standalone.

[4] The receiver can feed the data stream received over satellite to multiple hosts simultaneously over the local network.

[5] As mentioned earlier, C band support is required to receive the Telstar 18V C band beam in the Asia-Pacific region.

[6] The SDR receiver is an excellent option for a budget-limited setup. However, it is expected to have inferior performance due to software limitations.

2.2 Satellite Kit Components

Instead of purchasing a satellite kit, you can buy the individual components on your own. The elements of each kit are summarized in this section.

If you have decided to buy the satellite kit components on your own, you can proceed to the receiver setup once you collect them. Otherwise, if you would still like to learn more about hardware requirements or search for alternative compatible parts, including the affordable SDR receiver option, refer to the DIY hardware requirements.

2.2.1 Blockstream Satellite Basic Kit

This kit is no longer available on the Blockstream Store. It has been replaced by the Satellite Base Station.

Components:

- TBS 5927 DVB-S2 Tuner (see the note below).
- GEOSATpro UL1PLL Universal Ku Band PLL LNB.
- Titanium C1-PLL C Band PLL LNB.
- Titanium CS1 Conical Scalar Kit.
- Ku Band LNB mounting bracket.
- C Band LNB mounting bracket.
- 32 cm flat, bendable flat coaxial TV extension cable used to pass through window and door frames.

> Note 1: The TBS 5520SE tuner model is now also supported. You can replace the TBS 5927 with the 5520SE model.

> Note 2: the kit includes two LNBs so that it works in the C and Ku bands. However, you can purchase only the LNB required for your location. If you are in a C band location (Telstar 18V C Asia-Pacific region), you will need the Titanium C1-PLL LNB or similar, as well as the optional CS1 Conical Scalar Kit or similar if using an offset dish. Otherwise (in all other regions), you can purchase the GEOSATpro UL1PLL Universal LNB or similar.

The above list does not include the antenna nor the required coaxial cables. Please refer to the requirements for antennas and coaxial cables.

2.2.2 Blockstream Satellite Pro Kit

Available on Blockstream Store.

Components:

- Novra S400 Professional DVB-S2 Receiver.
- GEOSATpro UL1PLL Universal Ku Band PLL LNB.
- Titanium C1-PLL C Band PLL LNB.
- Titanium CS1 Conical Scalar Kit.
- Ku Band LNB mounting bracket.
- C Band LNB mounting bracket.
- 32 cm flat, bendable flat coaxial TV extension cable used to pass through window and door frames.

> Note: the kit includes two LNBs so that it works in the C and Ku bands. However, you can purchase only the LNB required for your location. If you are in a C band location (Telstar 18V C Asia-Pacific region), you will need the Titanium C1-PLL LNB or similar, as well as the optional CS1 Conical Scalar Kit or similar if using an offset dish. Otherwise (in all other regions), you can purchase the GEOSATpro UL1PLL Universal LNB or similar.

Note the kit does not include the antenna nor the required coaxial cables. Please refer to the requirements for antennas and coaxial cables.

2.2.3 Blockstream Satellite Base Station

Available on Blockstream Store.

Components:

- Selfsat>IP22 all-in-one Sat-IP flat-panel antenna.
- Power over Ethernet injector.
- Ethernet Cat5e cable.

2.3 DIY Hardware Requirements

This section explains the requirements to assemble a satellite receiver setup entirely on your own. Nevertheless, note this process can be time-consuming. If you prefer a faster solution, check out the available Satellite Kits.

2.3.1 Supported Receiver Options

The receiver is the device or software application that processes the incoming satellite signal and decodes the data stream from it. There are four supported types of receivers. For each of them, specific hardware components are required.

The receiver options are summarized below:

- **Software-defined Radio (SDR)**: this receiver is entirely implemented in software. You will need an SDR interface connected to your PC (typically via USB). The SDR interface collects and feeds signal samples to the receiver application/software running on your PC. The application, in turn, decodes and outputs the data stream to be fed into Bitcoin Satellite. This is the most affordable option among the three, as it works with inexpensive RTL-SDR USB dongles. However, it is also the option expected to present the most limited performance and reliability among the three. Moreover, this option is CPU-intensive since the receiver application will run in the CPU.
- **Linux USB Receiver**: in this setup, the demodulation is entirely carried out in hardware, in the external receiver device connected to your host via USB. Hence, unlike the SDR receiver, the Linux USB receiver is not CPU-intensive. With this option, you will need to install specific drivers and Linux DVB-S2 apps on the host to configure the external receiver and get the data from it. Overall, this option is expected to perform greatly and with negligible CPU usage. On the other hand, it can require a time-consuming initial setup due to the driver installation. You can try the driver installation on your intended host before committing to this receiver option.
- **Standalone Receiver**: this is also a hardware-based setup, with the difference that it is entirely independent of the host PC. It connects to the PC through the network and can potentially feed multiple PCs concurrently. This is also expected to be a great option in terms of performance. Besides, this is the only option capable of dual-satellite reception using a single receiver device.
- **Sat-IP Receiver**: this is another hardware-based and standalone receiver option. The difference is that it is based on an all-in-one antenna with a built-in DVB-S2 receiver and integrated LNB (see the Satellite Base Station). It is referred to as a Sat-IP receiver because it runs a Sat-IP server, to which your host will connect as a client. Overall, this option offers the easiest configuration and the most minimalist setup, given that it requires a single component (the all-in-one antenna). However, note it does not work with the Telstar 18V C band satellite covering the Asia-Pacific region.

For further insights, refer to the satellite receiver comparison table presented earlier.

Once you pick your preferred receiver option, you should gather all of its required components. The following section explains the common elements required for all types of receivers. Then, the subsequent section covers the specific parts for each receiver option.

2.3.2 Common Required Components

In addition to the DVB-S2 receiver, you will need an antenna and a low-noise block downconverter (LNB) to receive the satellite signal. Furthermore, you will need cables to connect them to each other.

The antenna and LNB components are required in all setups other than the all-in-one Satellite Base Station. The latter, in contrast, consists of an antenna with an integrated receiver and LNB, all in one

device.

Refer to the following requirements to select the appropriate antenna, LNB, and cables.

2.3.2.1 Satellite Antenna The most widely available antenna option is the regular satellite TV dish with a conventional parabolic reflector.

Blockstream Satellite is designed to work with small dishes, with diameters as low as 45 cm in Ku band and 60 cm in C band. However, we recommend checking our Link Analyzer tool for more specific antenna recommendations. After inputting your coordinates, you can obtain the list of supported antennas for your location. Then, you can feel free to pick the smallest supported option. However, if you are looking to obtain the best performance, a larger antenna is always preferable.

Other than size, the only additional requirement is that the antenna works with the frequency band that suits your coverage region. You can always use antennas designed for higher frequencies. For example, an antenna designed for the Ka band will work for the Ku and C bands, as it is designed for higher frequencies than those used by Blockstream Satellite. However, a C-band antenna will not work in the Ku band, as it is designed for lower frequencies. For further information regarding frequency bands, please refer to the frequency guide.

An alternative to conventional satellite dishes is a flat panel antenna, which is generally more compact and stylish. A recommended flat panel model is the Selfsat H50D, which was previously available on Blockstream Store before being replaced by the all-in-one Satellite Base Station. The Selfsat H50D includes the LNB internally, and so there is no need to purchase an LNB (nor an LNB bracket) when using it. However, note that this model has limited compatibility. It is an excellent option for:

1. **Linux USB** and **Standalone Receivers** in any Ku band region.
2. **SDR** receivers in Ku low band regions (Telstar 11N Africa, Telstar 11N Europe, and Telstar 18V Ku).

In contrast, the Selfsat H50D flat panel is **not** compatible with receivers (of any type) in the Telstar 18V C (C Band) region. It only works in Ku band.

The flat panel requires an extra 22 kHz generator to work with **SDR** receivers in the Ku high band regions (Galaxy 18 and Eutelsat 113). This antenna includes a built-in Universal LNB, which, as explained later, requires a 22 kHz tone generated by the receiver specifically for the reception in the Ku high band (i.e., the band used by G18 and E113). Refer to further information and a solution for 22 kHz generation on an SDR setup in the Universal LNB section.

2.3.2.2 LNB When choosing a low-noise block downconverter (LNB), the most relevant parameters are the following:

- Frequency range

- Polarization
- LO Stability

In a nutshell, you are advised to use a PLL LNB with linear polarization and LO stability within `+- 250 kHz` or less. Also, the LNB should be suitable for the frequency of the signal covering your location.

Regarding **frequency range**, you must verify that the input frequency range of the LNB encompasses the frequency of the Blockstream Satellite signal in your coverage area. For example, if you are located in North America and covered by the Eutelsat 113 satellite, the downlink frequency of interest is 12066.9 GHz. In this case, an LNB that operates from 11.7 GHz to 12.2 GHz would work. In contrast, an LNB that operates from 10.7 GHz to 11.7 GHz would **not** work. You can check the signal frequencies of each region in the frequency guide.

Regarding **polarization**, an LNB with **Linear Polarization** is required. While most Ku band LNBs are linearly polarized, some popular satellite TV services use circular polarization. A circularly polarized LNB will **not** work with Blockstream Satellite.

If an LNB is described to feature horizontal or vertical polarization, then it is linear. In contrast, if an LNB is described as Right-Hand or Left-Hand Circular Polarized (RHCP or LHCP), then it is circular and will **not** work with Blockstream Satellite.

Regarding **LO Stability**, a stability specification of `<= +/- 250 kHz` is preferable for better performance. Most LNBs will have a local oscillator (LO) stability parameter referred to as "LO stability," or metrics such as "LO accuracy" and "LO drift." These are usually specified in +/- XX Hz, kHz, or MHz. An LNB that relies on a phase-locked loop (PLL) frequency reference is typically more accurate and stable. Hence, we advise looking for a PLL LNB instead of a traditional dielectric oscillator (DRO) LNB.

If you would like (or you need) to use a less stable LNB, it can also be used. The disadvantage is that it will likely degrade your setup's reliability and performance (for example, increase the bit error rate).

A widely available LNB option is the so-called **Universal Ku band LNB**. However, please note that if you are using an SDR-based setup, a **Universal LNB** may pose extra difficulties. Please refer to the explanation regarding Universal LNBs. This limitation **does not** apply when using the Linux USB or Standalone receiver options.

Besides, another parameter of interest is the so-called F/D ratio. This parameter refers to the ratio between the parabolic reflector's focal length and its diameter. As such, it is a parameter of the dish, not the LNB. Nevertheless, the LNB should be designed for an F/D ratio compatible with the reflector.

For example, an offset dish (the most common dish type for Ku band) typically has an F/D ratio from 0.5 to 0.7. In contrast, a regular "front-fed" parabolic dish typically has an F/D in the 0.3 to 0.4 range. In any case, check the F/D specifications of your dish and make sure to use a compatible LNB. If necessary, attach a flat or conical scalar ring to change the F/D characteristics of the LNB.

Lastly, to avoid confusion, please note that *LNBF* and *LNB* often refer to the same thing. You can find further information later in this guide.

2.3.2.3 LNB Mounting Bracket The antenna dish likely comes with a mounting bracket, but you will need one designed to accept a generic LNB. Also, it is good to have a flexible bracket to facilitate the LNB rotation and control of its polarization angle. Although all mounting brackets do allow rotation, some can be limited in the rotation range.

Such mounting brackets attach to the antenna's feed arm and have a circular ring that will accept a generic LNB.

2.3.2.4 Coaxial Cables You will need a coaxial cable to connect the LNB to the receiver or, in the case of the SDR-based setup, to connect the LNB to the power supply. The most popular and recommended type of coaxial cable is the RG6 cable.

2.3.3 Setup-Specific Components

This section summarizes the additional components required for each type of setup, according to the receiver choice.

2.3.3.1 Software-defined Radio (SDR) Setup

Component	Requirement
SDR interface	RTL-SDR dongle model RTL2832U w/ TCXO
LNB Power Supply	SWM Power Supply
SMA Cable	Male to Male
SMA to F adapter	SMA Female, F Male

The supported **SDR interface** is the RTL-SDR, which is a low-cost USB dongle. More specifically, an RTL-SDR of model RTL2832U.

There are two specifications to observe when purchasing an RTL-SDR:

1. Oscillator
2. Tuner

We recommend using an RTL-SDR with a temperature-controlled crystal oscillator (TCXO), as the TCXO has better frequency stability than a conventional crystal oscillator (XO). A few models in the market feature a TCXO with frequency accuracy within 0.5 ppm to 1.0 ppm, which are good choices.

Regarding the tuner, the choice depends on the satellite covering your location. The two recommended tuners are the R820T2 and the E4000. The table below summarizes which tuner to pick for each satellite:

Satellite	RTL-SDR Tuner
Galaxy 18	R820T2
Eutelsat 113	R820T2
Telstar 11N Africa	E4000
Telstar 11N Europe	E4000
Telstar 18V Ku	E4000
Telstar 18V C	R820T2

This tuner recommendation has to do with the L-band frequencies expected in each region, as summarized in the frequency guide. The E4000 tuner is recommended for the areas where the L-band frequency is close to the maximum tuning range of the R820T2 tuner (1766 MHz).

Hence, for example, if you are going to receive from Galaxy 18, you should get an RTL-SDR RTL2832U with tuner R820T2 and TCXO. In contrast, if you are going to receive from Telstar 11N Africa, you should get an RTL-SDR RTL2832U with tuner E4000 and TCXO. Note that the RTL-SDR models featuring the E4000 tuner are marketed as **extended tuning range RTL-SDR** or **XTR RTL-SDR**.

The next component of the SDR receiver setup is the **LNB Power Supply** (or Power Inserter). This component supplies a DC voltage to the LNB via the coaxial cable, typically of 13 VDC or 18 VDC. On a non-SDR setup, the receiver itself can provide power to the LNB, so there is no need for an external power supply. In contrast, this is not possible with an SDR-based setup using the SDR interface alone. Hence, an external supply is required.

The type of power supply that is easy to find in the market is named "Single Wire Multiswitch" (SWM) power supply. You can look for an SWM power inserter and use it as illustrated below. The **non-powered** port of an SWM power supply is labeled "Signal to IRD," which means signal to integrated receiver/decoder. This is the port that should be connected to the SDR interface. The **powered** port, in turn, is labeled "Signal to SWM." This is the port that should be connected to the LNB.

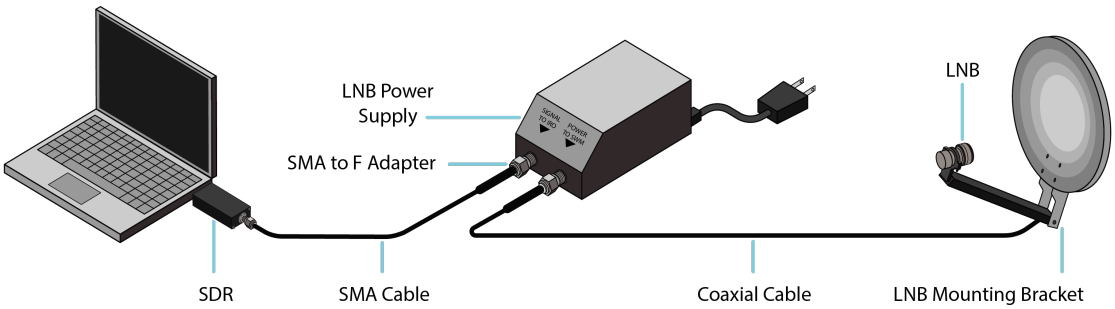


Figure 1: SDR receiver connections

IMPORTANT: Do **NOT** connect the powered port of the LNB power supply to the SDR interface. Permanent damage may occur to your SDR and/or your computer.

Also, please check the power/voltage requirement of your LNB and ensure that your power supply matches. It should be noted that some LNBs known as **dual-polarization LNBs** accept two DC voltage levels. Such LNBs use the supplied voltage to switch between vertical and horizontal polarization. A supplied DC voltage of +18 VDC sets the LNB to horizontal polarization, whereas +13 VDC sets the LNB to vertical polarization. Please keep this in mind when rotating the LNB for a specific polarization angle during the antenna pointing stage.

Further notes:

- **Alternative SDR interfaces**: the RTL-SDR is the supported SDR interface and the most popular among Blockstream Satellite users. Nevertheless, other SDR boards/interfaces can be used with minor tweaks, such as USRPs. The SDR interface must support L-band frequencies within the 1 GHz to 2 GHz range and sampling rates of 2 Msps (mega samples per second) or higher.
- **Connectors**: not every RTL-SDR has the same interface connector. Some use the SMA connector, while some others use MCX. Be sure to order the correct cable and adapters to make the necessary connections. In the above table, we assume the RTL-SDR has SMA female connector, while the power supply has two F female connectors. In this case, you need an SMA male-to-male cable and an SMA female to F male adapter to connect the RTL-SDR to the **non-powered** port ("Signal to IRD") of the SWM power supply.

2.3.3.2 Linux USB Receiver Setup

The only specific component in this setup is the external USB-based DVB-S2 receiver. The supported receivers are the TBS 5927 and the TBS 5520SE models, which connect to the Linux PC via a USB2.0 connection. In this case, the LNB is connected directly to the *LNB IN* interface of the TBS receiver. Also, the TBS device is powered up either directly by the host via USB (TBS 5520SE) or with a dedicated power supply that comes with it (TBS 5927).

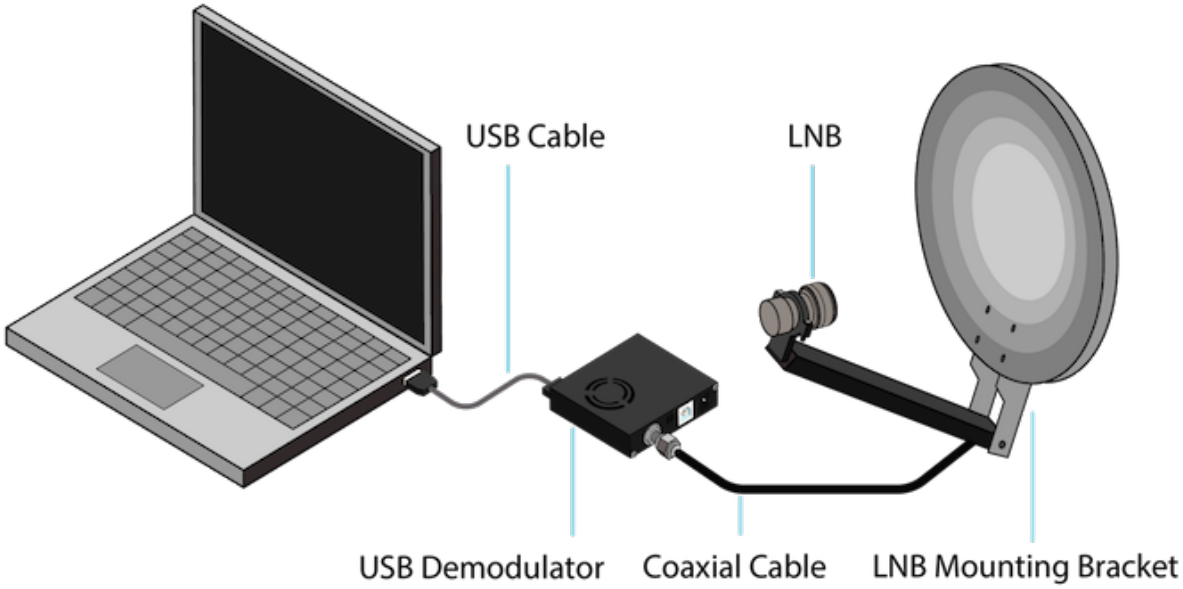


Figure 2: Linux USB receiver connections

> NOTE: although the TBS 5927 and 5520SE receivers offer Windows support, we currently do not support Windows as an operating system for a Blockstream Satellite setup.

2.3.3.3 Standalone Receiver Setup In this setup, a standalone DVB-S2 receiver connects to the host PC (or to the network) through an Ethernet cable. The standalone receiver that is currently supported is the Novra S400 PRO DVB satellite Receiver. Other than this receiver, you only need an Ethernet Cable.

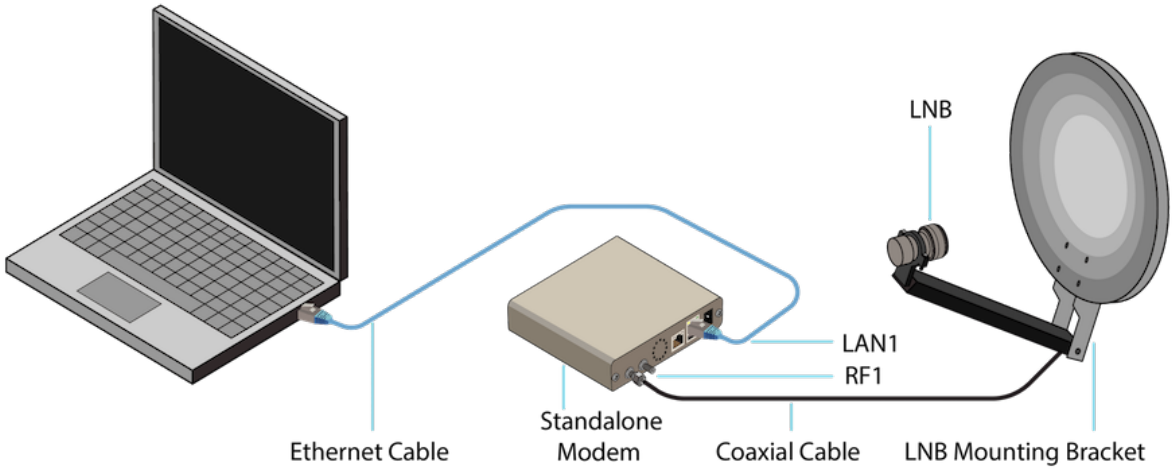


Figure 3: Standalone receiver connections

2.3.3.4 Sat-IP Receiver Setup The Sat-IP receiver that is currently supported is the all-in-one Self-sat>IP22 Sat-IP flat-panel antenna. It requires an Ethernet cable (Cat5e or superior) to connect to a switch/router or directly to your host. Furthermore, it requires Power over Ethernet (PoE). Hence, you will need a PoE injector if the switch/router or network adapter connecting to the Sat-IP antenna does not support PoE.

2.4 Further Notes

2.4.1 Universal LNB

A Universal LNB, also known as Universal Ku band LNB, is an LNB that supports both Ku low and Ku high bands. With such an LNB, the receiver becomes responsible for switching the Ku sub-band as needed. More specifically, the receiver sends a 22 kHz tone to the LNB when tuning to a Ku high band carrier. Otherwise, when tuning to a Ku low band carrier, the receiver simply does not send any tone. The absence of a 22 kHz tone configures the LNB to its default sub-band, the Ku low sub-band.

An important limitation applies to the SDR setup when using a Universal LNB. Note the SDR setup described in this guide is receiver-only. Hence, it cannot generate a 22 kHz tone to configure the Universal LNB. Consequently, a Universal LNB connected to an SDR receiver operates in Ku low band only. Thus, we recommend using this type of LNB only within Ku low band regions, i.e., within the areas covered by Telstar 11N Africa, Telstar 11N Europe, or Telstar 18V Ku.

Meanwhile, in contrast to an SDR setup, both Linux USB and Standalone receiver options support the generation of 22 kHz. Hence, it is perfectly acceptable to use a Universal LNB in any Ku band region when using one of these receivers.

Besides, there are workarounds to switch the sub-band of a Universal LNB even with an SDR setup. For instance, you can place a 22 kHz tone generator inline between the LNB and the power inserter. In this case, the tone generator will use power from the inserter while delivering the tone directly to the LNB. Such tone generators can be found in the market as pure generators. Alternatively, you can get a satellite finder device containing the 22 kHz generation functionality.

If choosing a satellite finder, it is essential to note that the finder must be one that can be used inline between the power supply and the LNB. In other words, it must be one with two connectors, one for signal input (from the LNB) and the other for output (towards the power inserter). Some finders contain a single connector, as they are not intended to be used inline. Furthermore, be aware that most finders do not include a 22 kHz generator. You must pick a satellite finder that supports the generation of a 22 kHz tone.

2.4.2 LNB vs. LNBF

The feedhorn is the horn antenna that attaches to the LNB. It collects the signals reflected by the satellite dish and feeds them into the LNB towards the receiver. The acronym LNBF stands for "LNB with feedhorn" and refers to the LNB that already contains an integrated feedhorn. This is the most typical type of LNB nowadays. Hence, almost always, the term LNB already refers to an LNBF implicitly. To avoid confusion, we advise looking for an LNBF.

3 Software Requirements

The Blockstream Satellite command-line interface (CLI) is required to configure and run your receiver. You can install it by executing the following command on the terminal:

```
sudo pip3 install blocksat-cli
```

> NOTE:
>
> 1. The CLI requires Python 3 and the above command requires Python3's package installer (pip3).
> 2. If you prefer to install the CLI on your local user directory (without `sudo`) instead of installing it globally (with `sudo`), make sure to add `~/.local/bin/` to your path (e.g., with `export PATH=$PATH:$HOME/.local/bin/`).

Next, run the configuration helper:

```
blocksat-cli cfg
```

Then, check out the instructions for your setup by running:

```
blocksat-cli instructions
```

Within the set of instructions, one of the required steps is the installation of software dependencies, which is accomplished by the following command:

```
blocksat-cli deps install
```

Once the above command completes successfully, you can move on to the next section, which discusses the receiver and host configuration.

4 Novra S400 Receiver

The Novra S400 is a standalone receiver, which receives a satellite signal and outputs IP packets to one or multiple hosts listening to it in the local network. Hence, you will need to configure both the S400 and the host(s).

4.1 Connections

The Novra S400 can be connected as follows:

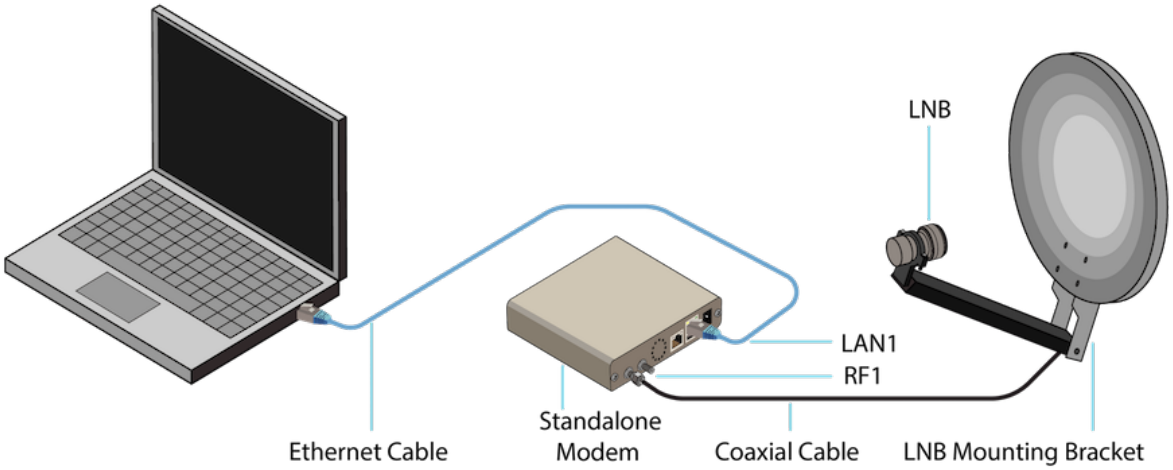


Figure 4: Novra S400 receiver connections

- Connect the LNB directly to interface RF1 of the S400 using a coaxial cable (an RG6 cable is recommended).
- Connect the S400's LAN1 interface to your computer or network.

4.2 Network Connection

Next, make sure the S400 receiver is reachable by the host.

First, configure your host's network interface to the same subnet as the S400. By default, the S400 is configured with IP address `192.168.1.2` on interface LAN1 and `192.168.2.2` on LAN2. Hence, if you connect to LAN1, make sure your host's network interface has IP address `192.168.1.x`, where "x" could be any number higher than 2. For example, you could configure your host's network interface with IP address 192.168.1.3.

After that, open the browser and access `192.168.1.2` (or `192.168.2.2` if connected to LAN 2). The web management console should open up successfully.

4.3 Software Requirements

Next, install all software pre-requisites on your host. Run:

```
blocksat-cli deps install
```

> Note: this command supports the `apt`, `dnf` and `yum` package managers.

4.4 Receiver and Host Configuration

Now, configure the S400 receiver and the host by running:

```
blocksat-cli standalone cfg
```

This command configures the signal parameters on the S400 receiver and network settings on the host. If you would like to review the changes that will be made to the host before applying them, first run the command in dry-run mode:

```
blocksat-cli standalone cfg --dry-run
```

Also, if you would like to apply the receiver configurations manually, refer to the instructions for configuration via the web UI.

> Note: the above commands assume the S400 has IP address `192.168.1.2` (the default address). You can specify another address on all `blocksat-cli standalone` commands by running with option `--addr [address]`.

4.5 Monitoring

> Added in blocksat-cli version v0.3.0. Refer to the upgrade instructions if necessary.

Finally, you can monitor your receiver by running:

```
blocksat-cli standalone monitor
```

4.6 Next Steps

At this point, if your dish is already correctly pointed, you should be able to start receiving data on Bitcoin Satellite. Please follow the instructions for Bitcoin Satellite configuration. If your antenna is not aligned yet, please follow the antenna alignment section.

4.7 Further Information

4.7.1 Dual-satellite Setup

On a dual-satellite setup, you need to configure the two RF interfaces of the S400. After configuring the first interface as instructed earlier, configure the second by running:

```
blocksat-cli --cfg rx2 standalone --demod 2 cfg --rx-only
```

Refer to further information on the dual-satellite setup guide.

4.7.2 S400 Configuration via the Web UI

1. Open the browser and access the IP address of the S400. By default, the address is `192.168.1.2` if connected to LAN 1 or `192.168.2.2` if connected to LAN 2.
2. Log in as admin on the top right of the page.
   - Default password: "password"
3. Check the signal parameters that apply to your setup. Run the following command and use the results in the next step.

```
blocksat-cli cfg show
```

4. Go to `Interfaces` > `RF1` and configure as follows:
   - DVB Mode: "DVB-S2"

Next, configure the LNB parameters at the bottom of the page:

- LNB Power On: Enable.
- L.O. Frequency: your LNB's local oscillator (LO) frequency in MHz.
- Polarization: the signal polarization (horizontal or vertical).
- Band (Tone): "Low/Off" by default. Set to "High/On" only when using a Universal LNB and receiving in Ku High Band (when receiving from Galaxy 18 or Eutelsat 113).
- Long Line Compensation: Disabled.

After that, configure the DVB Signal Parameters:

- Carrier Freq.: frequency in MHz of the satellite covering your location.
- LBand: leave it with the auto-filled value.[7]
- Symbol Rate: 1.0 MBaud.

[7] The L-band frequency is automatically defined once you inform both the **L.O. Frequency** and the **Carrier Freq**.

- MODCOD: AUTO.
- Gold Code: 0.
- Input Stream ID: 0.

Then, click **Apply**.

5. Verify that the S400 is locked.

- Check the "RF 1 Lock" indicator at the top of the page or the status LED in the S400's front panel. It should be green (locked) if your antenna is already pointed correctly. If not, you can work on the antenna pointing afterward.

6. Go to Services > RF1:

Scroll to "Manage MPE/ULE PIDs":

- Enter 32 on "New PID" and click "Add".
- Apply.

** Optional configurations:

- If you prefer to use another IP address on LAN1 or LAN2, go to Interfaces > Data (LAN1) or Interfaces > M&C (LAN2) and configure the IP addresses. Note LAN 1 is the interface that will deliver the data packets received over satellite, whereas LAN2 is optional and exclusively for management.
- If configuring the second RF interface on the S400 for a dual-satellite setup, on *step 2*, go to `Interfaces > RF2` instead of `RF1`. Correspondingly, on step 3, check the "RF 2 Lock" indicator, and on step 4, go to `Services > RF2`.

5 TBS Linux USB Receiver

The TBS 5927 and TBS 5520SE devices are USB-based DVB-S2 receivers. They receive the satellite signal fed via a coaxial interface and output data to the host over USB. They are also configured directly over USB, and the host is responsible for setting such configurations using specific Linux tools.

The instructions that follow prepare the host for driving the TBS receiver.

5.1 Hardware Connections

The TBS 5927/5520SE should be connected as follows:

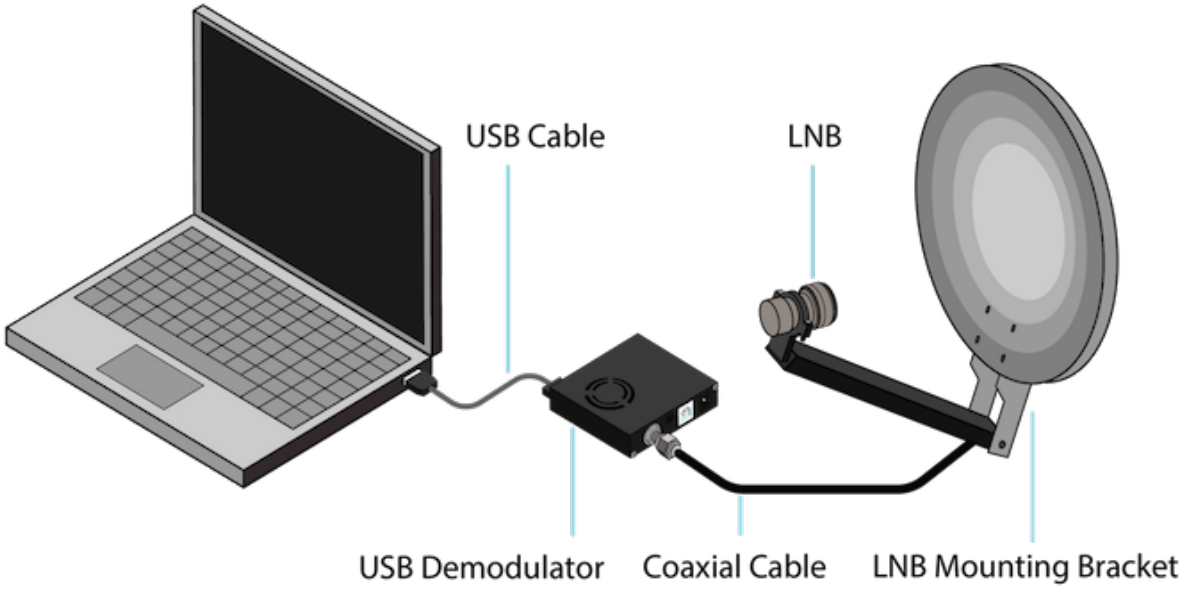


Figure 5: TBS5927 connections

- Connect the LNB directly to "LNB IN" interface of the TBS 5927/5520SE using a coaxial cable (an RG6 cable is recommended).
- Connect the TBS's USB2.0 interface to your computer.
- Power up the TBS device. For the TBS 5927 model, connect the 12V DC power supply. For the TBS 5520SE, connect both male connectors of the dual-male USB Y cable to your host.

5.2 TBS Drivers

Next, you will need to install specific device drivers to use the TBS 5927 or 5520SE receivers. These are installed by rebuilding and rewriting the Linux Media drivers. Hence, if you are not setting up a dedicated machine to host the TBS receiver, it would be safer and **recommended** to use a virtual machine (VM) as the receiver host so that the drivers can be installed directly on the VM instead of your main machine.

To install the drivers, run the following command:

```
1 blocksat-cli deps tbs-drivers
```

Once the script completes the installation, reboot the virtual machine.

5.3 Setup Configuration Helper

Some configurations depend on your specific setup. To obtain detailed instructions, please run the configuration helper and the instructions menu as follows:

```
blocksat-cli cfg
blocksat-cli instructions
```

5.4 Software Requirements

Now, install all software pre-requisites (in the virtual machine) by running:

```
blocksat-cli deps install
```

> Note: this command supports the `apt`, `dnf` and `yum` package managers. For other package managers, refer to the manual installation instructions and adapt package names accordingly.

5.5 Configure the Host

Next, you need to create and configure a network interface to output the IP traffic received via the TBS 5927/5520SE unit. You can do so by running the following command:

```
blocksat-cli usb config
```

If you would like to review the changes before applying them, first run the command in dry-run mode:

```
blocksat-cli usb config --dry-run
```

Note this command will define an arbitrary IP address to the interface. If you would like to set a specific IP address instead, for example, to avoid address conflicts, use the command-line argument `--ip`.

Furthermore, note that this configuration is not persistent across reboots. After a reboot, you need to run `blocksat-cli usb config` again.

5.6 Launch

Finally, start the receiver by running:

```
blocksat-cli usb launch
```

5.7 Next Steps

At this point, if your antenna is already correctly pointed, you should be able to start receiving data on Bitcoin Satellite. Please follow the instructions for Bitcoin Satellite configuration. If your antenna is not pointed yet, refer to the antenna alignment section.

5.8 Further Information

5.8.1 Docker

A Docker image is available for running the Linux USB receiver host on a container. Please refer to the instructions in the Docker section.

5.8.2 Useful Resources

- TBS 5927 Datasheet.
- TBS 5520SE Datasheet.
- TBS Drivers Wiki.

5.8.3 Install Binary Packages Manually

The following instructions are an alternative to the automatic installation via the CLI (with command `blocksat-cli deps install`).

On Ubuntu/Debian:

```
sudo apt apt update
sudo apt install python3 iproute2 iptables dvb-apps dvb-tools
```

On Fedora:

```
sudo dnf update
sudo dnf install python3 iproute iptables dvb-apps v4l-utils
```

On Fedora, package `dvb-apps` is not available via the main dnf repository. In this case, you can install it from our repository by running:

```
sudo dnf copr enable blockstream/satellite
sudo dnf install dvb-apps
```

> If command `dnf copr enable` is not available in your system, install package `dnf-plugins-core`.

5.8.4 Building dvb-apps from Source

Alternatively, you can build `dvb-apps` from source by running the following commands:

```
git clone https://github.com/Blockstream/dvb-apps
cd dvb-apps
make
sudo make install
```

6 Selfsat>IP22 Sat-IP Receiver

The Selfsat>IP22 is an all-in-one flat-panel antenna with an integrated DVB-S2 receiver and LNB. It is the basis of the Blockstream Satellite Base Station kit available on the Blockstream Store. This device receives the satellite signal and outputs IP packets to one or more Sat-IP clients listening to it in the local network. This section explains how you can connect to the base station device to receive the Blockstream Satellite traffic.

6.1 Connections

The integrated Sat-IP antenna-receiver can be connected as follows:

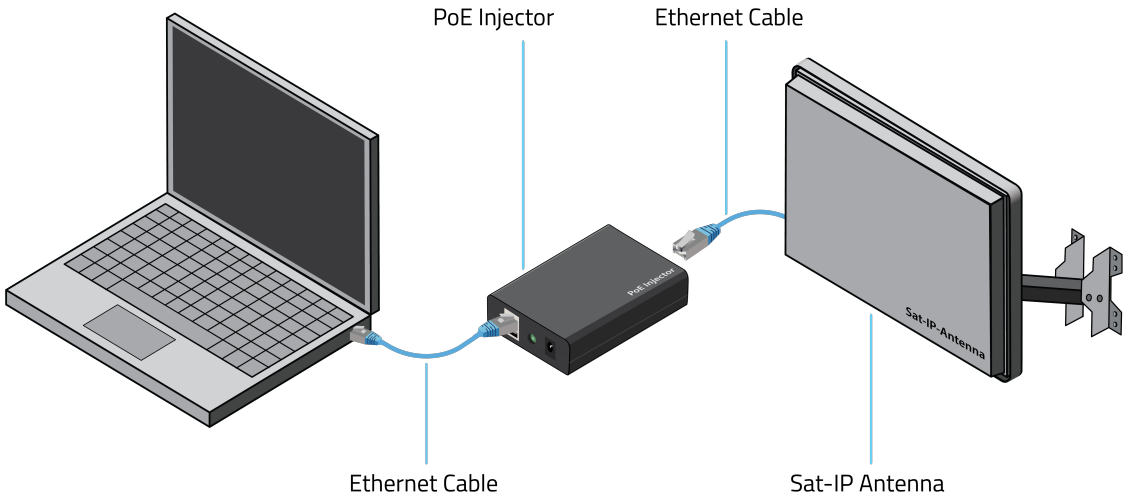


Figure 6: Sat-IP receiver connections

- Connect the Ethernet cable from your switch or computer's network adapter directly to the antenna's Sat>IP port.
- If your switch/adapter does not support Power over Ethernet (PoE), insert a PoE injector in-line between the switch/adapter and the antenna's Sat-IP port. Connect the injector's PoE-enabled

port to the Sat-IP antenna and the non-powered (non-PoE) port to the switch/adapter.

IMPORTANT: If using a PoE injector, make sure you are connecting the correct ports. Permanent damage may occur to your switch or network adapter otherwise.

6.2 Software Requirements

To install the required applications, run:

```
blocksat-cli deps install
```

> Note: this command supports the two most recent releases of Ubuntu LTS, Fedora, CentOS, Debian, and Raspbian. In case you are using another Linux distribution or version, please refer to the compilation instructions.

6.3 Running

You should now be ready to launch the Sat-IP client. You can run it by executing:

```
blocksat-cli sat-ip
```

> Note: the Sat-IP client discovers the server via UPnP. If your network blocks this traffic type, you can specify the Sat-IP server's IP address (i.e., the Satellite Base Station address) directly using option `-a`/`--addr`. Alternatively, see the troubleshooting section.

6.4 Next Steps

At this point, if your antenna is already correctly pointed, you should be able to start receiving data on Bitcoin Satellite. Please follow the instructions for Bitcoin Satellite configuration. If your antenna is not aligned yet, refer to the antenna alignment section.

6.5 Further Information

6.5.1 Software Updates

To stay up-to-date with the Sat-IP client software, run the following to search for updates and install them when available:

```
blocksat-cli deps update
```

6.5.2 Troubleshooting the Server Discovery

When the Sat-IP client returns the error "Could not find a Sat-IP receiver," it is possible that either the network or your host is blocking the UPnP traffic used to auto-discover the base station receiver.

To troubleshoot the problem, you can start by inspecting whether the devices in the network are replying to the SSDP packets sent by your host. To do so, run `tcpdump` as follows:

```
sudo tcpdump -i any port 1900
```

Typically, a reply will come from your local router, but we also need the response from the Sat-IP server (the base station device). If the base station is not replying, you can try changing the SSDP source port to 1900 and recheck the results on tcpdump:

```
blocksat-cli sat-ip --ssdp-src-port 1900
```

If you see the response coming from the base station now, but the Sat-IP client still does not connect to it, your host's firewall might be blocking the response. If you are on Ubuntu (using `ufw`), you can allow SSDP responses by running `sudo ufw allow 1900`. More specifically, it is generally preferable to configure the firewall rule with a reduced scope. For instance, you can enable only the SSDP packets coming specifically from the base station device. If you know the base station's IP address, you can do so by running:

```
sudo ufw allow from [base-station-address] to any port 1900
```

If you don't know the base station's IP address at this point, you can allow SSDP packets coming from any address in the local subnet. For instance, for a 192.168.1.1/24 subnet, run:

```
sudo ufw allow from 192.168.1.1/24 to any port 1900
```

More generally, if you are running another Linux distribution or firewall manager, make sure to allow UDP packets sent to port 1900.

6.5.3 Direct Connection to the Base Station

A typical use case for the base station is connecting it to a switch or router and accessing it from hosts within the same network. However, note it is also possible to connect the host directly to the base station without intermediate switches or routers. In this case, you need to pay attention to two aspects:

1. The network interface that the host shall use when attempting to discover the Sat-IP server automatically.
2. The IP address of the chosen interface.

By default, the host will send the device discovery requests via its default network interface (for instance, a WLAN interface). Meanwhile, you may have the base station connected directly to a secondary Ethernet interface. In this case, the discovery packets would never reach the IP22. To solve the problem, specify the Ethernet interface via the `--ssdp-net-if` option when launching the Sat-IP client.

Regarding the IP address configuration, note the base station typically will not receive an IP address via DHCP when connected directly to your host. Instead, it will fall back to a link-local address in the 169.254.0.0/16 subnet. Hence, to communicate with the base station, you need to configure your host's Ethernet interface with an arbitrary address in the 169.254.0.0/16 subnet. For instance, assuming the Ethernet interface is named `eth0`, run:

```
ip addr add 169.254.100.50/16 dev eth0
```

Finally, run the Sat-IP client as follows while replacing `eth0` with your interface name:

```
blocksat-cli sat-ip --ssdp-net-if eth0
```

6.5.4 Running on Docker

A Docker image is available for running the Sat-IP client on a container. Please refer to the instructions on the Docker section.

6.5.5 Compilation from Source

The Sat-IP setup relies on the TSDuck application. To build and install it from source, run:

```
git clone https://github.com/tsduck/tsduck.git
cd tsduck
build/install-prerequisites.sh
make NOTELETEXT=1 NOSRT=1 NOPCSC=1 NODTAPI=1
sudo make NOTELETEXT=1 NOSRT=1 NOPCSC=1 NODTAPI=1 install
```

> Refer to TSDuck's documentation for further information.

7 SDR Receiver

7.1 Connections

The SDR setup is connected as follows:

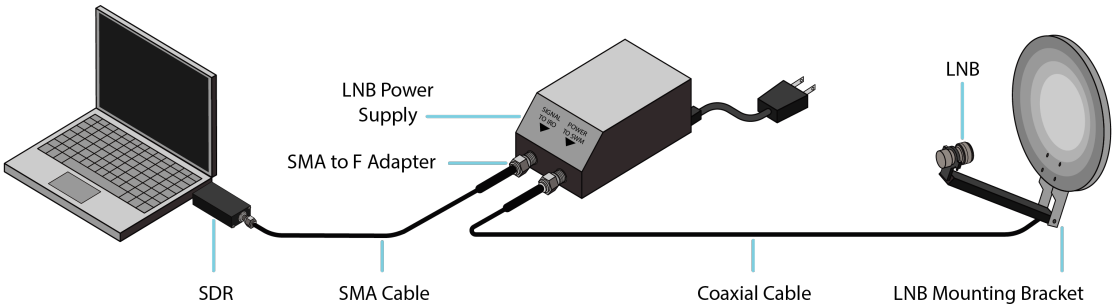


Figure 7: SDR connections

- Connect the RTL-SDR USB dongle to your host PC.
- Connect the **non-powered** port of the power supply (labeled as "Signal to IRD") to the RTL-SDR using an SMA cable and an SMA-to-F adapter.
- Connect the **powered** port (labeled "Signal to SWM") of the power supply to the LNB using a coaxial cable (an RG6 cable is recommended).

IMPORTANT: Do NOT connect the powered port of the power supply to the SDR interface. Permanent damage may occur to your SDR and/or your computer.

7.2 Software Requirements

The SDR-based setup relies on the applications listed below:

- leandvb or gr-dvbs2rx: the software-based DVB-S2 receiver application.
- rtl_sdr: reads samples taken by the RTL-SDR and feeds them into leandvb.
- TSDuck: unpacks the output of leandvb and produces IP packets to be fed to Bitcoin Satellite.
- Gqrx: useful for spectrum visualization during antenna pointing.

To install them all at once, run the following:

```
1 blocksat-cli deps install
```

> Note: this command supports the two most recent releases of Ubuntu LTS, Fedora, CentOS, Debian, and Raspbian. In case you are using another Linux distribution or version, please refer to the manual compilation and installation instructions.

If you prefer to install all software components manually, please refer to the manual installation section. Also, if you would like to try the alternative receiver application based on the GNU Radio framework, see the gr-dvbs2rx section.

7.3 Configuration

After installing, you can generate the configurations that are needed for gqrx by running:

```
1 blocksat-cli gqrx-conf
```

> Note: this command assumes you are using an RTL-SDR dongle.

7.4 Running

You should now be ready to launch the SDR receiver. You can run it by executing:

```
1 blocksat-cli sdr
```

More specifically, as thoroughly explained in the antenna alignment section, you might want to run with specific gain and de-rotation parameters that are suitable to your setup, like so:

```
1 blocksat-cli sdr -g [gain] --derotate [freq_offset]
```

where `[gain]` and `[freq_offset]` should be substituted by the appropriate values.

7.5 Next Steps

At this point, if your antenna is already correctly pointed, you should be able to start receiving data on Bitcoin Satellite. Please follow the instructions for Bitcoin Satellite configuration. If your antenna is not aligned yet, refer to the antenna alignment section.

7.6 Further Information

7.6.1 Software Updates

To update the SDR software to the most recent releases, run:

```
1 blocksat-cli deps update
```

7.6.2 Docker

A Docker image is available for running the SDR host on a container. Please refer to the instructions in the Docker section.

7.6.3 gr-dvbs2rx Receiver

An alternative software-defined DVB-S2 receiver implementation named gr-dvbs2rx is available on the CLI starting from version 0.4.5. This application is based on the GNU Radio framework for software-defined radio, and it is supported on Fedora 36 and Ubuntu 22.04 or later versions only.

The CLI installs gr-dvbs2rx automatically when available. However, if you are running an existing setup configured with a CLI version preceding 0.4.5, please rerun the installation command:

```
1 blocksat-cli deps install
```

Then, relaunch the SDR receiver with the regular command below. The CLI will choose the gr-dvbs2rx application by default when available. However, you can always toggle the implementation using option `--impl`.

```
1 blocksat-cli sdr
```

7.6.4 Manual Installation of SDR Software

If you do not wish to rely on the automatic installation handled by command `blocksat-cli deps install`, you can install all applications manually.

First, enable our repository for binary packages. On Ubuntu/Debian, run:

```
1 add-apt-repository ppa:blockstream/satellite
2 apt-get update
```

> If command `add-apt-repository` is not available in your system, install package `software-properties-common`.

On Fedora, run:

```
1 dnf copr enable blockstream/satellite
```

> If command `copr enable` is not available in your system, install package `dnf-plugins-core`.

Finally, install the applications:

```
1 sudo apt install rtl-sdr leandvb tsduck gqrx-sdr
```

or

```
1 sudo dnf install rtl-sdr leandvb tsduck gqrx
```

7.6.5 Manual Compilation of SDR Software

If `leandvb` and (or) `tsduck` are not available as binary packages in your distribution, you can build and install them from source.

7.6.5.1 Leandvb from source To build leandvb from source, first install the dependencies:

```
apt install git make g++ libx11-dev
```

or

```
dnf install git make g++ libX11-devel
```

Then, run:

```
git clone --recursive https://github.com/Blockstream/leansdr.git
cd leansdr/src/apps
make
sudo install leandvb /usr/bin
```

Next, build and install `ldpc_tool`:

```
cd ../../LDPC/
make CXX=g++ ldpc_tool
sudo install ldpc_tool /usr/bin
```

7.6.5.2 TSDuck from source To build and install TSDuck from source, run:

```
git clone https://github.com/tsduck/tsduck.git
cd tsduck
build/install-prerequisites.sh
make NOTELETEXT=1 NOSRT=1 NOPCSC=1 NOCURL=1 NODTAPI=1
sudo make NOTELETEXT=1 NOSRT=1 NOPCSC=1 NOCURL=1 NODTAPI=1 install
```

8 Antenna Pointing

Aligning a satellite antenna is a precise procedure. Remember that the satellites are over 35,000 km (22,000 mi) away. A tenth of a degree of error will miss the satellite by more than 3500 km. Hence, this is likely the most time-consuming step of the process. This section provides a step-by-step guide for antenna alignment.

8.1 Mount the Antenna

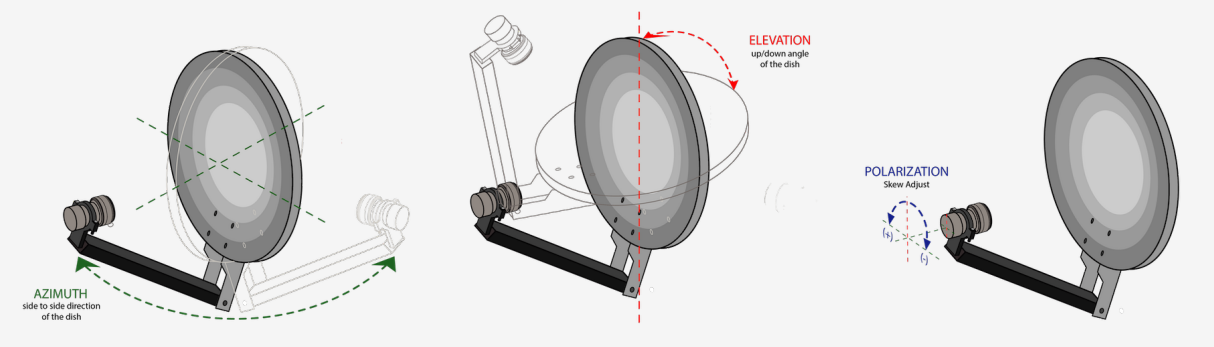
First, you should obtain the pointing angles required for your specific location using our dish alignment tool.

> NOTE: If your location is covered by two satellites and you only want to point to one of them, we recommend checking which of the two has the strongest signal in your location. For that, check the Link Analyzer tool. After inputting your coordinates and obtaining the link analysis, choose the satellite with the highest expected carrier-to-noise ratio (CNR) for your chosen antenna.

After entering your address or latitude/longitude, the tool will give you the following parameters:

- **Azimuth**: the side-to-side angle of your antenna. 0 degrees refers to North, 90 degrees to East, 180 degrees to South, and 270 degrees to West.
- **Elevation**: the up and down adjustment of your antenna. The antenna aiming tool provides the number of degrees above the horizon to which your antenna must point. 0 degrees represents pointing at the horizon, and 90 degrees is pointing straight up.
- **Polarity**: determines the rotation of the LNB. It is the angle of the LNB within the LNB mounting bracket (or holder). Often this is referred to also as the LNB *polarization angle* or *LNB skew*. A positive angle means a clockwise adjustment when looking at the LNB from behind the dish and facing forwards to the satellite in the sky.

The three angles are illustrated below:

Azimuth	Elevation	Polarity
		

Next, visually inspect the direction to which your antenna must point. Use a compass or smartphone app (e.g., Satellite Pointer for Android and iOS) to identify it. Ensure that there are no obstacles (like trees or buildings) between your antenna and the target area in the sky. You must have a clear line of sight to that area of the sky.

IMPORTANT: If using a compass app on a smartphone, make sure to configure the app to display **true north** instead of the **magnetic north**. The azimuth angle provided by our dish alignment tool refers

to true north. Also, if using an ordinary compass or a compass-based satellite finder, make sure to convert the true azimuth obtained from the dish alignment tool into the magnetic azimuth. You can get both the true and magnetic azimuth angles using a tool such as the Dish Pointer app.

Next, install the satellite antenna according to the directions accompanying it, or have it done professionally. If you install it yourself, proceed with the following steps:

1. Certify that the pole on which the dish is mounted is entirely level.
2. Set the elevation of the antenna to the parameter provided by the antenna aiming tool (above). Many antennas will have an elevation scale on the back of the dish that you can use to set the approximate elevation.
3. Set the LNB polarization to the parameter provided by the antenna aiming tool. This involves rotating the LNB. There is typically a polarization rotation scale on the LNB or the LNB mounting bracket.
4. Set the azimuth angle to the value obtained from the aiming tool.

> Note: if using an offset dish antenna (the most common dish type in Ku band), make sure to subtract the dish offset angle from the nominal elevation. You can check the offset angle in the dish specifications. For example, if the nominal elevation angle at your location is 20°, and your offset reflector has an offset angle of 25°, the resulting elevation becomes -5°. In this case, it will appear that the reflector is pointing down to the ground. In reality, however, the dish will be tilted correctly to receive the low-elevation signal at the offset focal point.

At this stage, you can leave the screws that control the azimuth angle slightly loose so that you can adjust the azimuth for pointing. You can do the same for elevation and polarization. Nevertheless, the azimuth is typically easier to sweep as an initial pointing attempt.

8.2 Find the Satellite and Lock the Signal

Assuming that the receiver is configured correctly and connected, your next step is to find the satellite. You will adjust the antenna pointing until the receiver can lock to the Blockstream Satellite signal. Please note that this is likely the most time-consuming part of the setup process, especially when doing it for the first time. As mentioned earlier, a single degree shifted on the dish represents a change of thousands of kilometers over the geosynchronous orbit.

The process will be easier with a laptop than can be watched while moving the antenna. If you cannot watch the computer, you'll need two people: one to move the antenna and one to monitor the computer.

To start, make sure that your receiver is running. Depending on your type of receiver, this step involves one of the following commands:

- For the TBS 5927 or 5520SE USB receivers: `blocksat-cli usb launch`.
- For the Novra S400 Standalone (Pro Kit) receiver: `blocksat-cli standalone monitor` (see the S400's instructions).
- For the Sat-IP receiver (Satellite Base Station): `blocksat-cli sat-ip`.
- For the SDR receiver: `blocksat-cli sdr`.

Next, you should monitor the receiver logs printed on the console. Initially, while the antenna is not pointed correctly, the receiver will be unlocked. In this case, the application will print logs like the following:

```
2020-10-23 14:26:14  Lock = False;
```

In this case, you should try to make adjustments to the antenna pointing. For example, keep the elevation angle fixed and slowly move the antenna side to side (vary the azimuth angle). Alternatively, keep the azimuth fixed and gradually change the elevation. Every time you adjust an angle, wait a few seconds and check if the receiver has found the signal in this position. If not, try another adjustment and so on.

Once the receiver finds the signal, it will lock and print a line with `Lock = True` like the following:

```
2020-10-23 14:32:25  Lock = True; Level = -47.98dBm; SNR = 11.30dB; BER
    = 0.00e+00;
```

From this point on, the application should remain locked.

You should pay special attention to the signal-to-noise ratio (SNR) parameter printed on the console. The higher the SNR, the better. Given that the receiver is already locked, you can infer that the antenna pointing is close to the optimal position. Hence, at this point, you should experiment with gentle adjustments to the pointing angles until you can maximize the SNR. The next section discusses the target SNR levels.

> Note: the Sat-IP receiver will print a *Signal Quality* metric instead of the SNR. Again, higher is better.

Furthermore, you can check that the signal level is within acceptable limits. The LNB amplifies the signal received over satellite and feeds a reasonably high signal level into the receiver. However, the signal experiences attenuation over the coaxial cable, connectors, and adapters. The expected minimum and maximum signal levels are summarized below:

Receiver	Minimum Signal Level	Maximum Signal Level
TBS 5927	-69 dBm	-23 dBm

Receiver	Minimum Signal Level	Maximum Signal Level
Novra S400	-65 dBm	-25 dBm

In the end, note that the antenna pointing procedure is entirely based on the locking indicator printed on the console. Once you find the signal and the receiver locks, the only remaining step is to optimize the SNR.

This approach works for all types of receivers. However, there are helpful receiver-specific instructions for the pointing process, listed below:

- **Novra S400 Standalone receiver**: you can find various receiver status metrics within the receiver's web interface. See the instructions.
- **SDR receiver**: with an SDR receiver, you can visualize the signal spectrum and point the antenna more easily. See the SDR instructions.

Alternatively, you can try to point the antenna using a satellite finder. This approach is generally more helpful for the Linux USB (TBS 5927 or 5520SE), standalone (Novra S400), and Sat-IP receivers. In contrast, for SDR-based receivers, the SDR signal visualization tools are usually better. Refer to the instructions in the satellite finder section.

8.3 Optimize the SNR

After the initial antenna pointing, we recommend experimenting with the pointing until you achieve your maximum SNR. The SNR level should be at least 2.23 dB. However, we recommend trying to reach an SNR of 8 dB or higher in clear sky conditions.

The SNR exceeding the minimum level determines the so-called link margin. For example, if your receiver operates at 8.2 dB, it has a 6 dB margin relative to the minimum SNR of roughly 2.2 dB. This margin means that your receiver can tolerate up to 6 dB signal attenuation in case of bad weather (such as rain).

If your receiver is already locked, try gentle adjustments around the current position and observe the SNR on the console. Stop once you find the pointing that achieves the best SNR.

If you are using a Sat-IP receiver, which does not print the SNR metric, you can optimize the Signal Quality metric instead. Try to achieve 100% quality or as close as possible.

8.4 Next Steps

Well done! You are now ready to run the Bitcoin Satellite application receiving data via the Blockstream Satellite Network. Please refer to the Bitcoin Satellite guide for further instructions.

8.5 Further Information

8.5.1 Novra S400's User Interface

The Novra S400 receiver features a web-based user interface (UI), which provides several receiver metrics.

At the top, the web UI has an *LNB* indicator, which indicates whether the S400 is supplying power to the LNB. Furthermore, it shows whether the S400 is locked. Assuming you have connected the LNB to input RF1, then the **RF1 Lock** indicator will be green when the unit is locked.

Figure 8: Novra S400 receiver searching signal

If the S400 is **not** locked yet, as depicted above (RF1 Lock indicator off), you should adjust the antenna pointing.

Once the S400 finally locks, the *RF1 Lock* indicator looks as follows:

Figure 9: Novra S400 receiver locked

You can also find signal quality and status metrics on page `Interfaces > RF1`, under *RF1 Detailed Status*. For example:

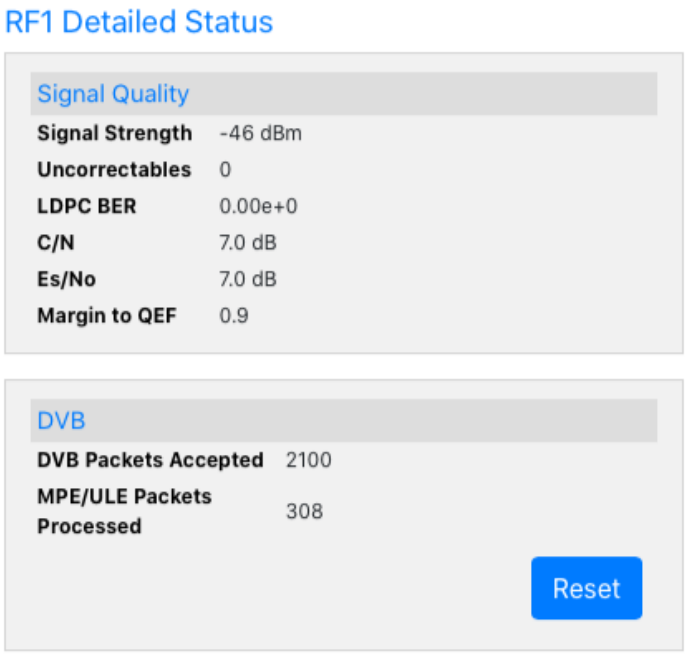


Figure 10: Novra S400 receiver's RF status

Note that the carrier-to-noise ratio (C/N) parameter relates to the SNR parameter that should be optimized during the antenna pointing.

8.5.2 Pointing with an SDR-based Receiver

The SDR-based receiver offers additional visualization tools that are very helpful for antenna pointing. With this receiver, the pointing procedure consists of two steps:

1. Visualization using `gqrx`;
2. Locking using the actual receiver application.

In the first step, you should launch `gqrx` (check the gqrx configuration instructions). Then, click the start icon ("Start DSP Processing") and see if you can recognize the Blockstream Satellite signal. Ideally, you would see a flat level spanning a frequency band (in the horizontal axis) of approximately 1 MHz. Here is an example:

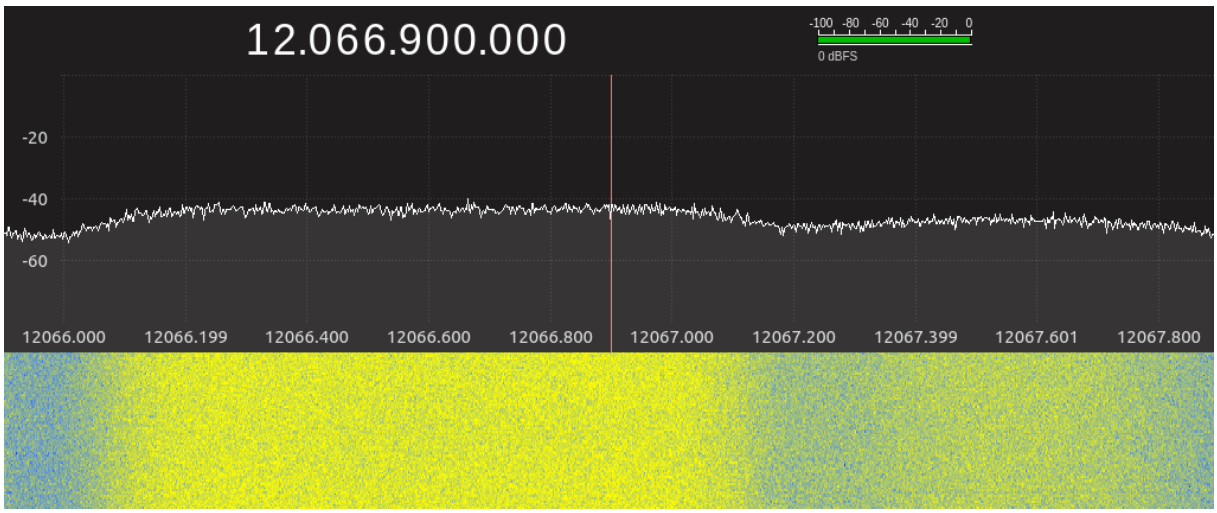


Figure 11: Blockstream Satellite signal visible on Gqrx

With the recommended gqrx configuration, gqrx should be configured to the center frequency of the signal band (in this example, of 12066.9 MHz, where the red line is). However, the observed signal commonly is offset from the nominal center frequency, given that LNBs introduce frequency offset. In the above example, note the signal is around 12066.6 MHz, which means a frequency offset of -300 kHz (to the left relative to the nominal center). If gqrx's center frequency is re-configured to 12066.6 MHz, then we can see the 1 MHz band well centered, like so:

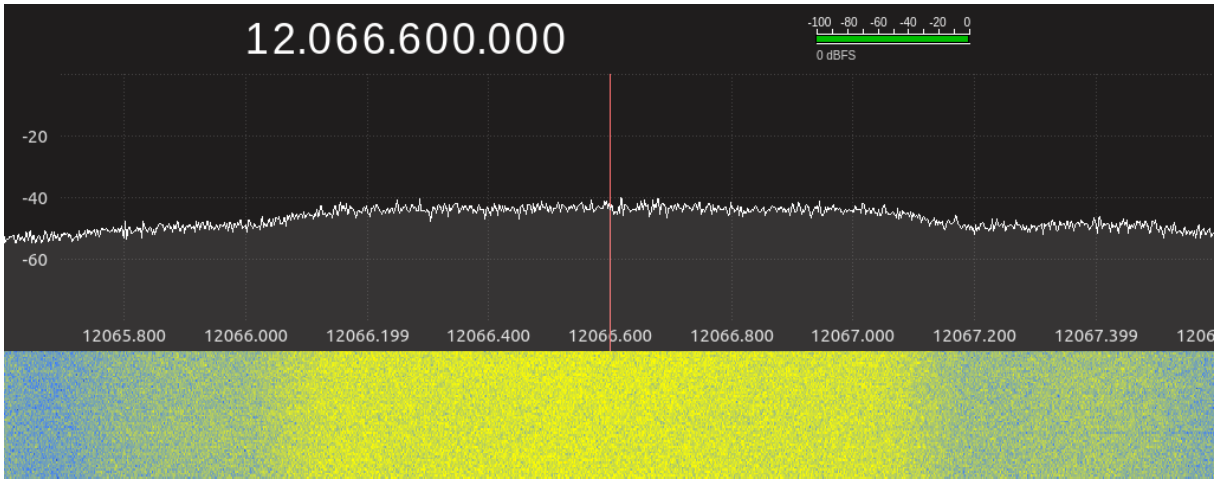


Figure 12: Blockstream Satellite signal centered on Gqrx

If you cannot see the signal on gqrx, you should try to make adjustments to the antenna pointing, as described earlier.

> NOTE:
>
> If you see two similar signal bands near each other, try to identify which one is more likely to be the Blockstream Satellite signal. The correct signal should span a flat level of 1 MHz, with 100 kHz of roll-off on each side. If the two signal bands are close to 1 MHz, please take note of both center frequencies and try both of them in the next steps until you get a lock.
>
> Furthermore, please note that, in some cases, there can be similar signal bands among different (but nearby) satellites. In this case, you need to adjust the pointing until you get a lock.

Once you finally find the signal in gqrx, you can proceed to run the actual SDR receiver application. As explained in the SDR guide, you can start it with:

```
blocksat-cli sdr
```

For pointing, however, it is helpful to run it in GUI mode, as follows:

```
blocksat-cli sdr --gui
```

At this point, before proceeding, we recommend inspecting whether the gain is well configured. Check the preprocessed (iq) plot. If it looks like the one below, with strongly scattered points around the two dimensions, the gain is likely too high. In this case, you can run with a lower gain specified using option `-g`, like so:

```
blocksat-cli sdr -g [gain]
```

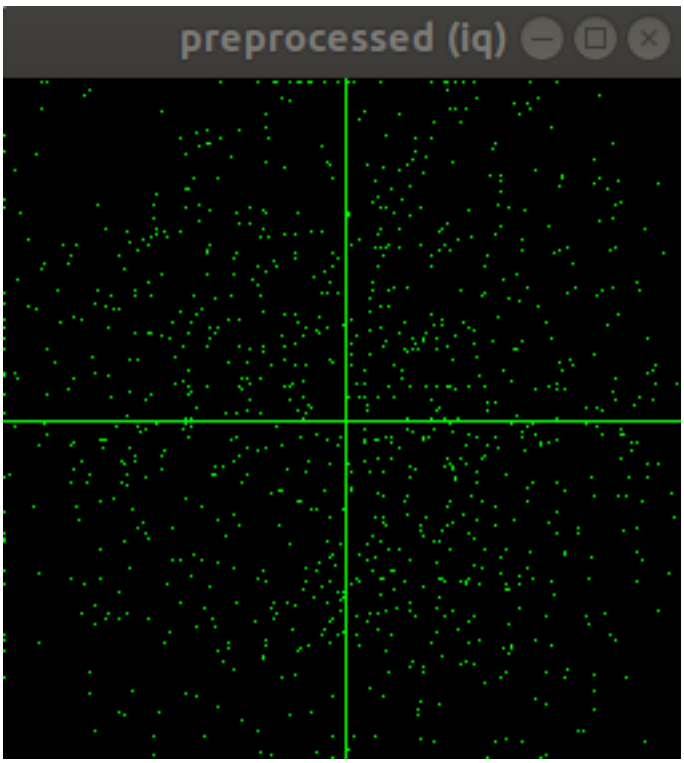


Figure 13: Pre-processed IQ samples

The default gain is 40, and you can then experiment with lower values.

The IQ points should form a more compact cloud of points, such as the one below:

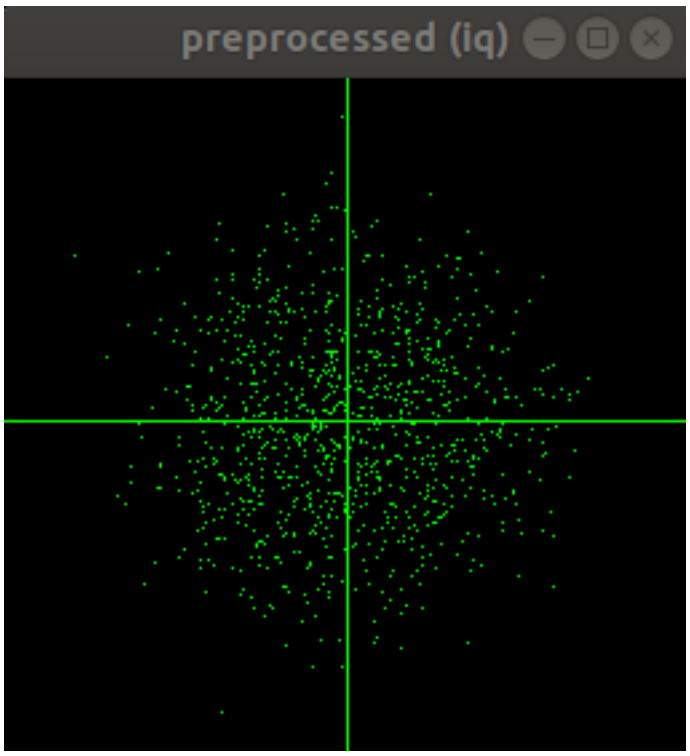


Figure 14: Pre-processed IQ with lower Rx gain

More information is available in Section 9.2 of the leandvb application's user guide.

Next, observe the spectrum plots. The spectrum plot shows the limits of the central band in red lines. In the example that follows, the signal presents the frequency offset of roughly -300 kHz that we already knew about from our observation on gqrx:

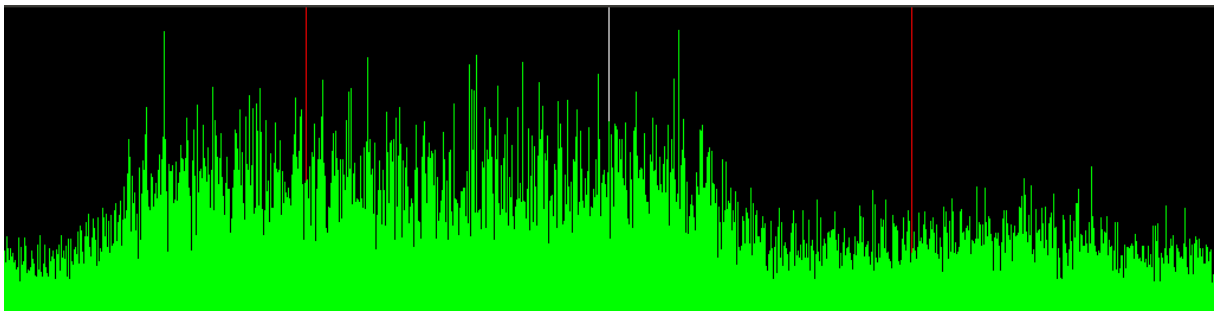


Figure 15: Leandvb's spectrum plot showing offset signal

> NOTE: each LNB introduces a unique frequency offset, which also varies over time. The above value of -300 kHz was specific to the example setup. Your frequency offset will be different.

To correct the known offset, you can run with option `--derotate`, as follows:

```
blocksat-cli sdr -g [gain] --derotate [freq_offset]
```

where `freq_offset` represents the offset in kHz that you want to correct.

With that, the preprocessed spectrum plot should be centered, as follows:

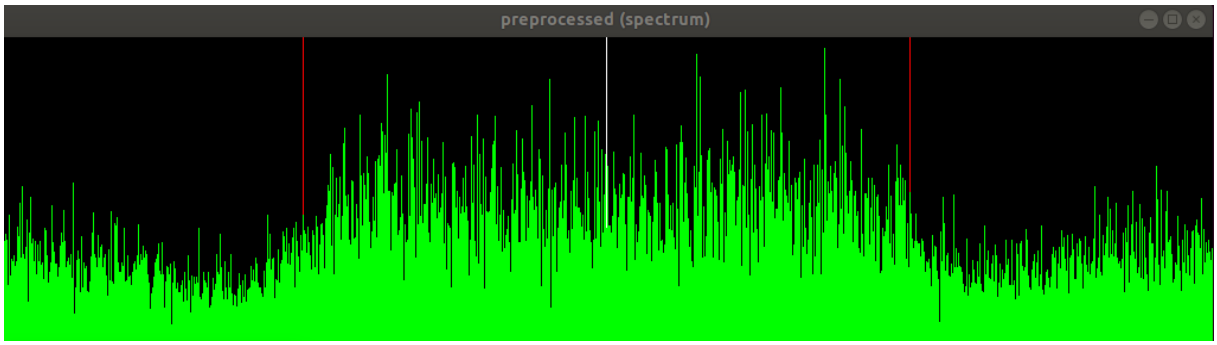


Figure 16: Leandvb's spectrum plot showing centered signal

At this point, if the antenna pointing is already reasonably good, you might see the "PLS cstln" plot showing four visible clouds:

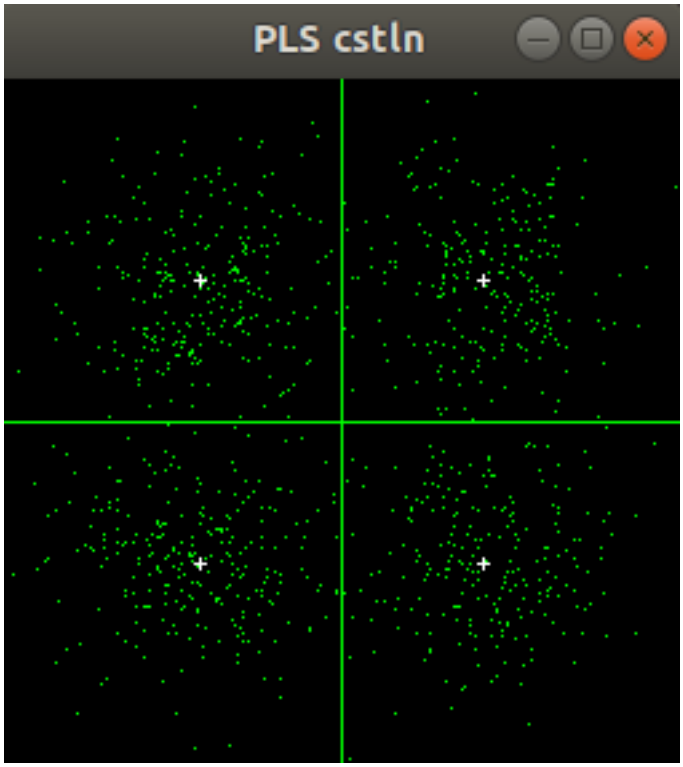


Figure 17: Physical layer signaling (PLS) symbols

This plot indicates that the receiver application is locked to the Blockstream Satellite signal. Note that

the more compact the four clouds of points are in this plot (around the white + marker), the better the signal quality.

If you cannot see the “PLS cstln” plot, it means you are not locked to the signal yet. You can troubleshoot further in debug mode by running like so (with argument `-d`):

```
blocksat-cli sdr -g [gain] --derotate [freq_offset] --gui -d
```

If you see the following logs continuously printed in the console, it means you are not locked to the signal yet:

```
DETECT
PROBE
```

When a lock is acquired, you will see the following log printed to the console:

```
LOCKED
```

After that, you should start seeing several underscores `_` printed consecutively as indicators of successful data reception. The reception indicator can be one of the three below:

- `_`: indicates a DVB-S2 frame received without errors.
- `.`: indicates an error-corrected DVB-S2 frame.
- `!`: indicates a DVB-S2 frame with remaining errors.

If you cannot lock to the signal, you should try further adjustments to the antenna. Assuming you have identified the signal on gqrx before, you can infer that the pointing is already very close. Therefore, only subtle adjustments are required at this point.

Furthermore, if you cannot find the signal on gqrx, you can search for satellite beacons instead. While the Blockstream Satellite signal is seen as a flat level spanning approximately 1 MHz, a beacon is a very narrow signal seen as a narrow pulse (or peak) on gqrx. All you need to do is change the frequency on gqrx to one of the beacon frequencies below:

Satellite	Beacons
Galaxy 18	11701 MHz (Horizontal), 12195 MHz (Vertical)
Eutelsat 113	11701.5 MHz (Vertical), 12199 MHz (Horizontal)
Telstar 11N	11199.25 MHz (Vertical), 11699.50 MHz (Vertical), 11198.25 MHz (Horizontal), 11698.50 MHz (Horizontal)
Telstar 18V	3623 MHz (Vertical), 3625 MHz (Vertical), 4199 (Horizontal)

Make sure to select a beacon whose polarization matches your target signal polarization, which you can check by running:

```
blocksat-cli cfg show
```

Once you find the beacon signal, make gentle adjustments to the pointing until you can maximize the observed signal level. Then, change the frequency on gqrx back to the original downlink frequency of interest (also printed by the above command). You should be able to visualize the Blockstream Satellite signal now.

Ultimately, once the SDR receiver locks to the signal, you can still try to improve the SNR. Observe the SNR values printed to the console and see if you can get better values after subtle adjustments to the pointing.

In the end, once you are satisfied with the SNR (see the target levels), you can monitor several aspects of your SDR receiver. For example:

- To monitor the bitrate, run with option `--ts-monitor-bitrate`.
- To monitor the integrity of the MPEG TS packet sequence, run with option `--ts-monitor-sequence`.
- For low-level debugging information, run with option `-dd`.

8.5.3 Pointing with a Satellite Finder

The antenna alignment procedure can be challenging when using a receiver other than the SDR receiver. The main limitation is that the Linux USB (TBS 5927 or 5520SE), Novra S400, and Sat-IP receivers indicate the lock status as a true or false metric. As a result, you will often see the lock status as false until you suddenly get the correct antenna direction and the lock status changes to true. In contrast, you can observe the received power spectrum in real-time with an SDR receiver and notice the satellite carrier (or beacon) even before the receiver locks. Hence, the alignment process is significantly more straightforward with an SDR receiver.

Other than using an SDR receiver, the alternative solution to obtain a gradual signal strength indicator during the antenna alignment procedure is to use a *satellite finder* device. You can plug the satellite finder into your setup and use it in conjunction with the satellite receiver. The finder can provide a signal strength indicator to allow for the optimal antenna alignment. At the same time, the receiver can confirm the pointing is correct when it properly locks to the signal.

A satellite finder usually has two connections: one to the LNB (typically labeled *satellite*) and the other to the receiver. The receiver is supposed to provide power to the finder (just like how it powers up the LNB), whereas the LNB feeds the signal of interest. Hence, the overall connection is as follows:

```
Receiver <---> Finder <---> LNB
```

Some finders also come with a dedicated power supply. And even if not, you could purchase a power supply such as the one used in the SDR setup. In this case, it suffices to connect the finder to the power supply instead of the receiver, as follows:

```
Power Supply <---> Finder <---> LNB
```

The latter case is handy when using a base station receiver. You can connect the finder inline between the base station's legacy coaxial output and the power supply. Otherwise, with the other receiver types, we recommend connecting the finder inline between the receiver and LNB so that you can inspect the receiver concurrently with the finder.

Satellite finders can also differ widely in their offered capabilities. For instance, some finders only measure the signal level, such as the SF-95DR model included on the satellite kits. Meanwhile, other more advanced finder models can go a step further and demodulate DVB-S2 signals. The instructions in the sequel cover both finder categories.

8.5.3.1 Pointing with a Level Measurement Finder

This section applies to satellite finders capable of measuring the signal level only, such as the SF-95DR model.

To start, connect the finder as instructed earlier and enable/launch the receiver as usual to power up the finder. You should see the finder powered on with an active display.

Next, make sure the finder has an adequate attenuation and gain configuration to observe the signal level. Proceed with the following steps:

1. Press the left ATT button as many times as possible to increase the attenuation to the maximum. Also, increase the dial on the side to its max and check the signal level on the finder's display. This step aims to prepare the finder to measure the strongest possible signal. By setting the highest attenuation first, an eventual strong signal can still fall within the measurable range.
2. If the measured signal level is not 99%, it means the signal received by the finder is not as strong as anticipated. Hence, reduce the attenuation to observe weaker signals. That is, press the right ATT button until you get to a 99% level. Keep the dial on the side at its maximum throughout this process so that you can adjust the attenuation setting first.
3. Once the finder measures 99% signal level, turn the side dial down (reduce the gain) until you get a reasonably low signal level on the finder's display. For example, try achieving a level between 5% and 30%.

At this point, you should be ready to start the antenna alignment process. Make sure to keep an eye on the receiver's lock status (refer to the lock status instructions) throughout the process and continue with the following steps:

1. Adjust the antenna roughly in the right direction based on the azimuth, elevation, and polarity angles obtained from the Coverage Map.

2. Choose a coordinate to optimize (azimuth, elevation, or polarity) and focus on it. For example, start by optimizing the azimuth.
3. Make subtle adjustments to the chosen coordinate until you can observe an increase in the signal level measured by the finder.
4. If the level increases too much and reaches 99%, turn the side dial (gain) down slightly such that the measured signal strength comes back to a low value. If the side dial reaches its minimum and the measured level remains at 99%, it means the signal is now too strong to be measured. Increase the attenuation by pressing the left ATT button once and readjust the side dial to get the measured level back at a low value.
5. Go back to step 3 and repeat the process until you can no longer observe improvements on the measured level for the chosen coordinate.
6. Go back to step 2 and pick the next coordinate to be optimized.

If the receiver logs `Lock = True` on the console at any point, it means you are in the right direction. Otherwise, if you have optimized all coordinates using the above procedure and the receiver still logs `Lock = False`, you may have pointed to the wrong satellite. In this case, inspect the final alignment and repeat the process if necessary.

Note the main disadvantage of a level-measurement finder is that it cannot guarantee the correctness of the satellite. It only measures the signal level, so it may inadvertently guide you to a stronger signal in the sky coming from an adjacent satellite. Hence, it is vital to keep an eye on the receiver's lock status throughout the process. Only the lock status can confirm the antenna alignment is correct.

8.5.3.2 Pointing with a DVB-S2 Capable Finder

If you are using a satellite finder that supports DVB-S2 demodulation, you can configure it to lock to the signal. To do so, you typically need to configure the following signal parameters:

- Downlink frequency
- LNB local oscillator (LO) frequency
- Signal polarization

You can check the appropriate values by running:

```
1 blocksat-cli cfg show
```

Additionally, you will need to define:

- Symbol rate: set it to "1000 kbaud" (or "1000000 baud", or "1 Mbaud", depending on the units adopted by your finder).
- 22 kHz: enable only when using a Universal LNB and pointing to Galaxy 18 or Eutelsat 113 (i.e., when receiving in Ku high band). Otherwise, leave it disabled. See the notes regarding Universal LNBs.

After the configuration, the finder will typically present you with signal strength and (or) quality indicators. At this point, try aligning your antenna until you can maximize these levels.

Once you lock to the signal using the finder, check that your receiver can lock too by following the instructions presented earlier.

9 Bitcoin Satellite

9.1 Overview

Bitcoin Satellite is a fork of FIBRE (Fast Internet Bitcoin Relay Engine) and, consequently, also a fork of Bitcoin Core. It features a version of the bitcoind application with support for the reception of blocks sent over satellite in UDP datagrams with multicast addressing. You can find detailed information regarding how Bitcoin Satellite works on the project's Wiki page.

9.2 Installation

Install bitcoin-satellite by running:

```
1 blocksat-cli deps install --btc
```

> NOTE:
>
> - This command supports the two most recent releases of Ubuntu LTS, Fedora, CentOS, Debian, and Raspbian.
> - bitcoin-satellite is a fork of bitcoin core. As such, it installs applications with the same name (i.e., `bitcoind`, `bitcoin-cli`, `bitcoin-qt`, and `bitcoin-tx`). Hence, the installation of `bitcoin-satellite` will fail if you already have bitcoin core installed.

Alternatively, you can install bitcoin-satellite manually from binary packages or from source.

9.3 Configuration

Next, you need to generate a `bitcoin.conf` file with configurations to receive bitcoin data via satellite. To do so, run:

```
1 blocksat-cli btc
```

By default, this command will place the generated `bitcoin.conf` file at `~/.bitcoin/`, which is the default Bitcoin data directory used by Bitcoin Satellite. If you so desire, you can specify an alternative `datadir` as follows:

```
blocksat-cli btc -d datadir
```

9.4 Running

Next, run `bitcoind` as usual, like so:

```
bitcoind
```

Note that other Bitcoin Core options are supported and can be added to the generated `bitcoin.conf` file as needed, or directly as arguments to the above command. For example, you can run the node based on satellite links only (unplugged from the internet) using option `connect=0` on `bitcoin.conf` or by running:

```
bitcoind -connect=0
```

Also, you can run `bitcoind` in daemon mode:

```
bitcoind -daemon
```

Once `bitcoind` is running, you can check the satellite interface is receiving data by running the following command:

```
bitcoin-cli getudpmulticastinfo
```

If the receiver is correctly locked to the satellite signal, you should see a bitrate around 1.09 Mbps.

Furthermore, you can check the number of blocks being received concurrently over satellite with the following command:

```
bitcoin-cli getchunkstats
```

9.5 Further Information

9.5.1 UDP Multicast Reception Option

In a Blockstream Satellite receiver setup, the satellite receiver will decode and output a UDP/IPv4 stream. Bitcoin Satellite, in turn, listens to such a stream when configured with option `udpmulticast` (added to the `bitcoin.conf` file).

There are several possibilities regarding the configuration of option `udpmulticast`. It depends on your hardware setup (for instance, your receiver type) and the satellite that you are receiving from. The option is described as follows:

```
  -udpmulticast=<if>,<dst_ip>:<port>,<src_ip>,<trusted>[,<label>]
```

> Listen to multicast-addressed UDP messages sent by towards : using interface . Set to 1 if sender is a trusted node. An optional may be defined for the multicast group in order to facilitate inspection of logs.

Here is an example:

```
udpmulticast=dvb0_0,239.0.0.2:4434,172.16.235.9,1,blocksat
```

In this case, we have that:

- `dvb0_0` is the name of the network interface that receives data out of the receiver.
- `239.0.0.2:4434` is the destination IP address and port of the packets that are sent over satellite.
- `172.16.235.9` is the IP address of the Blockstream ground station node broadcasting data over the satellite network (each satellite has a unique source IP address).
- `1` configures this stream as coming from a *trusted* source, which is helpful to speed-up block reception.
- `blocksat` is a label used simply to facilitate inspection of logs.

To simplify the process, command `blocksat-cli btc` generates the `bitcoin.conf` file for you.

9.5.2 Installation from Binary Packages

You can install `bitcoin-satellite` directly from binary packages that are available for the two most recent Ubuntu LTS, Fedora, CentOS, Debian, and Raspbian releases.

Ubuntu:

```
add-apt-repository ppa:blockstream/satellite
apt-get update
apt-get install bitcoin-satellite
```

> If command `add-apt-repository` is not available, install `software-properties-common`.

Debian:

```
add-apt-repository https://aptly.blockstream.com/satellite/debian/
apt-key adv --keyserver keyserver.ubuntu.com \
    --recv-keys 87D07253F69E4CD8629B0A21A94A007EC9D4458C
apt-get update
apt-get install bitcoin-satellite
```

> Install `gnupg`, `apt-transport-https`, and `software-properties-common`, if necessary.

Raspbian:

```
add-apt-repository https://aptly.blockstream.com/satellite/raspbian/
apt-key adv --keyserver keyserver.ubuntu.com \
    --recv-keys 87D07253F69E4CD8629B0A21A94A007EC9D4458C
apt-get update
apt-get install bitcoin-satellite
```

> Install `gnupg`, `apt-transport-https`, and `software-properties-common`, if necessary.

Fedora:

```
dnf copr enable blockstream/satellite
dnf install bitcoin-satellite
```

> If command `dnf copr enable` is not available, install `dnf-plugins-core`.

CentOS:

```
yum copr enable blockstream/satellite
yum install bitcoin-satellite
```

> If command `yum copr enable` is not available, install `yum-plugin-copr`.

9.5.3 Compilation from Source

To build Bitcoin Satellite from source, first clone the repository:

```
git clone https://github.com/Blockstream/bitcoinsatellite.git
cd bitcoinsatellite/
```

Then, install all build requirements listed in the project's documentation.

Next, run:

```
./autogen.sh
./configure
make
```

This will build the `bitcoind` application binary within the `src/` directory and you can execute it from there. Alternatively, you can install the application in your system:

```
make install
```

Detailed build instructions can be found within the project's documentation.

10 Dual-Satellite Connection

Some regions worldwide are covered by two satellites at the same time. For example, most of the US has coverage from both Galaxy 18 and Eutelsat 113. In Asia, there is extensive overlapping coverage from Telstar 18V C and Telstar 18V Ku. If you are in such a region with overlapping coverage, you can connect to two satellites simultaneously and double the Bitcoin block transfer speed. This section describes the hardware and host configurations required to run such a dual-satellite setup.

Before continuing, however, you should check if your location has overlapping coverage from two satellites using our Coverage Map. Also, note distinct signal strengths are expected from each satellite, given that the distance and viewing angles from your station to each satellite are different. Hence, we also recommend checking the Link Analyzer tool for more specific antenna recommendations for each satellite.

10.1 Required Hardware

A straightforward way to connect to two satellites simultaneously is to use separate antennas pointed to each satellite. To do so, you need two LNBs (one per dish), double the number of cables, twice as many connectors, etc. Nevertheless, it is also possible to receive from two satellites simultaneously while using a single parabolic reflector. However, this approach requires more advanced parts and installation skills.

For instance, if your location has overlapping coverage from Galaxy 18 and Eutelsat 113, you can use a single dish and two LNBs. To do so, you need a dual or multi-feed LNB holder capable of spacing LNBs around the parabolic reflector's focal region to receive from satellites that are 10° apart in orbit. However, note this setup requires a more intricate antenna alignment procedure. Hence, we recommend contacting a local professional installer. Otherwise, if you would rather have a more straightforward installation on your own, we recommend setting up two independent antennas, one for each satellite beam.

> If your location has overlapping coverage from the Telstar 18V C band and Ku band beams, note these are two different beams out of the same satellite. In principle, you could use a single dual-band (C and Ku) combo LNB if you were able to find one. However, such combo LNB models usually select one band or the other (C or Ku) instead of outputting both bands simultaneously. Hence, they are not sufficient for a dual-satellite setup. Once again, we recommend installing two independent antennas with independent LNBs instead. The Pro Kit already comes with C and Ku band LNBs (see the parts list), so you only need to purchase two dishes (and a few cables) in addition to the kit.

Next, note you also need two receivers for a dual-satellite setup, one connected to each LNB. The only exception is if your receiver is the Novra S400 of the Pro Ethernet Kit. This receiver model inherently

supports dual satellite connectivity by offering two independent radio-frequency (RF) channels. Otherwise, if using a Sat-IP, USB, or SDR-based receiver (or combinations of them), you need two receiver units, each connected to a different antenna/LNB. If using the Satellite Base Station (an integrated receiver-antenna), you also need two units, each pointed to a distinct satellite.

10.2 Host Configuration

Once you have the required hardware parts, the next step is to configure the receivers using the host computer. Regardless of the adopted hardware, you need to create different configurations on the command-line interface (CLI). Recall that your first step with the CLI is to run the configuration helper, as follows:

```
1 blocksat-cli cfg
```

To create a second receiver configuration, you need to use option `--cfg name`. For example, you can set up a second configuration named `rx2`, as follows:

```
1 blocksat-cli --cfg rx2 cfg
```

Then, select the other satellite of interest and inform the parts composing your second receiver setup. Subsequently, you can run all CLI commands using option `--cfg rx2`. Specific instructions are provided next.

10.2.1 Novra S400 Standalone Receiver

With the Novra S400, you need to configure the two RF interfaces separately. Each interface will be connected to a different antenna and receiving from a different satellite. As explained on the S400 guide, the first RF interface (RF1) is configured by the following command:

```
1 blocksat-cli standalone cfg
```

To configure the second RF interface, run:

```
1 blocksat-cli --cfg rx2 standalone --demod 2 cfg --rx-only
```

Next, access the S400 web management console as instructed in the S400 guide. Go to `Interfaces > RF2` and enable the RF2 interface.

Lastly, you need to configure Bitcoin Satellite to receive the second satellite stream. You can do so by running:

```
1 blocksat-cli --cfg rx2 btc --concat
```

10.2.2 TBS USB Receiver

With a TBS 5927 or 5520SE USB receiver, you would ordinarily run the following sequence of commands:

1. Initial configurations:

```
blocksat-cli cfg
```

2. Installation of dependencies:

```
blocksat-cli deps install
```

3. Configuration of the host interfaces:

```
blocksat-cli usb config
```

4. Receiver launch:

```
blocksat-cli usb launch
```

To use a second TBS unit as the second receiver of a dual-satellite setup, you only need to repeat steps 3 and 4 while including argument `--cfg rx2`, as follows:

```
blocksat-cli --cfg rx2 usb config

blocksat-cli --cfg rx2 usb launch
```

Make sure to select the second TBS 5927/5520SE unit on both steps.

Lastly, you need to configure Bitcoin Satellite to receive from the second TBS device. You can do so by running:

```
blocksat-cli --cfg rx2 btc --concat
```

10.2.3 Blockstream Base Station Sat-IP Receiver

With the Satellite Base Station Sat-IP receiver, you need two base station devices for a dual-satellite setup. Also, when launching the Sat-IP client, you need to select the correct receiver by IP address.

Run the first Sat-IP client and select the correct receiver when prompted:

```
blocksat-cli sat-ip
```

Next, launch the second Sat-IP client and, again, select the appropriate receiver:

```
blocksat-cli --cfg rx2 sat-ip
```

10.2.4 SDR Receiver

To set up an SDR-based receiver, you would ordinarily run the following sequence of commands:

1. Initial configurations:

```
blocksat-cli cfg
```

2. Installation of dependencies:

```
blocksat-cli deps install
```

3. Receiver launch:

```
blocksat-cli sdr
```

To run a second SDR-based receiver, you only need to repeat step 3 while switching to the second configuration, as follows:

```
blocksat-cli --cfg rx2 sdr
```

> NOTE: if you are running two SDR-based receivers on the same host, with two RTL-SDR dongles, you can select the RTL-SDR dongle using option `--rtl-idx`. For example, for the second RTL-SDR, run:
>
> ```
> blocksat-cli --cfg rx2 sdr --rtl-idx 1
> ```

Lastly, you need to configure Bitcoin Satellite to receive the second satellite stream. You can do so by running:

```
blocksat-cli --cfg rx2 btc --concat
```

11 Satellite API

The Blockstream Satellite API provides developers with an easy-to-use RESTful API that can be used to broadcast messages globally using the Blockstream Satellite network.

As illustrated in the diagram below, the process starts with a sender application, which requests the transmission of a particular file or text message. This transmission order gets queued up in the API server. Once the order is paid over the Bitcoin Lightning Network, the API server sends the message to the Blockstream Satellite Teleport (ground station). From there, the message is broadcast globally through the Blockstream Satellite network.

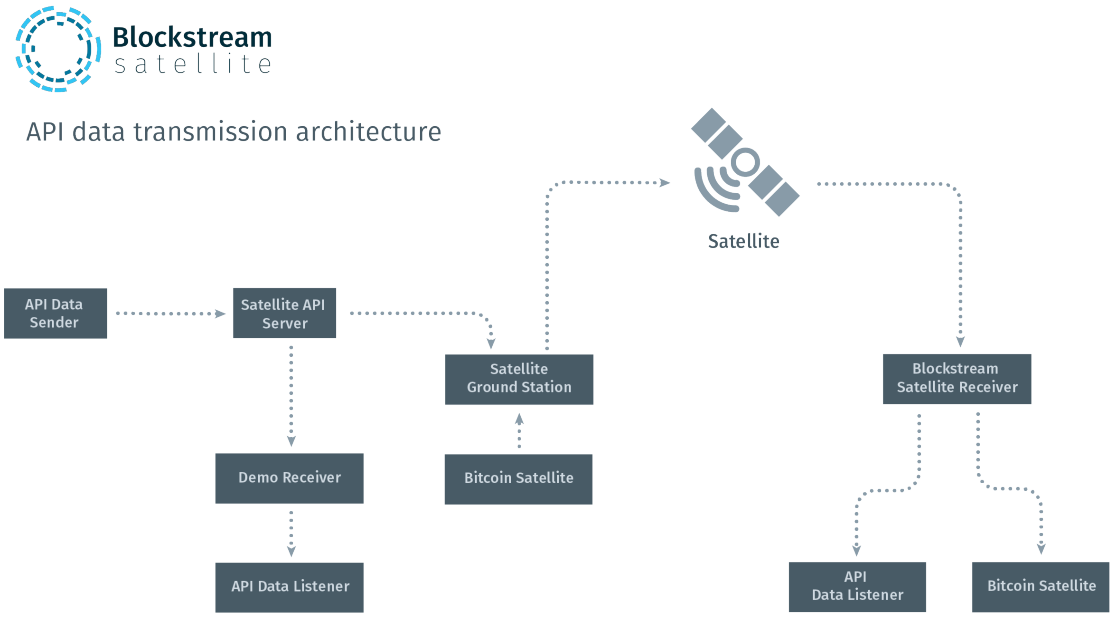


Figure 18: Blockstream Satellite API architecture

The `blocksat-cli` command-line interface (CLI) provides a range of commands for the interaction with the Satellite API. This guide clarifies these commands.

> To install the CLI, please refer to the installation instructions.

> The API support on the CLI is available starting from version `0.3.0`. Please refer to instructions regarding how to check and upgrade your CLI version.

For details regarding the RESTful API, please refer to the Satellite API's repository.

11.1 Encryption Keys

To start, we recommend setting up a GPG key pair for the encryption of messages sent through the satellite API. To do so, run:

```
1 blocksat-cli api cfg
```

After filling the requested information, this command will generate the key pair (public and private keys) and create a new keyring, by default, located at the `~/.blocksat/` directory.

11.2 Satellite API Transmission

To send a text message over the Blockstream Satellite network, run:

```
blocksat-cli api send
```

Alternatively, you can send a file by running:

```
blocksat-cli api send -f [file]
```

where `[file]` should be replaced with the path to the file.

The application asks for the bid in millisatoshis (msats) and suggests the minimum acceptable bid. To accept the suggested bid, simply press enter and continue. Otherwise, if you prefer to fill in the bid manually, make sure to satisfy the two requirements below:

1. The total bid must be greater than at least 1000 msats.
2. The ratio between the bid in msats and the number of bytes used for transmission (the so-called bid/byte ratio) must be at least 1 msat/byte.

After confirming the bid, get the *Lightning Invoice Number* printed on the console or the QR code and pay it using Bitcoin Lightning (refer to the list of Lightning wallet apps). Once the payment is confirmed, the transmission will start as soon as the transmission queue serves your message.

By default, the above commands encrypt your message or file using the GPG key you set up in the beginning. With that, only you (the owner of the private key) can decrypt the message on reception (see GPG's manual for further information). Other users listening for messages broadcast over the Blockstream Satellite network will still receive your encrypted message. However, they **will not** be able to decrypt it.

11.2.1 Choosing the Recipient

You can also define a specific recipient for your transmission. To do so, use argument `-r`, as follows:

```
blocksat-cli api send -r [fingerprint]
```

where `[fingerprint]` is the public key fingerprint corresponding to the target recipient. In other words, the *fingerprint* defines who is going to be able to decrypt the message.

In case you want to skip the validation of the recipient's public key and assume it is fully trusted, you can use argument `--trust`. For example:

```
blocksat-cli api send -r [fingerprint] --trust
```

These commands assume that the recipient's public key is available on the local GPG keyring. Thus, you need to import the public key into the keyring located at `~/.blocksat/.gnupg`. Assuming the

recipient has shared its *public key* with you on a file named `recipient.gpg`, you can do so using:

```
gpg --import recipent.gpg --homedir $HOME/.blocksat/.gnupg
```

11.2.2 Signing the Messages

You can also digitally sign a message so that the recipient can verify it was really generated by you and not altered in any way. To do so, use argument `--sign`. For example:

```
blocksat-cli api send --sign
```

By default, messages will be signed using the default private key from your keyring, but you can specify the key as follows:

```
blocksat-cli api send --sign --sign-key [fingerprint]
```

where `[fingerprint]` is the fingerprint of the private key you wish to use for signing.

11.3 Satellite API Reception

To receive messages sent over satellite through the Satellite API, first of all, you need to have your satellite receiver running. If you do not have a real receiver, you can experiment with the demo receiver explained in the sequel.

The satellite receiver continuously receives Bitcoin data and API messages broadcast over satellite. To listen for the API messages, run:

```
blocksat-cli api listen
```

Once an API message is received, this application first tries to decrypt it. If it succeeds, it then validates the integrity of the data and saves the final result into the download directory (by default at `~/.blocksat/api/downloads/`).

Note that the incoming data stream multiplexes transmissions from all users of the Satellite API. Hence, the application is expected to fail decryption in most cases except when it finds a message it is supposed to receive.

Also, by default, this app saves both files and text messages into the download directory. You can also configure it to print a received (successfully decrypted) text message to the console by running with:

```
blocksat-cli api listen --echo
```

11.3.1 Choosing the Sender

By default, the listener application processes any file that is successfully decrypted. In other words, it processes any message encrypted using your public key.

You can also filter the messages by the sender, as follows:

```
1 blocksat-cli api listen --sender [fingerprint]
```

where `[fingerprint]` is the public key fingerprint corresponding to the target sender.

In this case, the listener application only processes the messages that are digitally signed by the specific sender of interest.

11.4 Demo Receiver

The demo receiver application imitates the output of a real Blockstream Satellite receiver. It outputs UDP packets containing the API messages, just like the regular receiver does. The difference is that it fetches these messages directly through the internet, rather than receiving via satellite.

You can run the demo receiver with:

```
1 blocksat-cli api demo-rx
```

Then, on another terminal session, you can listen for API messages coming from the demo receiver by running:

```
1 blocksat-cli api listen --demo
```

> NOTE: the demo receiver sends packets with API data to the loopback interface. When the listener application is launched with option `--demo`, it will correspondingly listen to the loopback interface.

11.5 Further Information

11.5.1 Lightning Wallets

Here are some options of Lightning Wallets that can be used to pay for API transmissions:

- Bluewallet (iOS and Android)
- Breez (iOS and Android - in Beta)
- Zap (iOS, Android, Windows, Mac, and Linux)

You can also set up a Lightning node and use the lightning-cli to handle payments.

11.5.2 Plaintext Mode

In addition to encrypted messages, you can also send and receive plaintext messages, i.e., unencrypted messages. In this case, all users listening to Satellite API messages will be able to see your message. Correspondingly, you can receive all messages sent by other users in plaintext format.

To send a plaintext message, run:

```
1 blocksat-cli api send --plaintext
```

To receive plaintext messages, run:

```
1 blocksat-cli api listen --plaintext
```

In this case, please **be aware** that **all** API transmissions will be saved to the download directory. In contrast, in normal mode (with encryption), the listener application only saves the messages addressed to you (i.e., the messages you can decrypt).

Alternatively, you can filter plaintext messages by the sender, using the `--sender` option explained earlier. In this case, the application retains only the clearsigned messages sent by the specific sender of choice.

11.5.3 Receiving Messages Sent from the Browser

If you want to receive a file uploaded directly on the Satellite Queue page, from the browser, run the API listener application as follows:

```
1 blocksat-cli api listen --plaintext --save-raw
```

The rationale is that files uploaded to the Satellite Queue are sent in plaintext (i.e., without encryption). Furthermore, the browser transmission tool does not encapsulate the data in the same way as the CLI sender tool (i.e., command `blocksat-cli api send`). Argument `--save-raw` accounts for this missing encapsulation.

The CLI can also reproduce the message transmission format used by the Satellite Queue tool on the browser. To do so, run the sender application as follows:

```
1 blocksat-cli api send --plaintext --send-raw
```

11.5.4 Reliable Transmissions

The API messages sent over satellite are not guaranteed to be received by all satellite receivers. Each receiver experiences a unique reception quality, depending primarily on its location, weather conditions, and the adopted receiver hardware. When the signal quality is low, it becomes more likely for the receiver to fail on the reception of packets.

One way to increase the chances of successful reception is to use forward error correction (FEC). In essence, FEC adds redundancy to the transmitted data so that receivers can recover the original message even if some parts are missing. This is the mechanism that is used, for instance, in Bitcoin Satellite (see further details in the project's Wiki).

To send an API message using FEC encoding, run:

```
blocksat-cli api send --fec
```

> NOTE: the FEC feature requires the `zfec` dependency. You can enable it by installing the CLI with `sudo pip3 install blocksat-cli[fec]` or by installing the `zfec` package via pip3 directly.

The `api listen` command detects and decodes FEC-encoded messages automatically.

In general, the higher the number of extra (redundant) pieces of data sent over satellite, the better the protection to data loss over the satellite link. The user can tune this parameter using the command-line argument `--fec-overhead`. By default, this argument is `0.1` (equivalent to 10%), such that, for a message that originally occupies 10 packets, the application sends one extra packet.

11.5.5 Transmission over Selected Regions

By default, API messages are broadcast worldwide through the six satellite beams currently composing the Blockstream Satellite network. However, you can also explicitly select the regions for each transmission.

For example, to send over regions 0 (G18) and 1 (E113) only, call the sender application as follows:

```
blocksat-cli api send --regions 0 1
```

The regions are numbered from 0 to 5 according to the following mapping:

Region	Satellite Beam
0	Galaxy 18 (G18)
1	Eutelsat 113 (E113)
2	Telstar 11N Africa (T11N AFR)
3	Telstar 11N Europe (T11N EU)
4	Telstar 18V C band (T18V C)
5	Telstar 18V Ku band (T18V Ku)

11.5.6 Running on Testnet

The API commands described thus far interact with the API server that handles live broadcasting via the Blockstream Satellite network. This server operates on Bitcoin's Mainnet network and, therefore, the payment requires actual bitcoins.

Nevertheless, there is an alternative API server that operates on the Bitcoin Testnet network. You can interact with this Testnet server using argument `--net test`. For example:

```
1 blocksat-cli api --net test send
```

However, note that the Testnet server does not transmit data through the satellite network. It only broadcasts the data to clients that are connected directly to the server through the internet. Thus, you need to use the demo receiver to receive API messages sent through the Testnet server.

In this case, run the demo receiver as follows:

```
1 blocksat-cli api --net test demo-rx
```

11.5.7 Bump and Delete API orders

When users send messages to the Satellite API, these messages first go into the Satellite Queue. From there, the satellite transmitter serves the transmission orders with the highest bid (per byte) first.

If your message is still waiting for transmission due to other messages with a higher bid per byte, you can bump your bid by running:

```
1 blocksat-cli api bump
```

You can also delete (cancel) a transmission order by running:

```
1 blocksat-cli api del
```

Both of these commands will ask for the UUID and the authorization token of the order. These were printed to the console when the message was first sent.

11.5.8 Password-protected GPG keyring

By default, it is assumed that the private key in your keyring is password-protected. In case you want to use a private key that is **not** password-protected for signing a transmission or decrypting incoming messages, run the apps with option `--no-password`, as follows:

```
1 blocksat-cli api send --sign --no-password
```

```
1 blocksat-cli api listen --no-password
```

11.5.9 Automating Lightning Payments

The command used to send API messages (`blocksat-cli api send`) is by default interactive. It prompts for the bid and prints the Lightning invoice number. The user, in turn, has to choose the bid and pay the invoice manually.

Nevertheless, there are options to automate the payment. You can specify the bid using the command-line argument `--bid`. Additionally, you can run an arbitrary command with the invoice number. For example, you can use the lightning-cli to pay the invoice automatically as follows:

```
blocksat-cli api send -f [file] --bid [bid] \
    --invoice-exec "lightning-cli pay {}"
```

This command will send the transmission order to the server directly with the given bid. Subsequently, it will execute the `lightning-cli pay` command while substituting `{}` with the invoice number (i.e., the `bolt11` payment request string).

11.5.10 Executing Commands with Received Files

The API data listener application provides the command-line option `--exec`, which configures an arbitrary command to be executed for each file received through the satellite API stream. For example, the option can be used as follows:

```
blocksat-cli api listen --exec 'cat {}' --sender [fingerprint]
```

where `[fingerprint]` should be the public key fingerprint corresponding to the target sender.

In this case, the application will execute the given command (`cat`) for each downloaded file while substituting `{}` with the path to the file in the download directory. In other words, this example will print the contents of every incoming file on the standard output (similar to option `--echo`).

Due to security reasons, this option requires two safety layers:

- **Encryption**: it only runs the specified command for successfully decrypted messages.
- **Digital signature**: it requires digitally signed messages signed by a specific sender of choice and verified.

The encryption requirement guarantees that the `--exec` command only gets executed for files explicitly addressed to your node. In other words, the command applies to messages sent by someone who has your public key and encrypted specifically to your node as the recipient. Meanwhile, the digital signature guarantees that the sender is a selected (trustworthy) source and not an unintended one (e.g., a malicious user who has your public key).

If you don't want to specify the sender, you can still run the command with option `--insecure`, as follows:

```
blocksat-cli api listen --exec 'cat {}' --insecure
```

In this case, make sure to **avoid unsafe commands** and **use at your own risk**.

11.5.11 Satellite API Channels

The Satellite API messages are sent over satellite through multiple *channels*. Each channel is identified by a corresponding number. For example, channel 1 is the default channel used for user transmissions. Meanwhile, there are other active channels for applications described in the sequel.

When receiving API messages via satellite, you can tune to a specific channel. To do so, use argument `--channel` on the `api listen` command. For example, to listen to messages coming through channel 4 instead of the default channel 1, run the listener app as follows:

```
blocksat-cli api listen --channel 4
```

In most cases, however, it is not necessary to specify the channel number. The application configures the appropriate channel automatically based on other command-line arguments. For example, this is the case with the Lightning Gossip and Bitcoin source code options described next.

11.5.12 Lightning Gossip Snapshots

The satellite API has a channel dedicated to Lightning gossip messages, namely messages carrying snapshots of the gossip synchronization mechanism available for the Lightning Network. To receive such messages, simply run the listener application using argument `--gossip`, as follows:

```
blocksat-cli api listen --gossip
```

When this argument is specified, the listener application tunes to the appropriate channel for gossip messages (channel 4). Furthermore, it automatically applies other required configurations to receive the gossip messages. For example, it automatically invokes the `historian-cli` tool in order to load gossip snapshots downloaded via satellite.

11.5.13 Bitcoin Source Code Messages

The satellite API also has a channel dedicated to messages carrying the Bitcoin Satellite and Bitcoin Core source codes. To receive such messages, run the listener application using argument `--btc-src`, as follows:

```
blocksat-cli api listen --btc-src
```

This argument configures the listener application to tune to channel 5 and applies other required configurations.

12 Running on Docker

A Docker image is available to be used as the *Blockstream Satellite host*, that is, the system with everything you need to interface with the supported satellite receivers. All you need to do is run the `blockstream/satellite` Docker image while providing the appropriate resources to the container. This process is explained next.

12.1 Standalone Receiver

There are no special requirements to communicate with the standalone receiver from a container. You can launch the `satellite` image as follows:

```
docker run --rm -it \
    -v blocksat-cfg:/root/.blocksat/ \
    blockstream/satellite
```

Note the `blocksat-cfg` named volume is meant to provide persistent storage for the configurations created by `blocksat-cli`.

12.2 USB Receiver

First of all, there is an important limitation to running a Linux USB receiver inside a container. The USB receiver's drivers must be installed on the Docker host, not on the Docker container. Hence, the referred `blockstream/satellite` image does not contain the drivers. Instead, you will need to install the drivers on your Docker host. Please refer to the driver installation instructions on the USB receiver guide.

Next, after installing the drivers and connecting the USB receiver to your host, you can start the container. Just note that you will need to share the DVB network interface (visible on the Docker host) with the container. To do so, check the DVB interface at `/dev/dvb/` (typically named `adapter0`) and assign it to the container using option `--device` as follows:

```
docker run --rm -it \
    --device=/dev/dvb/adapter0 \
    --network=host \
    --cap-add=NET_ADMIN \
    --cap-add=SYS_ADMIN \
    -v blocksat-cfg:/root/.blocksat/ \
    blockstream/satellite
```

After that, you can run the USB configuration command inside the container:

```
blocksat-cli usb config
```

The above step creates a network interface (typically named `dvb0_0`), configures the appropriate firewall rules, and assigns an IP address to the interface. Additionally, it configures the so-called reverse-path filtering rule for the interface. However, this particular configuration will not take effect when executed inside the container, as the container does not have permission to change the reverse-path filtering rules. Hence, to complete the configuration, run the following command directly from the host instead:

```
blocksat-cli rp -i dvb0_0
```

> Note: If your network interface is named differently (not `dvb0_0`), you can find its name by running: `ip link show | grep dvb`.

Finally, launch the receiver inside the container:

```
blocksat-cli usb launch
```

12.3 SDR Receiver

The important point for running the SDR receiver inside a container is that you need to share the RTL-SDR USB device (connected to the Docker host) with the container. To do so, run the container as follows:

```
docker run --rm -it \
    --privileged \
    -v /dev/bus/usb:/dev/bus/usb \
    -v blocksat-cfg:/root/.blocksat/ \
    blockstream/satellite
```

Note **privileged mode** is used to grant access to the RTL-SDR USB device. Furthermore, it is used to allow the execution of `sysctl`, which the SDR application uses for changing option `fs.pipe-max-size`.

> On a Mac OSX host, you will need to set up a docker-machine to share the SDR USB device. Once the docker-machine is active, you can share the USB device through the machine driver's settings. Then, run the above `docker run` command normally.

12.4 Sat-IP Receiver

The essential point for running the Sat-IP client inside a container concerns the network configuration. By default, the Sat-IP client discovers the Sat-IP server via UPnP. However, the discovery mechanism

does not work if the container runs on an isolated network. To solve the problem, you need to launch the container using option `--network=host` as follows:

```
docker run --rm -it \
    --network=host \
    -v blocksat-cfg:/root/.blocksat/ \
    blockstream/satellite
```

Alternatively, if you know the IP address of the Sat-IP receiver, you can specify it directly using option `-a`/`--addr` when running the Sat-IP client. In this case, you don't need the `--network=host` option when launching the container.

12.5 Bitcoin Satellite

In addition to controlling or running the satellite receivers, you can also run Bitcoin Satellite using the `satellite` container image. For example, you can run the following:

```
docker run --rm -it \
    -v ~/.bitcoin/:/root/.bitcoin/ \
    -v blocksat-cfg:/root/.blocksat/ \
    blockstream/satellite
```

> NOTE: with option `-v ~/.bitcoin/:/root/.bitcoin/`, the `bitcoind` application running inside the container will use the host's `~/.bitcoin/` directory as its Bitcoin data directory.

Then, inside the container, run `bitcoind` as usual.

Also, if you have not generated your `bitcoin.conf` configuration file yet, you can run the following inside the container:

```
blocksat-cli btc
```

12.6 Build the Docker Image Locally

You can also build the Docker image locally, rather than pulling it from Docker Hub. To do so, run the following from the root directory of the repository:

```
make docker
```

13 Frequency Guide

This section summarizes several frequencies of interest.

13.1 Signal Bands

Blockstream Satellite operates in Ku high band, Ku low band and C band, depending on region. Ku high band is used in North America and South America. Ku low band is used in Africa, Europe, and Asia. C band is used in the Asia-Pacific region.

Band	Frequency Range
C band	3.7 GHZ - 4.2 GHz
Ku low band	10.7 to 11.7 GHz
Ku high band	11.7 to 12.75 GHz

13.2 Signal Frequencies

The following table summarizes the transmission bands and frequencies of the signals that we broadcast in each coverage region:

Satellite	Band	Frequency
Galaxy 18	Ku High	11913.4 MHz
Eutelsat 113	Ku High	12066.9 MHz
Telstar 11N Africa	Ku Low	11452.1 MHz
Telstar 11N Europe	Ku Low	11505.4 MHz
Telstar 18V Ku	Ku Low	11506.75 MHz
Telstar 18V C	C	4057.4 MHz

13.3 L-band Frequencies

Next, the following table summarizes the L-band frequencies to be used in each region based on the typical LNB local oscillator (LO) frequencies:

LO Frequency	5150 MHz	9750 MHz	10600 MHz	10750 MHz
Galaxy 18			1313.4 MHz	1163.4 MHz
Eutelsat 113			1466.9 MHz	1316.9 MHz

LO Frequency	5150 MHz	9750 MHz	10600 MHz	10750 MHz
Telstar 11N Africa		1702.1 MHz		
Telstar 11N Europe		1755.4 MHz		
Telstar 18V Ku Band		1756.75 MHz		
Telstar 18V C Band	1092.6 MHz			

14 Monitoring Server

While running your receiver, you can opt in to report your receiver performance metrics to a Blockstream-hosted server, referred to as the *satellite monitoring server*. To do so, you must explicitly enable the reporting functionality by appending option `--report` when launching the receiver. With this option, the CLI will periodically send reports over the internet. And by doing so, you will be helping us better plan and improve the satellite communications service and worldwide coverage.

The `--report` option can be appended to all supported receiver types, as follows:

TBS 5927 or 5520SE USB receiver:

```
1 blocksat-cli usb launch --report
```

Novra S400 standalone receiver:

```
1 blocksat-cli standalone monitor --report
```

SDR receiver:

```
1 blocksat-cli sdr --report
```

Sat-IP receiver:

```
1 blocksat-cli sat-ip --report
```

> NOTE: This feature was introduced in blocksat-cli version v0.4.0. Please, refer to the upgrade instructions if you are running an older version.

When using this option for the first time, the CLI runs an initial registration procedure to confirm you are running a functional satellite receiver. The procedure involves a two-factor authentication mechanism over satellite, described next.

14.1 Authentication over Satellite

The registration procedure is as illustrated below and involves the following steps:

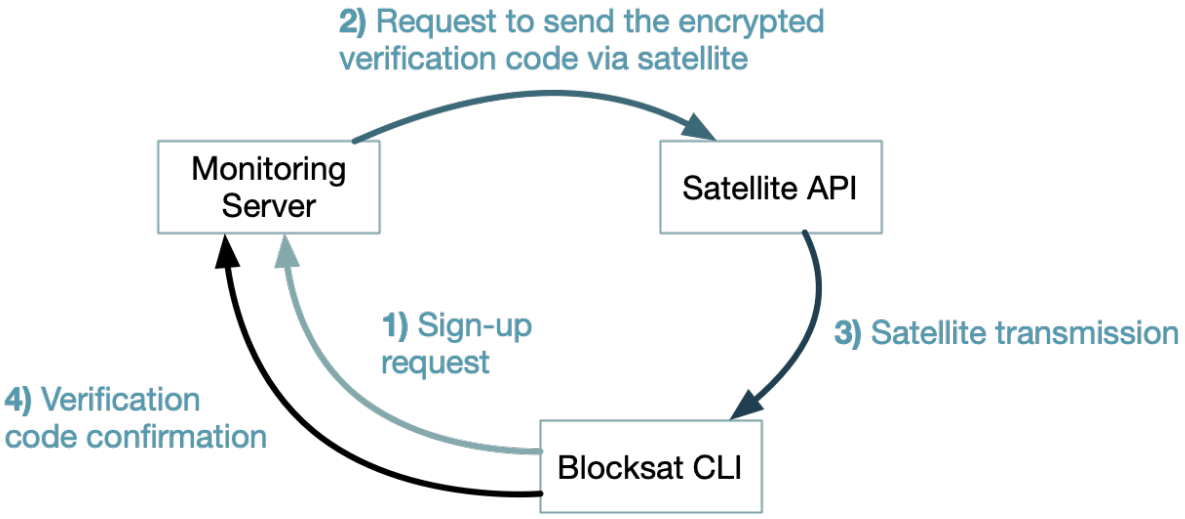


Figure 19: Two-factor authentication procedure

1. The CLI (blocksat-cli) sends a registration request over the internet to the monitoring server. The request includes the user's public GPG key used for API apps.
2. The monitoring server generates a random verification code, encrypts it using the user's public key, and sends the encrypted verification code to the Satellite API for transmission over satellite.
3. The satellite API relays the message over the satellite links worldwide.
4. The CLI (blocksat-cli) receives the encrypted verification code, decrypts it, and confirms the verification code back to the monitoring server.

In the end, this process confirms that the user owns the private key associated with the informed public key, as this is a pre-requisite to decrypt the verification code. Furthermore, it verifies the user is running a functional satellite receiver, given that the encrypted verification code is sent exclusively over the satellite network.

14.2 Authenticated Reports

After the initial registration, the CLI enables periodic reporting of the receiver status. While doing so, it automatically appends a detached GPG signature to every reported metric set. With that, the monitoring server validates the authenticity of each report.

Since version 0.4.5 of the CLI, reports can also be authenticated using a lightweight password-based mechanism as an alternative to GPG detached signatures. The CLI will create the password automatically and save it in ciphertext (encrypted) on your local configuration directory (`~/.blocksat/` by

default). From this point on, it will load the password and use it every time you launch the receiver with reporting enabled.

Note that because reports are GPG-signed, or because the CLI needs to decrypt the local password (since v0.4.5), every time you launch the receiver with option `--report`, the CLI will ask for the passphrase to your local private GPG key. You can observe the underlying process by running the CLI in debug mode. For example, with the USB receiver, you can execute the following command:

```
1 blocksat-cli --debug usb launch --report --log-scrolling
```

15 Quick Reference Guide

This section contains a quick reference guide for the Blockstream Satellite receiver setup and its general usage. Please refer to the main guide for detailed explanations on all steps.

15.1 CLI Installation and Upgrade

Install the command-line interface (CLI):

```
1 sudo pip3 install blocksat-cli
```

Alternatively, to upgrade a previous installation of the CLI, run:

```
1 sudo pip3 install blocksat-cli --upgrade
```

To check your current version, run:

```
1 blocksat-cli -v
```

15.2 Common Steps

These are the commands that are applicable to all the supported types of receivers.

Set initial configurations:

```
1 blocksat-cli cfg
```

Install software dependencies:

```
1 blocksat-cli deps install
```

> To update the dependencies, run:
>
> ```
> blocksat-cli deps update
> ```

Get instructions:

```
blocksat-cli instructions
```

15.3 Receiver-specific Configuration Steps

15.3.1 Novra S400 standalone receiver

Configure the receiver and the host by running:

```
blocksat-cli standalone cfg
```

Monitor the S400 receiver:

```
blocksat-cli standalone monitor
```

15.3.2 TBS USB receiver

Install the drivers:

```
blocksat-cli deps tbs-drivers
```

Configure the host and the TBS 5927/5520SE:

```
blocksat-cli usb config
```

Start the USB receiver:

```
blocksat-cli usb launch
```

15.3.3 Sat-IP receiver

Launch the Sat-IP client:

```
blocksat-cli sat-ip
```

15.3.4 SDR receiver

Configure Gqrx:

```
blocksat-cli gqrx-conf
```

Run the SDR receiver:

```
blocksat-cli sdr
```

15.4 Receiver-specific Antenna Alignment Steps

This is the most time-consuming part of the process and has detailed guidance on the antenna alignment section.

In summary, you will try to point your antenna until you get a signal lock on your receiver.

15.4.1 Novra S400 standalone receiver

Monitor the S400 receiver by running:

```
blocksat-cli standalone monitor
```

While pointing the antenna, check the logs on the terminal until the receiver logs a `Lock`. Alternatively, check the lock indicator on the S400's web UI or front panel until it becomes green (locked).

15.4.2 TBS USB receiver

Make sure that the USB receiver is running with:

```
blocksat-cli usb launch
```

While pointing the antenna, check the logs on the terminal until the receiver logs a `Lock`.

15.4.3 Sat-IP receiver

Launch the Sat-IP client:

```
blocksat-cli sat-ip
```

While pointing the antenna, check the logs on the terminal until the receiver logs a `Lock`.

15.4.4 SDR receiver

Run gqrx:

```
gqrx
```

Point the antenna until you can visualize the Blockstream Satellite signal spanning 1.2 MHz. Take note of the offset between the observed signal's center frequency and the nominal center frequency (at the center of the gqrx plot).

Run the receiver with the GUI enabled and in debug mode:

```
blocksat-cli sdr --gui -d --derotate freq_offset
```

where `freq_offset` is the offset (in units of kHz) you observed on the `gqrx` step.

On the plots that open up, confirm the presence of the signal. Then, wait until the receiver prints `LOCKED` on the terminal.

15.5 Bitcoin-satellite Setup

Install bitcoin-satellite:

```
blocksat-cli deps install --btc
```

> To update a previous installation of bitcoin-satellite, run:
>
> ```
> blocksat-cli deps update --btc
> ```

Generate the `bitcoin.conf` configuration file:

```
blocksat-cli btc
```

Run bitcoin-satellite:

```
bitcoind
```

15.6 Satellite API

Configure encryption keys:

```
blocksat-cli api cfg
```

Broadcast a message using the satellite API:

```
blocksat-cli api send
```

Bump the bid of an API transmission order:

```
blocksat-cli api bump
```

Delete an API transmission order:

```
blocksat-cli api del
```

Listen for API messages acquired by the satellite receiver:

```
blocksat-cli api listen
```

Run the demo receiver:

```
blocksat-cli api demo-rx
```

Listen to the API messages coming from the demo receiver:

```
blocksat-cli api listen -d
```