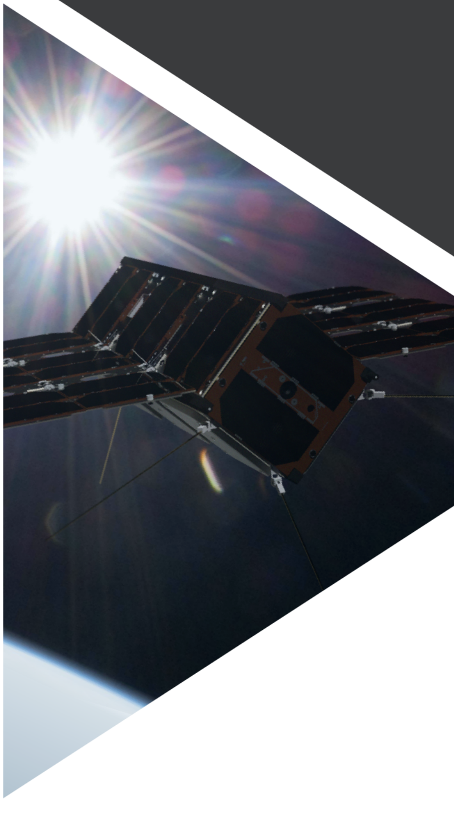
NanoCom Link S, X, SX

User Manual

Guidelines for Usage of the NanoCom Link S, X, and SX Radio Products



MAN 1076241

January 23, 2026

Revision 3.0

NanoCom Link S, X, SX User Manual
Guidelines for Usage of the NanoCom Link S, X, and SX Radio Products

Document reference: MAN 1076241
Source reference: doc-nanocom-link-sx-user-manual
Date: January 23, 2026
Revision: 3.0

Information contained in this document is up-to-date and correct as at the date of issue. As GomSpace A/S cannot control or anticipate the conditions under which this information may be used, each user should review the information in specific context of the planned use. To the maximum extent permitted by law, GomSpace A/S will not be responsible for damages of any nature resulting from the use or reliance upon the information contained in this document. No express or implied warranties are given other than those implied mandatory by law.

GomSpace A/S
Langagervej 6, 9220 Aalborg East
Denmark
Phone: +45 71 741 741
www.gomspace.com

Contents

List of Figures

List of Tables

List of Abbreviations

AES Advanced Encryption Standard.

AES256 Advanced Encryption Standard 256-bit key length.

BLCB bootloader control block.

CAN Controller Area Network.

CCSDS Consultative Committee for Space Data Systems.

CLI command-line interface.

CSP Cubesat Space Protocol.

DVB-S2 Digital Video Broadcasting - Satellite - Second Generation.

eMMC embedded multi-media controller.

ESD electrostatic discharge.

GCM Galois/Counter Mode.

GOSH GomSpace Shell.

GSSE GomSpace Stream Encapsulation.

GSUFTP GomSpace Unidirectional File Transfer Protocol.

ICD interface control document.

IP Internet Protocol.

IPA isopropyl alcohol.

IPv4 Internet Protocol version 4.

IV initialization vector.

LVDS low-voltage differential signal.

MODCOD modulation and coding.

MTU maximum transmission unit.

NOR negated or.

OBC on-board computer.

PC personal computer.

PCBA printed circuit board assembly.

QPSK quadrature phase-shift keying.

RF radio frequency.

RX receive.

SDR software-defined radio.

SGL space-ground link.

SSH Secure Shell.

TCP Transmission Control Protocol.

TCP/IP Transmission Control Protocol/Internet Protocol.

TTY teletypewriter.

TX transmit.

UART universal asynchronous receiver/transmitter.

UDP User Datagram Protocol.

USB universal serial bus.

ZMQ ZeroMQ.

Changelog

Version	Change
3.0	Add section on safe shutdown procedure.
	Add ‘Recovery Boot’ chapter.
2.0	Add ‘RF Link Security’ chapter.
1.2	Update RS-422 interface capabilities to include CSP mode
	Include IPv4 over DVB-S2 in examples including bidirectional link using S-band uplink
	Update examples to new GSUFTP implementation running over UDP
	Update ‘Storing SpaceWire Data’ use-case with to use /data-striped partition
	Add section on ‘Storage’ in ‘System Overview’
1.1	Correct parameters for S-band telemetry example
1.0	Initial revision

Table 1: Changelog

1 Introduction

1.1 Purpose

This document contains guidelines for usage of the GomSpace NanoCom Link S, X and SX radio products. This includes guidelines for getting started, configuration, and operation.

1.2 Scope

The document is applicable within the scope of using the NanoCom Link S, X and SX as satellite radio communications products. The following products are covered in this document

- NanoCom Link S for S-band communications between the space segment and ground segment using the Consultative Committee for Space Data Systems (CCSDS) 131.0-B-3 standard [1].
- NanoCom Link X for X-band communications from the space segment towards the ground segment using the Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) standard [2].
- NanoCom Link SX for combined communications using the S- and X-band variants simultaneously.

Note that technical specifications such as performance characteristics are not included in this document as it is focused on usage. Refer to the datasheet for technical specifications [3].

1.3 Structure

The document is structured as follows:

- Section 2 provides a hands-on description of how to use the product.
- Section 3 provides a system overview.
- Section 4 describes configuration of the product.
- Section 5 describes how the Internet Protocol (IP) network on the product is configured.
- Section 6 describes how to enable and configure the security feature on the S-band radio.
- Section 7 describes a use-case for streaming payload data to/from the ground segment via S-band.
- Section 8 describes a use-case for exchanging files with bi-directional link on NanoCom Link S and SX.
- Section 9 describes a use-case for downloading files via X-band.
- Section 10 describes a use-case for CSP communication using the S-band link.
- Section 11 describes how storage of SpaceWire data link layer frames is configured and operated on the product.

1.4 Related Documents

The user manual is complemented by the NanoCom Link S, X, SX interface control documents (ICDs) [4, 5, 6] and datasheet [3].

2 Getting Started

This section describes the necessary steps for getting started with the NanoCom Link S, X and SX products.

2.1 Unpacking and Handling Precautions

The NanoCom Link products and their components are electrostatic discharge (ESD) sensitive devices. Proper precautions must be observed when handling the devices. Use an ESD mat and a wrist strap as a minimum. Wear gloves to avoid fingerprints on the anodized aluminium parts, as these are particularly difficult to rinse off. If any cleaning of the parts are required prior to flight, use ESD safe cleaning methods with a neutral, non-reactive, isopropyl alcohol (IPA) solvent.

2.2 Included Components

The NanoCom Link products consist of different components depending on the specific product. The different components for each product are listed on Table 2.1.

Component	Included in		
	Link S	**Link X**	**Link SX**
NanoCom Link software-defined radio (SDR) unit	X	X	X
NanoCom ANT2150 antenna	X		X
NanoCom ANT8250 antenna		X	X

Table 2.1: NanoCom Link components included in products

2.2.1 SDR unit

The SDR unit is shown on Figure 2.1.

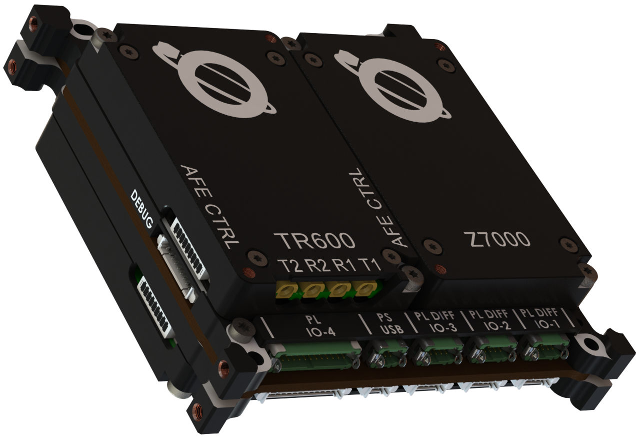


Figure 2.1: NanoCom Link SDR unit

It consists of a main board where a processing module is installed alongside one or more transceiver modules. The processing module is labelled “Z7000” while the transceiver modules are labelled “TR600”. The main board can hold up to three transceiver modules in so-called “slots”.

The SDR unit is delivered with a cable kit which includes

- Power cable
- Universal serial bus (USB) to serial cable
- Debug breakout printed circuit board assembly (PCBA) with debug cable

The usage of the slots is described on Table 2.2.

Transceiver Slot	Usage	Physical Location
A	S and SX products for communication with NanoCom ANT2150	Next to the processing module on the same side of the main board.
B	X and SX products for communication with NanoCom ANT8250	Directly opposite to slot A on the other side of the main board.
C	Not used.	Next to slot B on the same side of the main board.

Table 2.2: NanoCom Link transceiver slot usage and location

2.2.2 NanoCom ANT2150 for S and SX products

The NanoCom ANT2150 is shown on Figure 2.2 and on Figure 2.3 with connectors highlighted.

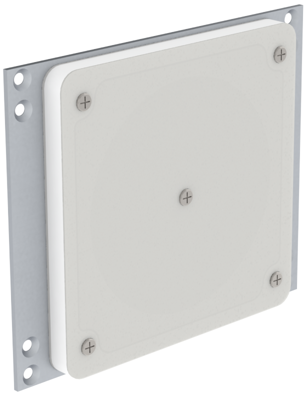


Figure 2.2: NanoCom ANT2150 antenna

It is delivered with a cable kit which includes

- Power cable
- Controller Area Network (CAN) control cable
- Two radio frequency (RF) cables

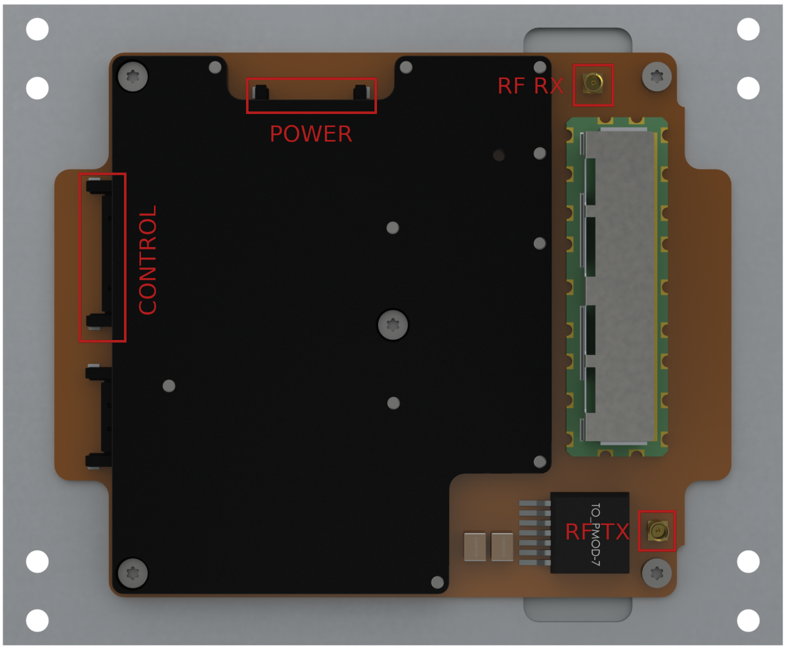


Figure 2.3: NanoCom ANT2150 bottom-view with connectors highlighted

The relevant connectors are described on Table 2.3.

Connector	Description
RF receive (RX)	Connected using one RF cable
RF transmit (TX)	Connected using one RF cable
Power	Connected using power cable
Control	Connected using CAN control cable

Table 2.3: NanoCom ANT2150 connector overview

2.2.3 NanoCom ANT8250 for X and SX products

The NanoCom ANT8250 is shown on Figures 2.4 and 2.5.

These are delivered with a cable kit which includes

- Power cable
- CAN control cable
- RF cable

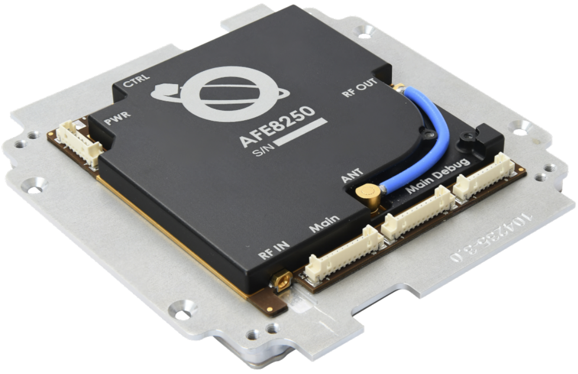


Figure 2.4: NanoCom ANT8250 with AFE8250 RF front-end

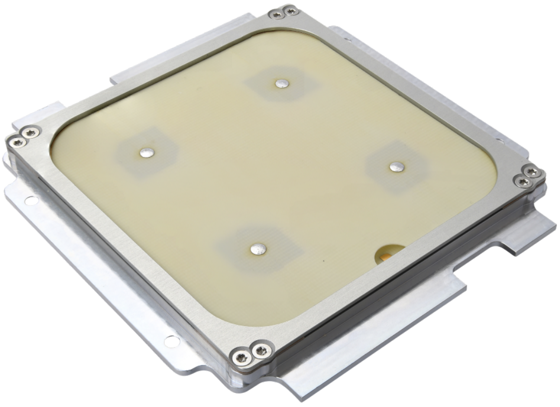


Figure 2.5: NanoCom ANT8250 antenna

2.3 Connecting the Components

The NanoCom Link products must have an RF connection and a CAN connection between the SDR unit and the RF front-end(s).

2.3.1 NanoCom ANT2150 for S and SX products

Use the SDR transceiver module in slot A for connection between the SDR unit and the NanoCom ANT2150. Use the supplied RF cable to connect the transceiver port labelled "TX1" to the NanoCom ANT2150 J300 connector. Use the other supplied RF cable to connect the transceiver port labelled "RX1" to the NanoCom ANT2150 J102 connector. Use the supplied CAN cable to connect the transceiver connector labelled "AFE CONTROL" to the NanoCom ANT2150 J401 connector.

2.3.2 NanoCom ANT8250 for X and SX products

Use the SDR transceiver module in slot B for connection between the SDR unit and the NanoCom ANT8250. Use the supplied RF cable to connect the transceiver port labelled "TX1" to the NanoCom AFE8250 port labelled "RF IN". Use the supplied CAN cable to connect the transceiver connector labelled "AFE CONTROL" to the NanoCom AFE8250 connector labelled "MAIN".

2.4 Powering the Components

2.4.1 SDR unit

The SDR unit requires a single power supply. For stand-alone testing, use the supplied power cable and a lab power supply with current limiting capabilities. Connect the power cable to the connector labelled "PWR1" on the SDR and the other end to the power supply channel. Configure the power supply channel for e.g. 12 V with 1 A current limit. Refer to the NanoCom Link S, X, SX Datasheet[3] for the supported supply voltage range.

The SDR is now ready to be powered on.

It is recommended to do a safe shutdown of the SDR unit before cutting power to avoid potential file system corruption. A safe shutdown can be done remotely using the CSP 'shutdown' service. As an example, using a GomSpace Shell (GOSH) capable application, call the command 'shutdown 13', where '13' is the CSP address of the SDR unit. A safe shutdown can also be done through the debug serial interface, by calling the Linux command-line interface (CLI) command 'shutdown now'.

After issuing a shutdown command, you must wait about 60 s for the SDR unit to complete the shutdown process before cutting power.

2.4.2 NanoCom ANT2150 for S and SX products

The NanoCom ANT2150 requires an 8 V–18 V power supply. For stand-alone testing, use a similar setup as the SDR. That is, use the supplied power cable and a lab power supply with current limiting capabilities. Connect the power cable to the J400 connector on the NanoCom ANT2150 and the other end to the power supply channels. Configure the power supply channels for e.g. 12 V with 1 A current limit.

The NanoCom ANT2150 is now ready to be powered on.

2.4.3 NanoCom ANT8250 for X and SX products

The NanoCom ANT8250 requires a 5 V and a 12 V–33 V power supply. For stand-alone testing, use a similar setup as the SDR. That is, use the supplied power cable and a lab power supply with current limiting capabilities. Connect the power cable to the connector labelled "PWR" on the NanoCom AFE8250 and the other end to the power supply channels. Configure the power supply channels for e.g. 5 V with 1 A current limit and 12 V with 2 A current limit.

The NanoCom ANT8250 is now ready to be powered on.

2.5 Communicating over Debug Serial Interface

The CAN interface is the primary interface for control and telemetry [4]. The serial interface is only used for lab debugging purposes and getting started.

Connect the SDR breakout PCBA with the SDR unit using the supplied debug cable. Use the supplied USB to serial cable to connect your personal computer (PC) to the breakout PCBA. Open the teletypewriter (TTY) device in a serial communication program, e.g., minicom or tio [7, 8]. Configure the serial communication program for 115 200 bit/s, 8 databits, 1 stopbit and no parity. On Listing 2.1 an example is shown using *tio* [8] from a Linux shell. Note there should be no output at this point, as the board is still powered down.

Listing 2.1: Example of SDR serial communication setup with *tio*

```
user@HostPC:~$ tio -b 115200 /dev/ttyUSB0
[tio 11:15:57] tio v1.20
[tio 11:15:57] Press ctrl-t q to quit
[tio 11:15:57] Connected
```

2.6 Bootstrapping the Components

2.6.1 SDR unit

Bootstrap the SDR by enabling the power supply channel and wait for output to appear on the serial connection. The bootstrapping process includes a u-boot console which automatically continues to a Linux startup process. When the bootstrapping process is done a Linux prompt is available on the serial interface. Note that additional time is necessary for all software components to start before the SDR is operational.

Listing 2.2 shows the output on the serial interface that appears when a Linux prompt is available.

Listing 2.2: Output on serial interface when Linux prompt is reached

```
 GomSpace Linux 4.1.4 nanomind-z7030-nv3 ttyPS1

 nanomind-z7030-nv3 login: root (automatic login)

    ____  ___  __  __
   / ___|/ _ \|  \/  |___ _ __   __ _  ___ ___
  | |  _| | | | |\/| / __| '_ \ / _` |/ __/ _ \
  | |_| | |_| | |  | \__ \ |_) | (_| | (_|  __/
   \____|\___/|_|  |_|___/ .__/ \__,_|\___\___|
                         |_|

  GomSpace NanoCom SGL Image

  root@nanomind-z7030-nv3:~#
```

2.6.2 NanoCom ANT2150 for S and SX products

Bootstrap the NanoCom ANT2150 by simply enabling the power supply channels associated with it.

2.6.3 NanoCom ANT8250 for X and SX products

Bootstrap the NanoCom ANT8250 by simply enabling the power supply channels associated with it.

2.7 Enabling RF Transmission

The following section uses the SDR debug serial interface for commanding the unit. In production the CAN interface should be used as described in [4].

To enable RF transmission access the Linux prompt on the SDR and run

```
gosh localhost 5005
```

This opens the GOSH for the space-ground link (SGL) control software. The NanoCom Link products use the GomSpace parameter system for control and telemetry. This is described in detail in [4].

2.7.1 Enabling S-band transmission for S and SX products

Access the parameter table for CCSDS configuration using the `param select` GOSH command. Set the `enable` parameter to `true` using the `param set` GOSH command. This procedure is illustrated on Listing 2.3.

Listing 2.3: Example of enabling S-band transmission over SDR debug serial interface

```
root@nanomind-z7045-nv3:~# gosh localhost 5005
connected to 127.0.0.1:5005
SGL # param select ccsds_base_cfg
SGL # param set enable true
```

The S-band transmission is now active using the default configuration. The configuration can be seen using the `param list` GOSH command as illustrated on Listing 2.4.

Listing 2.4: Example of listing S-band transmission configuration over SDR debug serial interface

```
SGL # param list ccsds_base_cfg
Table ccsds_base_cfg (1):
  0x0000 enable          BL  true
  0x0004 tx_freq         FLT 2200.000000
  0x0008 rx_freq         FLT 2025.000000
  0x000C tx_symrate      FLT 2.000000
  0x0010 rx_symrate      FLT 2.000000
  0x0014 tx_mod          STR "qpsk"
  0x001C rx_mod          STR "qpsk"
  0x0024 tx_pwr_lvl      U8  0
```

A detailed list of configuration parameters and descriptions is available in [4]. To access telemetry use the `param list` GOSH command as illustrated on Listing 2.5.

Listing 2.5: Example of listing S-band transmission telemetry over SDR debug serial interface

```
SGL # param list ccsds_base_telem
Table ccsds_base_telem (4):
  0x0000 bb_running      BL  true
  0x0001 tr_running      BL  true
  0x0002 afe_running     BL  true
  0x0003 afe_temp_off    BL  false
  0x0004 afe_temp_pa     FLT 27.800001
  0x0008 afe_temp_mcu    FLT 22.700001
  0x000C afe_uptime      U32 68
  0x0010 afe_power_in    FLT -12.426064
  0x0014 afe_steady      BL  true
  0x0018 rx_gain         FLT 61.000000
  0x001C tx_gain         FLT -14.250000
  0x0020 rssi            FLT -83.500000
  0x0024 rx_lock         BL  false
  0x0025 rx_sig_pres     BL  false
  0x0020 rx_sig_evm      FLT 15.864301
  0x002C rx_sig_offset   I32 -399
```

A detailed list of telemetry parameters and descriptions is available in [4].

2.7.2 Enabling X-band transmission for X and SX products

Access the parameter table for DVB-S2 configuration using the `param select` GOSH command. Set the `enable` parameter to `true` using the `param set` GOSH command. This procedure is illustrated on Listing 2.6.

Listing 2.6: Example of enabling X-band transmission over SDR debug serial interface

```
root@nanomind-z7045-nv3:~# gosh localhost 5005
connected to 127.0.0.1:5005
SGL # param select dvbs2_base_cfg
SGL # param set enable true
```

The X-band transmission is now active using the default configuration. The configuration can be seen using the `param list` GOSH command as illustrated on Listing 2.7.

Listing 2.7: Example of listing X-band transmission configuration over SDR debug serial interface

```
SGL # param list dvbs2_base_cfg
Table dvbs2_base_cfg (2):
  0x0000 enable          BL  true
  0x0002 modcod          U16 1
  0x0004 tx_freq         FLT 8150.000000
  0x0008 symrate         FLT 50.000000
  0x000C tx_pwr_lvl      U8  0
```

A detailed list of configuration parameters and descriptions is available in [4]. To access telemetry use the `param list` GOSH command as illustrated on Listing 2.8.

Listing 2.8: Example of listing X-band transmission telemetry over SDR debug serial interface

```
SGL # param list dvbs2_base_telem
Table dvbs2_base_telem (5):
  0x0000 mod_enabled     BL  true
  0x0001 if_enabled      BL  true
  0x0002 afe_enabled     BL  true
  0x0004 mod_pl_sent     U32 1
  0x0008 mod_dum_sent    U32 4519248
  0x000C mod_bbf_sent    U32 0
  0x0010 mod_bbp_sent    U32 1
  0x0014 mod_bb_buf      U32 0
  0x0018 mod_bb_proc     U32 4
  0x001C mod_data_halt   BL  false
  0x0020 if_tx_gain      FLT -30.500000
  0x0024 afe_running     BL  true
  0x0025 afe_steady      BL  true
  0x0026 afe_ctrl_st     U16 3
  0x0028 afe_bootcount   U32 118
  0x002C afe_temp_pa     FLT 30.000000
  0x0030 afe_temp_mcu    FLT 29.000000
  0x0034 afe_if_pow      FLT -28.000263
  0x0038 afe_rf_pow      FLT 26.834335
```

A detailed list of telemetry parameters and descriptions is available in [4].

3 System Overview

The NanoCom Link SX product is depicted on Figure 3.1 as a general block-diagram.

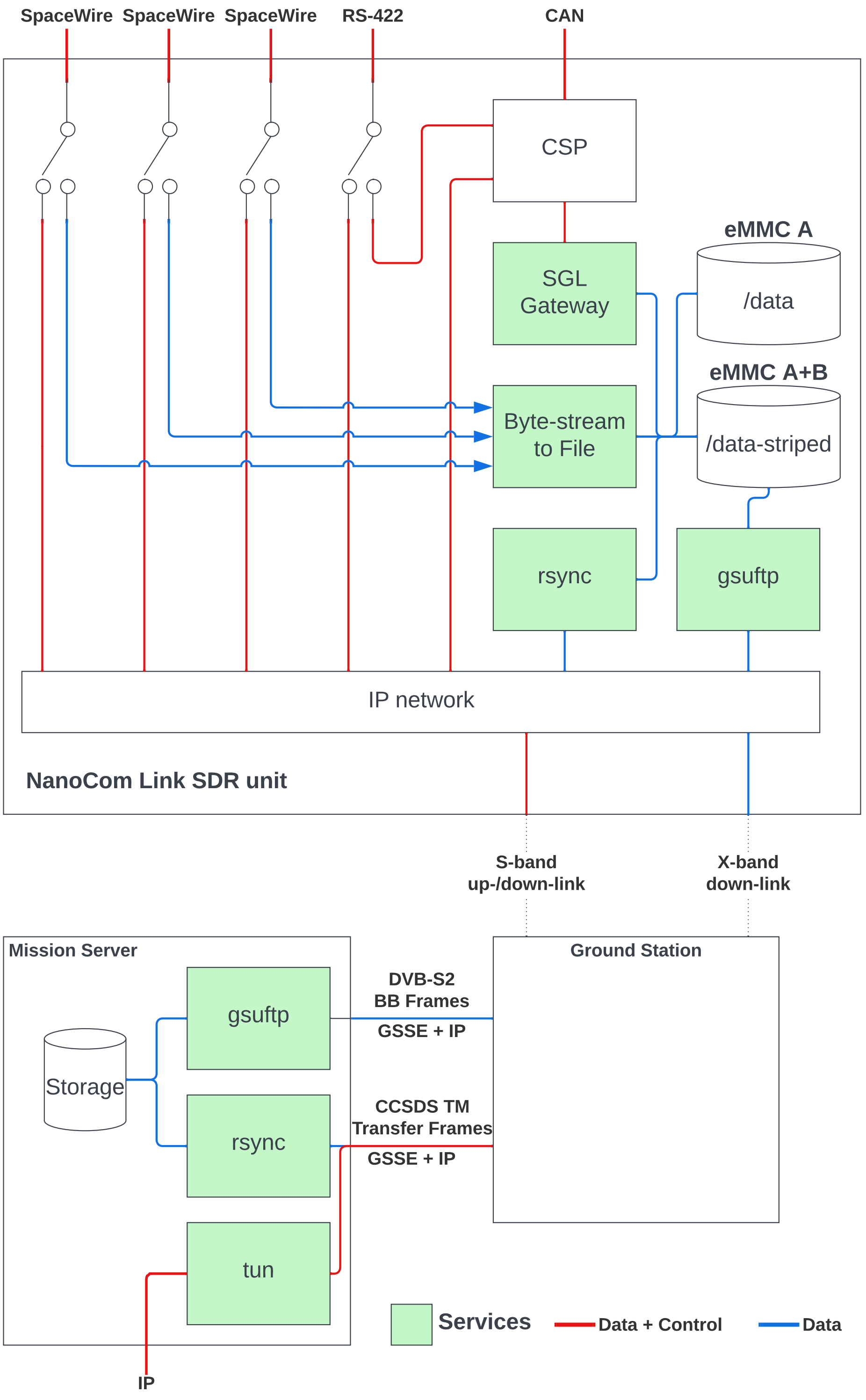


Figure 3.1: NanoCom Link System Overview

3.1 Data Plane

3.1.1 Data Exchange with Spacecraft Bus

The SpaceWire and RS-422 external interfaces are primarily used for data. The details of these interfaces are described in [5]. In summary, these interfaces support data exchange as Transmission Control Protocol/Internet Protocol (TCP/IP). Additionally, the SpaceWire interfaces supports data exchange as a byte-stream to file. On the SDR unit, TCP/IP is realized as a Linux IP TUN interface for ease of integration with third-party Linux tools.

3.1.2 Data Exchange via S-band or X-band with Bidirectional Communication

The interfaces mentioned in Section 3.1.1 can be used for direct, instantaneous transmission of data or, alternatively, to store data for future transmission.

When data is stored for future transmission it is put into files and stored on the SDR file system. These files can e.g. be downlinked with the Linux `rsync` application. These use-cases are described in more detail in the following sections.

Direct transmission is enabled by another IP TUN interface named `rf0`. The IP packets sent on this interface are transmitted over the S-band RF link towards ground which also exposes an IP TUN interface on the mission server. The data-plane is thus composed of multiple interfaces that can all be represented by IP TUN interfaces and interconnected using standard Linux IP routing capabilities. These use-cases are described in more detail in Section 8 and Section 7.

Additionally, both the S-band and X-band RF links support secure communication using Advanced Encryption Standard 256-bit key length (AES256)-Galois/Counter Mode (GCM). This feature is optional but recommended for secure data transmission. The security feature is described in Section 6.

3.1.3 Data Exchange via X-band with Unidirectional Communication

The interfaces mentioned in Section 3.1.1 can be used to store data for future transmission. When data is stored for future transmission it is put into files and stored on the SDR file system.

These files can be downlinked with the GomSpace `gsuftp` application.

This use-case is described in more detail in Section 9.

3.2 Control Plane

The CAN and RS-422 interfaces are the primary control interfaces, although other interface may also be used. This is described in detail in [4]. In summary, the NanoCom Link products use CSP and the GomSpace parameter system. Consequently, the payloads and/or on-board computer (OBC) must support CSP to control and retrieve telemetry from a NanoCom Link product.

3.3 Storage

Two storage locations are available for data storage. The details of the storage is shown in Table 3.1.

Mount Point	Capacity	Physical Media	Read/Write Performance
`/data`	10 GB	eMMC A	low
`/data-striped`	115 GB	eMMC A + eMMC B	high

Table 3.1: Data storage properties

The `/data-striped` storage has the highest performance and capacity. The drawback compared to `/data` is that the data is distributed across two separate physical devices. In case either one of the physical mediums malfunction, the `/data-striped` storage in unavailable.

4 Configuring the Radio Parameters

It is recommended to configure and save the NanoCom Link radio parameters prior to flight. Follow the steps in this section to configure the radio using the SDR debug serial interface.

4.1 Configuring the CCSDS S-band Radio for S and SX products

The following configuration is used as an example.

- 2200 MHz TX frequency
- 2025 MHz RX frequency
- Quadrature phase-shift keying (QPSK) TX modulation
- QPSK RX modulation
- 2 MBd TX symbol-rate
- 2 MBd RX symbol-rate
- 0.5 W TX power

Access the SDR unit Linux terminal using the serial debug interface as described in Section 2. Open the GOSH for the SGL control application:

```
gosh localhost 5005
```

While in the SGL control application GOSH access the parameter table for basic CCSDS S-band radio configuration:

```
param select ccsds_base_cfg
```

Set the parameters:

```
param set tx_freq 2200
param set rx_freq 2025
param set tx_mod qpsk
param set rx_mod qpsk
param set tx_symrate 2
param set rx_symrate 2
param set tx_pwr_lvl 0
```

Save the changes:

```
param save ccsds_base_cfg
```

The changes are now stored in persistent storage. The parameters can be changed any time during the mission including on-orbit. During operation, however, the dedicated control interfaces should be used as described in [4].

4.2 Configuring the DVB-S2 X-band Radio for X and SX products

The following configuration is used as an example.

- 8250 MHz TX frequency
- 30 MBd TX symbol-rate
- 2 W TX power
- DVB-S2 modulation and coding (MODCOD) index 17

Access the SDR unit Linux terminal using the serial debug interface as described in Section 2. Open the GOSH for the SGL control application:

```
gosh localhost 5005
```

While in the SGL control application GOSH access the parameter table for basic DVB-S2 X-band radio configuration:

```
param select dvbs2_base_cfg
```

Set the parameters:

```
param set tx_freq 8250
param set symrate 30
param set tx_pwr_lvl 2
param set modcod 17
```

Save the changes:

```
param save dvbs2_base_cfg
```

The changes are now stored in persistent storage. The parameters can be changed any time during the mission including on-orbit. During operation, however, the dedicated control interfaces should be used as described in [4].

5 IP Networking

The NanoCom Link S, X and SX products have multiple data interfaces that support IP based communication. This section describes how to leverage this to build an end-to-end IP network that interconnects all nodes in the space segment with the ground segment through the NanoCom Link S or SX.

Figure 5.1 shows an overview of the IP network.

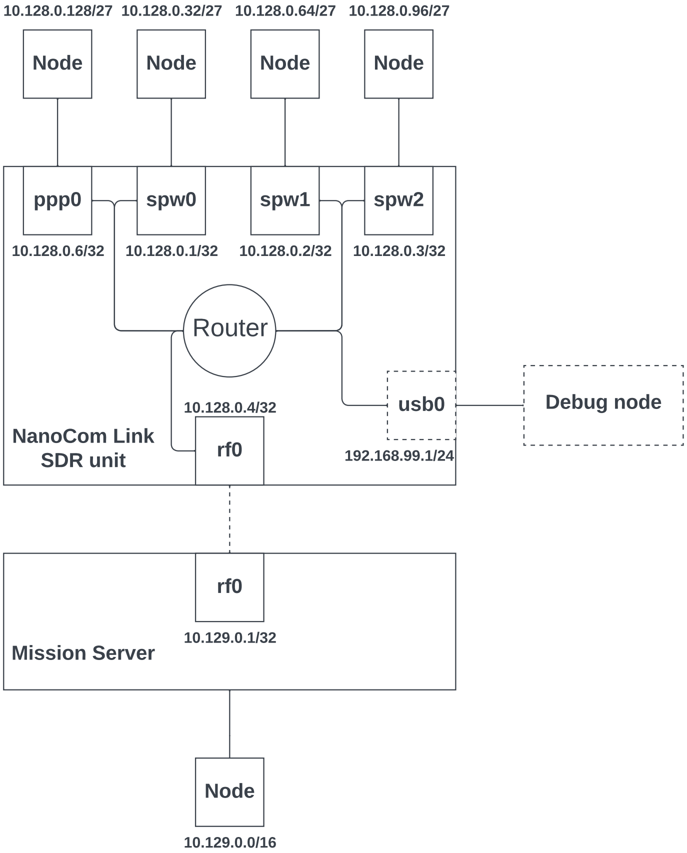


Figure 5.1: IP network of the NanoCom Link system

5.1 Interfaces with IP Capabilities

Any interface in the NanoCom Link products that supports IP is accessible through a Linux IP TUN network interface on the SDR unit. This means any Linux utility that uses an IP connection such as `rsync` [9, 10] and `ssh` [11, 12] can operate over any given interface seamlessly.

Table 5.1 lists the IP TUN interfaces available on NanoCom Link products. The SpaceWire interfaces automatically detect if there is a working connection to the other end, and set the TUN network interface up/down accordingly. The `rf0` interface can operate in both one-way and two-way links. With one-way links, the interface has no way of knowing if the link is established on the receiver in the other end. To accommodate one-way links, the `rf0` interface is always reported as up.

Interface Name	Description
spw0	Interface using SpaceWire on PL DIFF IO-1 connector
spw1	Interface using SpaceWire on PL DIFF IO-2 connector
spw2	Interface using SpaceWire on PL DIFF IO-3 connector
ppp0	Interface using RS-422 on COM1/COM2 connector
rf0	Ingress and egress IP data to/from the radio link

Table 5.1: IP interfaces available on NanoCom Link

5.2 Addresses and Routing in Space Segment

All interfaces are configured with a default Internet Protocol version 4 (IPv4) address and routing. The interfaces are configured to act like bridges by default thus forwarding any IPv4 packets with a destination address in their subnet. The default addresses and routes are listed on Table 5.2.

Interface Name	IP address	Routing	MTU
spw0	10.128.0.1/32	10.128.0.32/27	9000
spw1	10.128.0.2/32	10.128.0.64/27	9000
spw2	10.128.0.3/32	10.128.0.96/27	9000
ppp0	10.128.0.6/32	10.128.0.128/27	1500
rf0	10.128.0.4/32	10.129.0.0/16	9000

Table 5.2: IP configuration

Consider, as an example, the `spw0` interface which is configured with an IPv4 address of `10.128.0.1`. This interface is, by default, configured to route any IPv4 packet in the `10.128.0.32/27` subnet. This means any IPv4 packet with a destination in the range `10.128.0.32` to `10.128.0.63` is forwarded over this interface. Any IPv4 enabled payload, connected to the SpaceWire interface via the PL DIFF IO-1 connector, should be configured with an IPv4 address in the `10.128.0.32/27` network.

Note that the `rf0` interface is configured to route any IPv4 packet with a destination in the `10.129.0.0/16` subnet. This subnet is reserved for nodes in the ground-segment while the `10.128.0.0/16` subnet is reserved for nodes in space-segment.

5.3 Addresses and Routing in Ground Segment

The ground segment must run the NanoCom Link Connect software stack or a custom software application that receives the GomSpace Stream Encapsulation (GSSE) byte stream and extracts the IP packets.

NanoCom Link Connect creates a Linux IP TUN network interface called `rf0`. The `rf0` interface on the mission-server acts as a point-to-point connection with the `rf0` interface on the SDR unit. The `rf0` interface on the mission-server is configured, by default, with an IPv4 address of `10.129.0.1`. In addition, it is configured to route any IPv4 packets with a destination in the `10.128.0.0/16` subnet.

The resulting IP network allows for the mission server to reach e.g. a payload connected to the `spw0` interface on the SDR by connecting to `10.128.0.33` (assuming the payload is configured with this address).

Similarly the payload can reach the mission server by connecting to `10.129.0.1`. In addition, any payload can communicate with another in the space-segment through the SDR.

Note that this section only considers the network layer for each interface. For a detailed description of the protocol stack associated with each interface see [5, 6].

6 Security on RF Link

As described in Section 3, NanoCom Link supports secure communication over the S- and X-band links. This chapter describes the practical concepts of the security feature on the NanoCom Link S- and X-band radios.

The NanoCom Link S-band radio utilizes AES256-GCM encryption and authentication, while the X-band radio only supports encryption, since it can only be used for downlink. AES256-GCM is a symmetric encryption algorithm that provides both confidentiality and integrity through the use of a 256-bit key and a 128-bit authentication tag. A 96-bit initialization vector (IV) with an incrementing counter ensures unique outputs for each packet. As AES256-GCM is a symmetric encryption algorithm, the same key must be shared between the NanoCom Link and the ground counterpart (e.g. NanoGround Link Connect).

The crypto protocol is implemented in the data link layer of the S- and X-band protocol stacks. As such, the entire IPv4 packet, including the headers, is secured by Advanced Encryption Standard (AES)-GCM. The encrypted packets are encapsulated with the key index, IV and authentication tag. See the NanoCom Link S, X, SX RF Interface [6] for additional information on the crypto protocol in the S- and X-band protocol stacks.

The security feature is transparent to the user, meaning all services and protocols are operated the same way, regardless of whether the security feature is enabled or not. Details on the background and theory of the security feature is provided in NanoCom Link and AX2150 Information Security [13].

6.1 Key Management

A critical aspect to the security of the AES256-GCM encryption is the management of the encryption keys. A detailed description of background for key management is provided in NanoCom Link and AX2150 Information Security [13]. This section provides a description from a practical perspective on how to manage the keys for the radio.

6.1.1 Master Keys

The security feature uses master keys as the basis for all cryptographic operations. The master keys are 256-bit keys that are used to derive the session keys used for encryption and decryption.

Master keys are loaded into the radio before launch, as a pre-shared secret between the radio and the ground counterpart. After launch, no new master keys can be loaded into the radio.

As a fallback mechanism, master keys can be used for encryption and decryption, if no session keys are available. However, the nominal operation is that master keys are only used to derive session keys.

Each master key is identified by a key index, which is an integer value between 1 and 65534.

6.1.2 Session Keys

Session keys are derived from a master key. The session keys are used for encryption and decryption of the data packets.

Unlike master keys, session keys are derived continuously during operation.

Each session key is identified by a key index, which is an integer value between 1 and 65534. The session key index is used as an input to the key derivation function that derives the session key from the master key.

6.1.3 Invocation Counter

Each key, both master and session keys, has an associated invocation counter. The invocation counter is a 64-bit unsigned integer that is incremented each time the key is used for encryption. The invocation counter is used to ensure that the same key is not used more than once with the same IV, which would compromise the security of the encryption. The stored invocation counter for encryption is incremented after each use of the key, while the decrypt side pulls the invocation counter from the received packet if the packet is successfully authenticated. When a key is selected for encryption, the invocation counter is incremented with a safety margin to ensure that crashes or reboots do not cause the same IV to be used again.

Note that the invocation counter is not increased, when a master key is used to derive a session key.

The invocation counter is transmitted in clear-text as part of the IV in each encrypted packet.

6.1.4 Protection Against Replay Attacks

The invocation counter is implemented, as part of an anti replay mechanism. When a packet is received, the invocation counter included in the packet is compared to the last invocation counter seen for that key. The packet is only accepted if the counter is strictly greater than last recorded for that key. If the invocation counter is less than or equal to the last seen value, the packet is dropped, a warning is logged and relevant telemetry is updated.

6.1.5 Key States

Each key is attributed a state, which defines how the key can be used. The key states are defined based on CCSDS Magenta Book 354.0-M-1 Symmetric Key Management [14].

The key states are:

- **Pre-operational:** The key is available, but has not yet been used. This is the initial state of a key after it has been loaded or derived.
- **Active:** The key is in active use, either for encryption/decryption or for deriving session keys.
- **Deactivated:** The key has been de-activated. It is no longer available for derivation or encryption/decryption.
- **Suspended:** The key is suspended, unavailable for encryption/decryption and derivation. The key can be re-activated by the operator.
- **Destroyed:** A key can only transition into the destroyed state, as this action removes all information about the key from the system.

NOTE: When a master key is destroyed, any session keys that reference it will have their `parent_id` field nullified, ensuring that the association to the destroyed master key is also removed. Subsequently, the operator must decide how to handle the resulting orphaned session keys.

The following conditions will automatically change the state of a key:

- A pre-operational key will transition to the active state, when it is used for encryption/decryption or for deriving session keys.
- An active key will transition to the suspended state, when the invocation counter is close to the maximum value.
- An active key will transition to the suspended state, when an issue with the key in the crypto engine is detected, e.g. the key data has been corrupted.

All other state transitions are initiated by the operator.

Figure 6.1 shows the valid state transitions for a key. The valid state transitions are aligned with CCSDS Magenta Book 354.0-M-1 Symmetric Key Management [14].

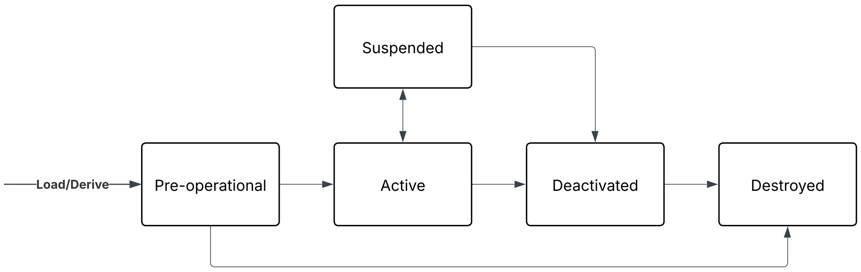


Figure 6.1: Key state transitions.

6.1.6 Automatic Key Rollover

To ease the operational burden of key management, the ‘crypto‘ system support automatic key rollover. Automatic key rollover means that the system will automatically take new keys into use for encryption, when the current key is no longer available.

The selection of a new key is based on the following list of priorities. Each condition is followed in order, until a single key is found.

1. A session key in the active state with the highest invocation counter.
2. A session key in the pre-operational state with the lowest key index.

When an eligible session key is found, the invocation counter is checked to ensure it does not exceed the key invocation suspension threshold (including the safety margin). If the threshold is exceeded, the key is suspended and the next eligible key is checked. The key invocation suspension threshold is configurable by the operator through the ‘ic_threshold‘ parameter in NanoCom Link, and the ‘key_invocation_suspension_threshold‘ attribute in each adapter in NanoGround. Note that the invocation count of the active encryption key is continuously monitored as well.

In case no eligible session key is found, the downlink (encryption) is blocked. The operator must provide an eligible session key by deriving new keys, or re-activating a suspended key. As a fallback mechanism, the operator can choose a specific master key index for encryption. Master keys are never selected automatically for encryption.

6.2 Key Storage

The keys are stored in non-volatile memory in the radio. Each key has associated metadata, including the key index, state and invocation counter.

The keys are stored in a keystore. Individual keystores exists for uplink and downlink for each radio. Keys are not shared between keystores, meaning that there are unique master and session keys for each of the keystores.

Each keystore contains three copies of each key object. The three copies are used to recover from corruption of the key data automatically.

Corruption is detected on each key object periodically, and when a key is used for encryption/decryption or derivation.

Each keystore can store up to 512 keys at a time.

6.3 Available Decryption Keys

Due to size constraints in the crypto engine, only a limited number of keys can be used for decryption at the same time. Up to 64 keys are available for decryption on the S-band link.

All active or pre-operational master and session keys in the uplink keystore are eligible for decryption. Consequently, if all session keys in the uplink keystore are exhausted, master keys can be used as a fallback. It should be noted, however, that master keys will only be used for decryption if they have been manually selected for encryption in the ground counterpart.

If the number of eligible keys exceeds the maximum number of keys available for decryption, the keys are sorted based on the following priorities, and the keys with the lowest priority are removed from the active decryption list.

1. Session keys
2. Keys with the highest invocation counter
3. Active keys (not pre-operational)
4. Keys with the lowest key index

6.4 Preparing master keys

Master keys must be loaded into the ‘crypto‘ system before the security feature can be used. The first step is the generate master keys. For this purpose, a key generation tool ‘gs_key_transit‘ is provided with the delivery. The tool generates cryptographically secure random keys, and outputs the keys in a format that can be loaded into the system.

To generate a master key, run the ‘gs_key_transit‘ tool. The tool will prompt for a passphrase to protect the master key, until it is loaded into the system. The passphrase must be at least 20 characters long. The user input will not be echoed to the terminal for security reasons. Next, the tool will prompt for a keystore that the key is to be loaded into. Select the keystore based on the intended use of the key:

- s_up: S-band uplink (ground to satellite)
- s_down: S-band downlink (satellite to ground)
- x_down: X-band downlink (satellite to ground)
- ax_up: AX2150 uplink (ground to satellite)
- ax_down: AX2150 downlink (satellite to ground)

Finally, the tool will prompt for a key index for the master key. This index must be unique within the selected keystore.

An example of generating a master key with index 1 for the S-band uplink keystore is shown below. Note the command output in the example below is truncated, to avoid presenting a real key.

```
$ gs_key_transit
Enter a passphrase (at least 20 characters):
Repeat the passphrase:
Enter the keystore the master key should be stored in (s_up, s_down, x_down, ax_up, ax_down):
 s_up
Enter the key ID (16-bit decimal unsigned integer, or leave blank to generate a random ID): 1
Assigned key ID: 1
GOSH command to execute to load the key:
        crypto load_key 0001000157d6ba88a8f50ef391a753cbb3c14fcbf66f0ddc0c79ba3078c6357
```

The keys are loaded using the GOSH command line interface of the system.

Call crypto command to set the expected passphrase.

```
crypto passphrase
```

Write the passphrase used when generating the key. The input will not be echoed to the terminal for security reasons. The passphrase remains active until the GOSH application is restarted.

Next, execute the command output by the ‘gs_key_transit’ tool to load the key into the system.

```
crypto load_key 0001000157d6ba88a8f50ef391a753cbb3c14fcbf66f0ddc0c79ba3078c6357
```

To verify that the key has been loaded correctly, use the following command to list the keys in the S-band uplink keystore.

```
crypto list_keys s_up
```

Once all master keys in the keystore have been loaded, the loading of new master keys in the keystore can be disabled, by freezing the keystore.

```
crypto freeze s_up
```

Freezing the keystore is a permanent action, and the only way to load new master keys into the keystore is by wiping all the keys in the system:

```
crypto wipe_all_keys
```

6.5 Deriving session keys

Under nominal operation, session keys are used for encryption and decryption of data packets. Session keys are derived from master keys, and the operator must initiate the derivation of new session keys.

Session keys can be derived using the ‘crypto derive_key’ command in GOSH, or via the ‘rcrypto derive_key’ command in a GOSH application in the ground segment, which includes the ‘rcrypto’ client commands.

In this example, a session key with index 10 is derived from the master key with index 1 in the S-band uplink keystore, using GOSH.

```
crypto derive_key s_up 1 10
```

The requested keystore name, session key index, and master key index are used as input to the key derivation function that generates the session key. This allows you to derive the same session key on both the radio and the ground counterpart, as long as the same master key is used.

To verify that the session key has been derived correctly, use the following command to list the keys in the S-band uplink keystore.

```
crypto list_keys s_up
```

6.6 Operational Workflows

This section describes the typical workflows for operating the encryption features. This includes the actions to be taken before launch, as well as during operations after launch.

6.6.1 Before Launch

The following actions should be done before launch through the GOSH CLI:

- Generate master key(s) using ‘gs-key-transit‘ as described in Section 6.4.
- Use ‘crypto passphrase‘ to set the keystore passphrase (should match what is used in ‘gs-key-transit‘).
- Use ‘crypto load_key‘ to load the master key(s) into the keystore.
- Use ‘crypto freeze‘ to permanently freeze a keystore from loading new master keys (deriving keys is still allowed).
- Use ‘crypto derive_key‘ to derive initial session key(s) from the master key(s).

All operations should be performed on both the ground **and** the radio counterpart to ensure that both sides have the same keys available.

- Use ‘crypto load_key‘ on the radio to load the same master key(s) into the radio’s keystore.
- Use ‘crypto freeze‘ on the radio to permanently freeze a keystore from loading new master keys (deriving keys are still allowed).
- Use ‘crypto derive_key‘ on the radio, or ‘rcrypto derive_key‘ to derive the same initial session key(s) from the master key(s).

Note that key derivation is deterministic, as long as the same arguments are used for the ‘load_key‘ and ‘derive_key‘ commands.

6.6.2 After Launch

If enabled, encryption is automatically used for all radio communications with the satellite. Key management is handled by the operator as needed.

Below are a few scenarios expected to be common during operations.

Creating and Using New Session Keys
In the case where new session keys are to be used, the operator must derive new session keys from a master key on both sides. This is done by first deriving a new key on both ground and on the radio using 'crypto' and 'rcrypto'. Say the current state of keys on both sides is as shown in Listing 6.1.

Listing 6.1: Active session keys on both radio and ground.

```
Radio 's_up' Keystore
ID          Type     State           Parent ID   Invocation Count
---------------------------------------------------------------------
1           master   active          0           0
2           session  active          1           0
---------------------------------------------------------------------

Ground 's_up' Keystore
ID          Type     State           Parent ID   Invocation Count
---------------------------------------------------------------------
1           master   active          0           0
2           session  active          1           0
---------------------------------------------------------------------
```

In Listing 6.2, a new session key (3) is derived from the active master key (1) on both sides. Note that the order of operations here is important to avoid losing connectivity. After these operations, session key 2 should be deactivated on both sides, and the new session key 3 is in the preoperational state. Preoperational session keys are automatically used by the system if no other active session keys are available, so the resulting state of the keystores should be as depicted in Listing 6.3.

Listing 6.2: Deriving and activating new session keys on both sides.

```
# Derive a new key on ground
GOSH # crypto derive_key s_up 1 3

# Derive a new key on the radio
GOSH # rcrypto derive_key 13 s_up 1 3

# Deactivate the old session key (2) on the radio
GOSH # rcrypto change_state 13 s_up 2 deactivated

# Deactivate the old session key (2) on ground
GOSH # crypto change_state s_up 2 deactivated
```

Listing 6.3: New session keys.

```
Radio 's_up' Keystore
ID          Type     State           Parent ID   Invocation Count
---------------------------------------------------------------------
```

```
1           master   active          0           0
2           session  deactivated     1           0
3           session  active          1           0
---------------------------------------------------------------------

Ground 's_up' Keystore
ID          Type     State           Parent ID   Invocation Count
---------------------------------------------------------------------
1           master   active          0           0
2           session  deactivated     1           0
3           session  active          1           0
---------------------------------------------------------------------
```

Note that you are not limited to registering a single session key, multiple session keys can be derived and activated as needed. You can even have multiple preoperational session keys available, and the system will automatically select which one to use as described in Section 6.1.

No Session Keys Left

Only active and pre-operational session keys are used for encryption in ‘auto‘ mode. Active and pre-operational master keys are available for decryption. In a scenario where there are no session keys available, any communications with a radio is done in the blind with no feedback on whether the packets are received or not. This is a very undesirable state to be in, and the operator should immediately derive new session keys from a master key on both sides as described in Listing 6.2. Note that since no session keys are active, the ‘rcrypto‘ command on the radio side is not acknowledged, and the operator must assume it was successful.

Handling Compromised Keys

If a key is suspected or confirmed to be compromised, the operator has several options for mitigating the risk:

- **Suspend the key:** The key is made unavailable, but can be re-activated by the operator if needed.
- **Deactivate the key:** The key is made irreversibly unavailable, but will still exist in the filesystem until it is destroyed.
- **Deactivating and destroying the key:** The key and all associated information are permanently deleted from the system.

It is recommended to either deactivate or destroy compromised keys whenever possible to prevent any future use. Leaving compromised keys in a suspended state increases the risk of accidental use.

Note that destroying a master key will also remove the association to any session keys derived from it, as shown in Listing 6.4.

Listing 6.4: Destroying a master key with children.

```
LINK # crypto list_keys s_up
Found 5 keys in 's_up'

ID          Type     State           Parent ID   Invocation Count
---------------------------------------------------------------------
1           session  preoperational  1234        0
```

```
2           session  preoperational  1234        0
3           session  preoperational  1234        0
4           session  preoperational  1234        0
1234        master   active          0           0
---------------------------------------------------------------------
LINK # crypto change_state s_up 1234 deactivated
LINK # crypto change_state s_up 1234 destroyed
LINK # crypto list_keys s_up
Found 4 keys in 's_up'

ID          Type     State           Parent ID   Invocation Count
---------------------------------------------------------------------
1           session  preoperational  0           0
2           session  preoperational  0           0
3           session  preoperational  0           0
4           session  preoperational  0           0
---------------------------------------------------------------------
```

6.7 Enabling the security feature

Each uplink and downlink for each of the S- and X-band links can be configured to use the security feature independently. The security feature is enabled through the 'ccsds_crypto_cfg' and 'dvbs2_crypto_cfg' parameter tables. Refer to [4] for details on the paramter tables.

'encrypt' and 'decrypt' parameters in the tables are used to enable encryption and decryption, respectively. In addition, a 'encrypt_key' parameter is used to either select a specific key to use for encryption, or to enable automatic key selection and rollover.

The following example configures the S-band link to enable encryption and decryption, using automatic key rollover for encryption. The configuration is then saved to make it persistent across reboots.

The example uses GOSH of the Link gateway application, but the same configuration can be done using the 'rparam' commands in a GOSH application in the ground segment.

```
param select ccsds_crypto_cfg
param set encrypt 1
param set decrypt 1
param set encrypt_key 0
param save ccsds_crypto_cfg
```

Given that the same keys are loaded/derived on both the NanoCom Link and the ground counterpart, the security feature is now enabled, and secure communication can take place over the S-band link.

6.8 Telemetry

Status of the security feature is available as telemetry parameters in the ‘ccsds_crypto_telem’ and ‘dvbs2_crypto_telem’ parameter tables. Refer to [4] for details on the parameter tables.

In addition to these parameter tables, the status of the crypto service is available in the ‘system_telem’ parameter table. Parameters in this table includes a count of logged warnings and errors related to the crypto service.

The log files are located in ‘/data/log/link-crypto-service.log’ on the SDR unit.

7 Streaming Data over S-band

The following example describes how to configure and operate the NanoCom Link S and SX for a data streaming use-case. The example is depicted on Figure 7.1. As indicated on the figure, the example assumes that the payload is connected to the radio using SpaceWire. Note, that it is possible to achieve similar functionality using other data interface [5].

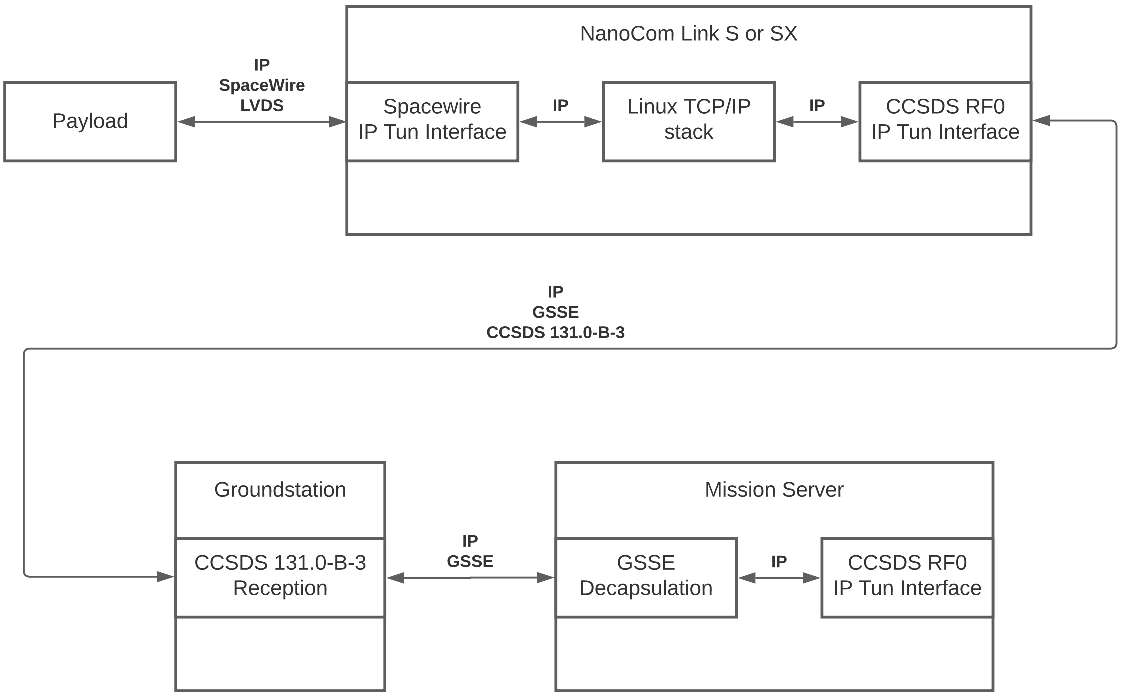


Figure 7.1: Overview of data streaming over S-band example

The example utilizes that the NanoCom Link S/SX supports an IP network layer on top of the SpaceWire data-link layer. This means the payload can send IP packets with a destination address belonging to the mission server. The Linux TCP/IP stack on the SDR unit routes the received packets to the `rf0` TUN network interface which transmits them towards ground. On ground, the packets are received and forwarded to the mission server which has a corresponding TUN network interface.

7.1 Payload Prerequisites

The payload must support low-voltage differential signal (LVDS) and SpaceWire data-link layer. In addition, it must be capable of running an IP network layer on top of the SpaceWire data-link layer. Finally, it must be configured with an IPv4 address that matches the subnets described in Section 5. Refer to [5] for details regarding the SpaceWire interface protocol stack.

7.2 Configuration

On the NanoCom Link SDR unit the SpaceWire network layer is managed by `systemd`. The configuration files are located in `/etc/systemd/network/`. The files are named according to their interface, i.e, `spw0.network`, `spw1.network`, and `spw2.network`. By default the files contain IPv4 addresses and routing setups as described in Section 5. It is recommended to use the defaults but the configuration can be changed to fit mission requirements.

7.3 CCSDS S-band Radio Configuration

The NanoCom Link S/SX must be configured according to the allocated radio channel, bandwidth, etc. as described in Section 4.1. In addition, the `rf0` tun interface must be configured with an IP address. Similar to SpaceWire, the configuration files are located in `/etc/systemd/network/` where the `rf0` config file is named `rf0.network`. The default address and routing can be used which is recommended.

7.4 Ground-station Configuration

The ground-station must be configured to match the chosen radio configuration. That is, frequency, symbol-rate, modulation, etc. Apart from this, the configuration depends entirely on the ground-station chosen for the specific mission. See the NanoCom Link S, X, SX RF ICD for details [6]. Note that the ground-station must be configured to forward received CCSDS 131.0-B-3 frames to the mission server as illustrated on Figure 7.1.

7.5 Mission Server Configuration

The mission server must run the NanoCom Link Connect software stack or a custom software application that receives the GSSE byte stream and extracts the IP packets. This application must also connect to the ground-station to transmit IP packets. The NanoCom Link Connect software stack creates an IP TUN interface named `rf0` which acts as access point to the CCSDS S-band uplink and the satellite. If default addresses and routing are used in the space segment this interface requires no configuration. Otherwise, configure the `rf0` interface IP address and ensure IP traffic is routed correctly.

7.6 Operation

During a link opportunity power on the NanoCom Link SDR unit and NanoCom ANT2150. Set the `enable` parameter to `true` in the CCSDS config table as described in Section 2.7.1. Observe the CCSDS S-band radio telemetry to ensure RX signal lock has been required. When the RX signal lock has been acquired the link is ready for user data transfer. On the payload, send data to the mission server IP address configured on the `rf0` interface. On the ground, send data to the payload IP address.

8 Transferring Files over S-band/X-band Bidirectionally

The following example describes how to configure and operate the NanoCom Link S and SX for a bi-directional file transfer use-case. The example is depicted on Figure 8.1.

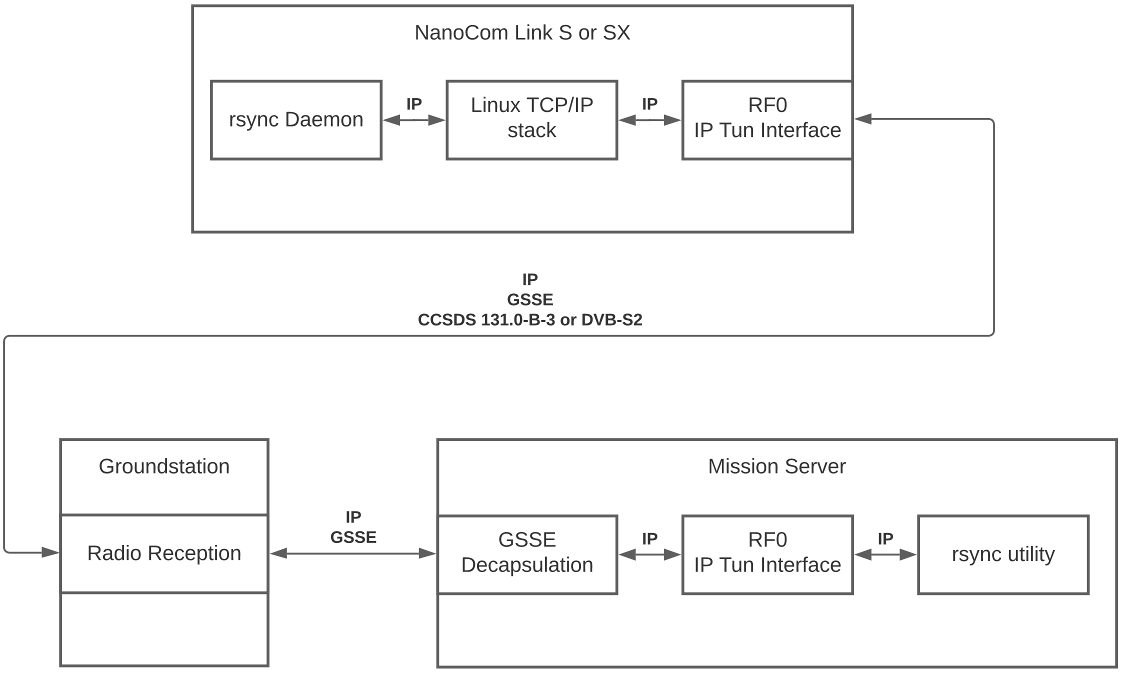


Figure 8.1: Overview of file transfer over S-band example

Note that files are transferred between the storage available on the NanoCom Link S/SX and ground.

8.1 Radio Configuration

The NanoCom Link S-band radio must be configured according to the allocated radio channel, bandwidth, etc. as described in Section 4.1. If available, the X-band radio must be configured according to the allocated radio channel, MODCOD, bandwidth, etc. as described in Section 4.2.

In addition, the `rf0` tun interface must be configured with an IP address. Similar to SpaceWire, the configuration files are located in `/etc/systemd/network/` where the `rf0` config file is named `rf0.network`. The default address and routing can be used which is recommended.

8.2 Ground-station Configuration

The ground-station must be configured to match the chosen radio configuration(s). That is, frequency, symbol-rate, modulation, etc. Apart from this, the configuration depends entirely on the ground-station chosen for the specific mission. See the NanoCom Link S, X, SX RF ICD for details [6]. Note that the ground-station must be configured to forward received CCSDS 131.0-B-3 frames to the mission server as illustrated on Figure 7.1.

8.3 Mission Server Configuration

The mission server must run the NanoCom Link Connect software stack or a custom software application that receives the GSSE byte stream and extracts the IP packets. This application must also connect to the ground-station to transmit IP packets. The NanoCom Link Connect software stack creates an IP TUN interface named `rf0` which acts as access point to the CCSDS S-band uplink and the satellite. If default addresses and routing are used in the space segment this interface requires no configuration. Otherwise, configure the `rf0` interface IP address and ensure IP traffic is routed correctly.

8.4 Operation

During a link opportunity turn on the NanoCom Link SDR unit and NanoCom ANT2150 (and NanoCom ANT8250 if available). Set the `enable` parameter to `true` in the CCSDS and DVB-S2 config tables as described in Sections 2.7.1 and 2.7.2. Observe the CCSDS S-band radio telemetry to ensure RX signal lock has been required. When the RX signal lock has been acquired the link is ready for file transfer.

File transfer can be done using the `rsync` application [9]. As an example, consider Listing 8.1 where a local folder is uplinked to the NanoCom Link SDR unit using `rsync`. In this example, the SDR unit has an IPv4 address of `10.128.0.4` on the `rf0` IP TUN interface. The `rsync` command features multiple useful options for this use-case which are summarized on Table 8.1.

Listing 8.1: Example of file transfer using `rsync`

```
user@hostname:~# rsync -a some_data_dir/ root@10.128.0.4:/data/some_data_dir/
```

Option	Description
`--append`	This causes `rsync` to update a file by appending data onto the end of the file, which presumes that the data that already exists on the receiving side is identical with the start of the file on the sending side.
`--append-verify`	This works just like the `--append` option, but the existing data on the receiving side is included in the full-file checksum verification step, which will cause a file to be resent if the final verification step fails.
`--remove-source-files`	This tells `rsync` to remove from the sending side the files (meaning non-directories) that are a part of the transfer and have been successfully duplicated on the receiving side.

Table 8.1: Highlighted `rsync` options from the manpage [9]

By default, the S-band downlink is used. If an X-band downlink is available this may be used instead of the S-band downlink. Set the `dl_select` parameter in the TUN config table to `highspeed` to use the X-band downlink.

9 Transferring Files over X-band Unidirectionally

Transferring from the space craft to the ground segment via X-band is a uni-directional operation when no S-band uplink is available. To reliably transmit files, the utility GomSpace Unidirectional File Transfer Protocol (GSUFTP) is available. This section focuses on an example use-case for GSUFTP. The use-case is illustrated on Figure 9.1.

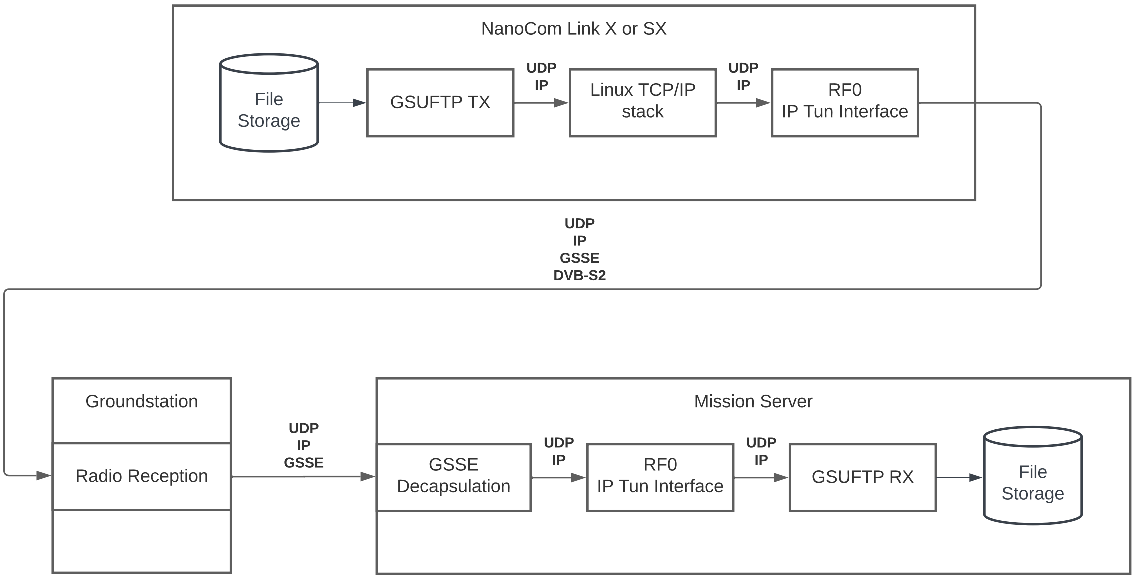


Figure 9.1: Overview of file transfer over X-band example

9.1 DVB-S2 X-band Radio Configuration

The NanoCom Link X/SX must be configured according to the allocated radio channel, MODCOD, bandwidth, etc. as described in Section 4.2.

9.2 GSUFTP Transmission Configuration

Similar to the radio configuration, the GSUFTP configuration is performed in a parameter table. The transmission must be configured with a destination IP address matching the IP address of the mission server. In addition, it is possible to configure which directories to transmit files from.

9.3 Ground-station Configuration

The ground-station must be configured to match the chosen radio configuration. That is, frequency, symbol-rate, etc. Apart from this, the configuration depends entirely on the ground-station chosen for the specific mission. See the NanoCom Link S, X, SX RF ICD for details [6]. Note that the ground-station must be configured to forward received DVB-S2 base-band frames to the mission server as illustrated on Figure 9.1.

9.4 Operation

During a link opportunity turn on the NanoCom Link SDR unit and NanoCom ANT8250. Set the `enable` parameter to `true` in the DVB-S2 config table as described in Section 2.7.2. Set the `enable` parameter to `true` in the GSUFTP config table. Files are now being transmitted from the directories specified during configuration.

9.5 Splitting File Data Ahead of Down-link Opportunity

GSUFTP verifies integrity of down-linked data on a file-by-file basis. This means that any lost data requires a re-transmission of the entire file. If the payload data to be down-linked is large, it may be a benefit to first split the data in several files. GSUFTP has utilities to split, and later combine, large files to/from smaller files.

Using GSUFTP split requires executing commands on the Linux shell of the NanoCom Link SDR unit. In testing and lab conditions, this can be done through the terminal available on the serial debug interface.

In operation, this can be done be using the Nanomind HP Gateway *cmd run* CSP command, refer to NanoMind HP - Gateway Interface[15]. Alternatively, commands can be executed via Secure Shell (SSH) access on an IP capable interface.

To split all files in a directory, run the following command:

```
$ gsuftp-split /data-striped/download/* /data-striped/download-split
```

This splits all files in `/data-striped/download/` into smaller files, copying them to the output directory `/data-striped/download-split/`.

9.6 Clean-up

The files down-linked from the NanoCom Link SDR unit are not automatically removed from the source. Once the integrity of the files received on ground is verified, the source files on the spacecraft can be deleted.

10 CSP over S-band

NanoCom Link supports CSP over S-band. This chapter describes an example of CSP communication using the NanoCom Link S-band radio.

10.1 Overview

The components involved this example use-case are shown in Figure 10.1.

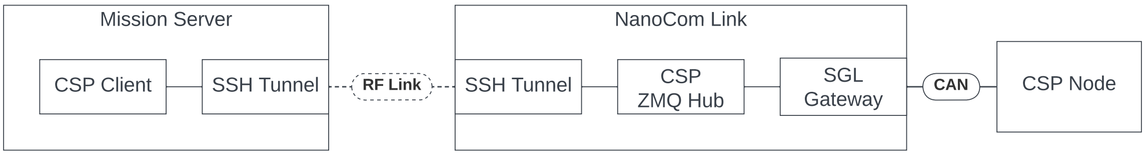


Figure 10.1: CSP over S-band example overview

A CSP client on a mission server connects to a CSP ZeroMQ (ZMQ) hub on the NanoCom Link S-band radio. The connection uses a set of SSH tunnels, which provides an encrypted connection on the RF link.

The NanoCom Link S-band radio is configured with a route for the mission server CSP client through the ZMQ hub. These steps enable CSP communication between the mission server CSP client and the SGL Gateway on the NanoCom Link S-band radio.

Furthermore, a CSP client is connected via CAN to the SGL Gateway, and a route to/from the mission server CSP client via the SGL Gateway is configured.

This provides a full end-to-end connectivity of the mission server CSP client, and a CSP node connected via CAN to the NanoCom Link S-band radio.

10.2 Configure link

The S-band link between the mission server and the NanoCom Link S-band radio must be established.

This is done by configuring the parameter `enable` in table `ccsds_base_cfg` to `True`. Monitor parameter `rx_lock` in table `ccsds_base_telem` for a value of `True` to indicate that the link is established.

On the mission server, an IPv4 tunnel using the link must be established.

10.3 SSH tunnel

Once the link with an IPv4 tunnel is established, a set of SSH tunnels is configured to provide encrypted access to the ZMQ hub on the NanoCom Link S-band radio.

The ZMQ hub uses Transmission Control Protocol (TCP) ports 6000 and 7000 for CSP communication.

The SSH tunnels are configured as follows from the mission server Linux shell:

```
$ ssh -MfN -S <tmp-socket-file-1> -l root -L 6000:localhost:6000 -o ExitOnForwardFailure=yes
-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no 10.128.0.4
$ ssh -MfN -S <tmp-socket-file-2> -l root -L 7000:localhost:7000 -o ExitOnForwardFailure=yes
-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no 10.128.0.4
```

The tunnels can be exited by running the following command from the mission server Linux shell:

```
$ ssh -S <tmp-socket-file-1> -O exit -l root 10.128.0.4
$ ssh -S <tmp-socket-file-2> -O exit -l root 10.128.0.4
```

10.4 CSP configuration

With the SSH tunnels running, the CSP client on the mission server can connect to the ZMQ hub on the NanoCom Link S-band radio, through localhost.

```
$ csp-client --csp-address=24 --csp-zmq=localhost
```

From the CSP client, routing can be configured to allow communication with the SGL Gateway on the NanoCom Link S-band radio.

```
# cmp route_set 13 1000 24 255 ZMQHUB
```

The CSP client should now be able to ping the SGL Gateway on the NanoCom Link S-band radio.

```
# ping 13
```

To reach the CSP node connected to the NanoCom Link S-band radio via CAN, a route must be configured on the node to route to ground via the SGL Gateway. The CSP node on CAN is configured with CSP address 1.

This is configured from the CSP client on the mission server.

First we configure a route from the mission server to the CAN node:

```
# cmp route_set 24 1000 1 13 ZMQHUB
```

Now we configure the return route

```
# cmp route_set 1 1000 24 13 CAN
```

Now the mission server CSP client should be able to ping the CSP node on CAN:

```
# ping 1
```

11 Storing SpaceWire Data

The following example describes how to store data from the SpaceWire data link layer directly to the NanoCom Link SDR unit. The example is depicted on Figure 11.1.

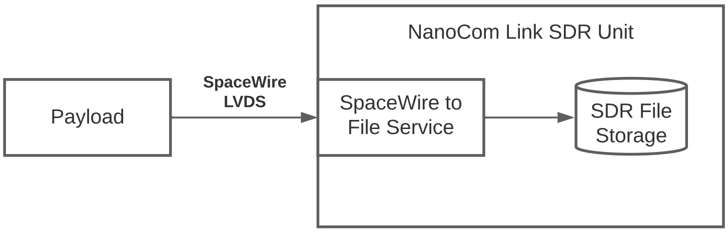


Figure 11.1: Overview of SpaceWire data storage example

The payload data in the data link layer frames is stored as files on the SDR unit filesystem with configurable name, size, and location. The frames are never split between files but the boundaries between frames are not preserved by the NanoCom Link SDR unit. That is, the payload must provide the necessary means for recovering frames boundaries if these are needed for decoding. This can be done by adding additional protocol layers on the top of the SpaceWire data link layer.

11.1 Payload Prerequisites

The payload must be compatible with the SpaceWire standard [16] for the physical and data link layers. There are no requirements to the format of the data link layer frames except the frame size which must not exceed 65 535 B. For more details regarding the SpaceWire interface protocol stack refer to [5].

11.2 SpaceWire Configuration

The NanoCom Link SpaceWire interface must be configured to operate in *file* mode. To configure the mode, access the SpaceWire parameter table and change the *mode* parameter. In the following, this is illustrated using the debug serial interface as shown in Section 2.

On the SDR unit, open the GOSH for the link control application.

```
gosh localhost 5005
```

Access the SpaceWire configuration parameter table

```
param select 3
```

Set the mode for the SpaceWire interface to *file*. In the following SpaceWire interface 0 is assumed. Use `s1_mode` or `s2_mode` instead of `s0_mode` depending on the target SpaceWire interface.

```
param set s0_mode file
```

The SpaceWire interface is now operating in file mode. See Listing 11.1 for an example of how to configure file names, sizes, and output directory. This example configures the SpaceWire data link layer frames to be stored in /data-striped/spw-data/. The files are named as spw0-data-<date>-<counter>.bin where <date> is e.g. 24-12-2022 for the 24th of December, 2022. The <counter> is a 9-digit zero-padded number running from 000000001 and upwards. This number is preserved across all configurations and multiple runs. If no configuration values are provided the values in this example are used by default.

Listing 11.1: Example of SpaceWire data file configuration

```
param set s0_file_pre spw0-data-
param set s0_file_size 5
param set s0_file_ext bin
param set s0_file_out /data-striped/spw-data/
```

When SpaceWire data is received, it is stored in a temporary file until that file reaches the configured size of e.g. 5 MB. At this point the temporary file is renamed according to the configuration and naming scheme. In case the SpaceWire interface is idle for a period of time the temporary file is renamed regardless of its size to avoid data being stuck in the temporary file. The period of time to wait while idle is also configurable. Refer to [4] for details regarding all configuration parameters.

11.2.1 Storage Full Protection

The default configuration protects the file storage from becoming full. If the usage of the file storage goes above a limit, configured by the *sX_file_du_t* parameters, data is automatically discarded.

Only files matching any of the configured data file configurations (see Listing 11.1) are considered for removal.

Whether newest or oldest data files are removed is configurable through the *sX_file_rm_n* parameters.

The default configuration is shown in Listing 11.2.

Listing 11.2: Default configuration for disk storage full protection

```
param set s0_file_du_t 0.9
param set s0_file_rm_n false
```

11.2.2 Persisting Configuration

To save the configuration to persistent storage run the following.

```
param save 3
```

This ensure the SpaceWire interface is configured correctly when the SDR unit is powered on.

11.3 SpaceWire Telemetry

The SpaceWire interfaces offer various telemetry parameters. To see these, access parameter table 9 in the link control application. In the following, this is illustrated using the debug serial interface as shown in Section 2.

On the SDR unit, open the GOSH for the link control application.

```
gosh localhost 5005
```

Access the SpaceWire telemetry parameter table

```
param select 9
```

To see the current mode for e.g. SpaceWire interface 0 run

```
param get s0_mode
```

To see if SpaceWire interface 0 has an active connection run

```
param get s0_up
```

For a complete list of all telemetry parameters in the table run

```
param list
```

Refer to [4]for details regarding all telemetry parameters.

11.4 Operation

When the SpaceWire file storage is needed, turn on the NanoCom Link SDR unit. When the SDR unit is booted it is ready to accept SpaceWire data link layer frames. Turn on the payload and keep the SDR unit running while the payload is producing SpaceWire data. Verify by observing the telemetry parameters as described in Section 11.3.

When the payload operation is finished the SDR may be turned off.

12 Recovery Boot

The NanoCom Link radio includes a secondary boot process called **recovery boot**. Recovery boot serves as a fallback mechanism in case the primary bootflow is corrupted or misconfigured and cannot load properly. This alternate bootflow loads a minimal image stored in negated or (NOR) flash memory, separate from the primary image.

The recovery boot is automatically triggered after two successive failed attempts to boot the primary payload.

12.1 Manually Triggering Recovery Boot

Two methods exist to manually trigger the recovery boot process.

U-Boot

The recovery boot process can be manually initiated by entering the U-Boot shell during startup via the universal asynchronous receiver/transmitter (UART) debug connector. To do this, press the **ESC** key when prompted during the boot sequence. This will halt the normal boot process and instead enter the U-Boot command prompt.

From here, the recovery boot process can be started by entering the following command:

```
Nanomind> run boot_recovery
```

BLCB

A recovery boot can be configured to occur on the next reboot through the NanoMind HP - Gateway Interface's bootloader control block (BLCB) command group [15]. This can be done through the GOSH interface on the device itself or via CSP commands. To do it through the former, open GOSH for the `nanomind-hp-gateway` application:

```
root@nanomind-z7045-nv3:~# gosh
Nanomind-HP #
```

Use the following commands to set the boot mode to recovery and reboot the device:

```
Nanomind-HP # nanomind_hp blcb set_boot_mode 1
Nanomind-HP # reboot 13
```

The `set_boot_mode` only affects the next boot cycle, so the device will return to normal boot mode on subsequent reboots unless the command is issued again.

Alternatively, the boot mode can be set using the `gs-blcb` command-line tool directly from the Linux shell:

```
root@nanomind-z7045-nv3:~# gs-blcb -m 1
set mode 1
root@nanomind-z7045-nv3:~# reboot
```

This is functionally equivalent to the GOSH command above.

12.2 Configuring Recovery Boot

In order to use the recovery boot for radio communication, the `link-gateway` application must be started after booting into recovery mode. The application can be started by entering the following command when booted into recovery mode:

```
root@nanomind-z7045-nv3:~# systemctl start link-gateway
```

Alternatively, the `link-gateway` application can be configured to start automatically during recovery boot by replacing the `start` command with `enable` in the above command.

The configuration parameters (see NanoSoft - Product Interface Application [17] for information on the Parameter System) are stored separately for the primary payload and recovery boot. This allows for different settings to be applied when operating in recovery mode. For example, the radio configuration used during recovery boot can be set to prioritise robustness and reliability over data throughput.

To ensure immediate radio link functionality when entering recovery mode, it is recommended to configure the radio settings for recovery boot in advance. The recovery image should be set up using the same procedure as the main payload image as described in Section 4.

To enable secure radio links in recovery mode, the recovery image must also be provisioned with the necessary cryptographic keys and settings as described in Section 6.

NOTE: As loading master keys into the system requires serial access to the device, the keys must be loaded **before launch** in order to utilise secured radio links in recovery mode.

12.3 Payload Recovery from Recovery Boot

If corruption on the payload root filesystem or the embedded multi-media controller (eMMC) is suspected, tools like `fsck` are available to check and repair the filesystem from recovery boot. To check the payload root filesystem, which is located on the second partition of the eMMC device (`/dev/mmcblk1p2`), use the following command:

```
root@nanomind-z7045-nv3-recovery:~# fsck /dev/mmcblk1p2
```

NOTE: Ensure that the payload root filesystem is unmounted before running `fsck` to prevent potential data corruption.

The payload root filesystem can be mounted and files can be manually inspected or modified. To do this, mount the payload root filesystem as shown below:

```
root@nanomind-z7045-nv3-recovery:~# mount /mnt/payload/rootfs/
```

This will mount the payload root filesystem to the `/mnt/payload/rootfs/` directory, allowing access to its contents.

13 References

[1] **Consultative Committee for Space Data Systems (CCSDS)**
Blue Book 131.0-B-3
TM Synchronization and Channel Coding
Sept. 2017, Issue 3
Available at `https://public.ccsds.org/Pubs/131x0b3e1.pdf`
Cited on pages 2, 31, 32.

[2] **European Telecommunications Standards Institute (ETSI)**
European Standard EN 302 307-1
Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 1: DVB-S2
July 2014, V1.4.1
Available at `https://tinyurl.com/y5vuwykh`
Cited on page 2.

[3] **GomSpace**
DS 1076242
NanoCom Link S, X, SX Datasheet
Cited on pages 2, 7

[4] **GomSpace**
ICD 1076238
NanoCom Link S, X, SX Control Interface
Cited on pages 2, 8–11, 13, 15, 16, 28, 29, 39, 40

[5] **GomSpace**
ICD 1076239
NanoCom Link S, X, SX Data Interface
Cited on pages 2, 13, 19, 30, 31, 38

[6] **GomSpace**
ICD 1076240
NanoCom Link S, X, SX RF Interface
Cited on pages 2, 19, 20, 31, 32, 34

[7] **minicom**
Software tool
Available at `https://github.com/Distrotech/minicom`
Cited on page 8

[8] **tio**
Software tool
Available at `https://github.com/tio/tio`
Cited on page 8

[9] **rsync(1)**
Unix manual page
Available at `https://man7.org/linux/man-pages/man1/rsync.1.html`
Cited on pages 17, 33

[10] **rsync**
Software tool
Available at `https://rsync.samba.org/`
Cited on page 17

[11] **ssh(1)**
Unix manual page
Available at `https://man7.org/linux/man-pages/man1/ssh.1.html`
Cited on page 17

[12] **OpenSSH**
Software tool
Available at `https://www.openssh.com/`
Cited on page 17

[13] **GomSpace**
TN 1069542
NanoCom Link and AX2150 Information Security
Cited on page 20

[14] **Consultative Committee for Space Data Systems (CCSDS)**
Magenta Book 354.0-M-1
Symmetric Key Management
Dec. 2023, Issue 1
Available at `https://public.ccsds.org/Pubs/354x0m1.pdf`
Cited on pages 21, 22.

[15] **GomSpace**
Interface Control Document 1046534
NanoMind HP - Gateway Interface
Cited on pages 35, 41

[16] **European Cooperation for Space Standardization (ECSS)**
Standard ECSS-E-ST-50-12C
SpaceWire - Links, nodes, routers and networks
May 2019, Revision 1
Available at `https://tinyurl.com/yx8k4pzt`
Cited on page 38.

[17] **GomSpace**
Manual 1018560
NanoSoft - Product Interface Application
Cited on page 42