numurus

SMART SYSTEM SOLUTIONS

NEPI ENGINE

LOCAL SYSTEM INTERACE API

TECHNICAL REFERENCE MANUAL

Version Number: 2.0.0
Version Date: June 30th, 2023

Table of Contents

Table of Figures

Disclaimer

Numurus LLC makes our best effort to ensure the accuracy and content in its entirety of this user manual yet there are continuous development improvements that are in progress. As such, consider the version of this document to be accurate at the time of printing. Numurus will make every effort to keep the most up-to-date version at www.numurus.com/library. Numurus is not subject to liability for errors, omissions or other differences between this document and future versions.

Introduction

This document describes the API for NEPI Edge devices. It is intended for software developers interacting with and integrating a NEPI-enabled device and those developing custom software solutions to run in a NEPI Edge environment.

NEPI API Options

NEPI-enabled edge devices provide a number of APIs for integration with custom and third-party on-device software and network attached systems. The following list provides a high-level view of the available APIs, including some notes about appropriate use. See additional sections in this document for more details.

- Websocket API for small payload command and control via web apps
    - JSON-based protocol provided via *rosbridge*: http://wiki.ros.org/rosbridge_suite
    - Freely available client libraries for many programming languages: https://github.com/RobotWebTools/rosbridge_suite#clients
    - Not appropriate for large data like imagery, pointclouds, etc.
    - Leveraged by NEPI Resident User Interface (RUI) for device configuration, command, and control
- HTTP video/snapshot server
    - Provided via *web_video_server*: http://wiki.ros.org/web_video_server
    - Makes all image topics in the ROS network available to client apps
    - Parameters to configure stream or snapshot size, quality, transport, etc.
    - Leveraged by NEPI RUI for image streams
- ROS API
    - Primary API based on ROS communication primitives
    - Most complete API in terms of access to NEPI functionality
    - Requires ROS installation on any external systems
- Sensor-specific APIs
    - Sensors and actuators attached to NEPI-enabled devices often provide additional APIs and webservers, which are made directly available to applications via NEPI network bridging
    - Examples include HTTP and RTSP for webcams, ONVIF for security-based IP cameras, etc.
- (Planned) ROS2 API
    - Currently available in limited sense via the ros1_bridge: https://github.com/ros2/ros1_bridge
- (Planned) Ultra low-latency direct image and pointcloud socket servers
    - Direct access to all sensor streams without any ROS translation overhead

NEPI Websocket API

NEPI runs a websocket server that converts JSON-formatted messages to/from ROS according to the *rosbridge* protocol. This capability is provided by the *rosbridge_server*, as detailed here: https://github.com/RobotWebTools/rosbridge_suite

This API is well-suited for external clients that are not able to install ROS directly on the external system or for software written in programming languages that do not have robust ROS client libraries (i.e., javascript). It provides a complete NEPI ROS API replacement, including all topics and services.

For small payload topics and services, the translation overhead is minimal. However, large binary data payloads (e.g., images, pointclouds) are converted to Base64-formatted character strings, a process that incurs considerable overhead on both the encoding and decoding side, leading to extreme transfer latencies for these kinds of payloads. It is not recommended to use the websocket server for these kinds of data products. Instead, a non-ROS user can use the HTTP Video and Snapshot API detailed below to obtain NEPI imagery in the browser or in a client application. At present, point cloud streaming is only supported for ROS clients.
By default, the rosbridge websocket server runs on the NEPI device at port 9090. The RUI leverages the websocket server, so adjusting this port also requires modifying and rebuilding the RUI.

The process of connecting programmatically to the rosbridge websocket server depends on the client-side programming language and client library. For example, in javascript using the *roslibjs* client library, the following would work:

```
self.ros = new ROSLIB.Ros({ url: 'ws://192.168.179.103:9090'});
```

It is not strictly necessary to use a *rosbridge* client library; it is possible to interact directly with the device as a "raw" websocket server. However, this requires considerably more client-side knowledge and coding, so it is advised to use a client library whenever practical – The *rosbridge_suite* link above is a good place to find some of the available client libraries and a complete set of documentation.

HTTP Video and Snapshot API

NEPI provides all ROS image topics over a web server as an HTTP stream as detailed here: http://wiki.ros.org/web_video_server

The weblink list of available image streams is provided at URL http://192.168.179.103:9091/

And individual image streams are available as stand-alone webpages at URLs like the following:
http://192.168.179.103:9091/stream_viewer?topic=/nepi/s2x_prototype/onwote_hd_poe/idx/color_2d_image
where additional parameters can be supplied as URL parameters/query strings as in
http://192.168.179.103:9091/stream_viewer?topic=/nepi/s2x_prototype/onwote_hd_poe/idx/color_2d_image&**quality=10**

Embedding these image streams in a browser app requires a slightly modified URL, replacing *stream_viewer* with *stream*:
http://192.168.179.103:9091/stream?topic=/nepi/s2x_prototype/onwote_hd_poe/idx/color_2d_image

Finally, a still-frame snapshot of the next image is available at URLs like
http://192.168.179.103:9091/snapshot?topic=/nepi/s2x_prototype/onwote_hd_poe/idx/color_2d_image

While these are principally useful for web viewing, they can also be leveraged programmatically, of course, by any program that can interact with a RESTful API, so they are the preferred method for non-ROS programmatic image ingestion. For example, a client application may use an HTTP-capable client library like OpenCV to receive imagery. In python this may look something like:

```
import cv2

cap =
cv2.VideoCapture("http://192.168.179.103:9091/stream_viewer?topic=/nepi/s2x_prototype/onwote_hd_poe/idx/color_2d_image")

while True:
   ret, frame = cap.read()
   cv2.imshow('video', frame)
```

NEPI ROS

NEPI leverages ROS 1 (Robot Operating System) robotics middleware for onboard software modules, configuration and data output. Although ROS is not an operating system in the traditional sense, it provides services designed for a heterogeneous computer cluster, including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. ROS communication primitives form the foundation of the NEPI ROS API.

ROS 1, generally referred to simply as ROS and distinct from ROS 2, is widely adopted and very well documented online; we do not attempt to recreate that primary documentation here. For a good introduction and jumping-off point for ROS, start here:
http://wiki.ros.org/

Host Installation

It is helpful, but not strictly required, to install ROS on a development host when testing NEPI-enabled devices and creating custom software that leverages the NEPI interface. Alternatively, it is usually

possible to use the NEPI-device itself for this kind of development and testing, as NEPI software installations include a complete ROS build environment.

ROS installation generally requires a Debian-based Linux host, though much progress has been made in recent years to allow ROS installations on other platforms (including Windows) and in Linux-based Docker containers. Presently, NEPI runs the "Melodic" version of ROS, which is tailored for Ubuntu 18.04. It is suggested that you install ROS-Melodic (Full Desktop Version) on an Ubuntu 18.04 host, though Ubuntu 16.04/ROS-Kinetic and Ubuntu 20.04/ROS-Noetic should both work without issue.

NEPI developers routinely use a VirtualBox-based Ubuntu 18.04 Virtual Machine on a Windows host for this Ubuntu/ROS installation with good results. This development path requires proper hypervisor network configuration to ensure that the VM is able to reach the NEPI device over the network.

See here for ROS installation details:
http://wiki.ros.org/melodic/Installation/Ubuntu

Host Configuration

After installing ROS on a Linux host, some additional configuration is required to interface with the NEPI system. This is generally described here:
http://wiki.ros.org/ROS/Tutorials/MultipleMachines

In subsections below we detail specific settings for interfacing with NEPI.

IP Address Settings

The host PC and NEPI system must be configured with IPv4 addresses on the same subnetwork. The default NEPI subnetwork is 192.168.179.0/24, so the easiest way to accomplish this to assign the host PC a fixed IP address on that subnet.

In some integration scenarios it may not be possible to assign a host an IP on the NEPI subnet (though it is worth reminding that most modern systems allow for multiple IP address aliases to the same network interface device, so it is often as simple as adding an IP alias). If your system cannot provide an IP address on the NEPI default subnet, it is possible to assign one or more alias addresses via ROS topics as described in the NEPI Topics List below (or via the RUI). Initially, you will need a system configured for the default subnet in order to publish on the *add_ip_addr* topic and the *save_config* topic. After that the NEPI will be reachable at the new IP address (in addition to its default address).

Environment Variables

The following environment variables must be set either manually in your shell or by adding appropriate entries to your *.bashrc* file:

ROS_MASTER_URI=http://192.168.179.103:11311
ROS_IP=<Your Local IPv4 Addr on 192.168.179.0 subnet>

Hosts File

The NEPI device must be identifiable by hostname to your system. The easiest way to accomplish that is by adding the following entries to your /etc/hosts file.

192.168.179.103 <nepi-hostname>

where <nepi_hostname> should be set to the actual hostname of your NEPI device.

Custom Message Types

Some of the ROS topics and services that constitute the NEPI ROS API include custom message and service types that must be known to your host device. These are hosted in a public Git repository:
https://bitbucket.org/numurus/nepi_ros_interfaces/src/master/

The *nepi_ros_interfaces* repository is organized for inclusion in a ROS Catkin workspace, and can be built using standard catkin or "catkin tools." Some general guidelines are provided in the following links:

http://wiki.ros.org/catkin/conceptual_overview
https://catkin-tools.readthedocs.io/en/latest/

After building the workspace, you must *source* the resulting *setup.bash* file into your environment. You can do this manually at the command-line with the *source* command, or it can be automated as part of a start-up script or in your *.bashrc* depending on the overall integration goals.

Disabling On-Board RosMaster

By default the NEPI device launches a *rosmaster* upon startup. When interfacing with a system with an existing *rosmaster*, the NEPI device can be configured not to launch its own *rosmaster* instance. This can be accomplished via the *set_rosmaster* ROS topic as described later in this document. Note that after this update, a reboot is required. At that point, the NEPI device will set its ROS_MASTER_URI environment variable as specified and will wait to launch any nodes until a connection to the master node can be established.

Warning: It is possible to effectively disable the entire NEPI ROS interface (and any other interfaces/APIs that depend on it) by misconfiguring the rosmaster URI. Care should be taken.

NEPI ROS API

Command-line and programmatic interaction with the NEPI device is provided by a collection of ROS communication primitives. Standard ROS command-line and graphical tools can be used to view data, save data, and interact with the device.

The NEPI device operates a complex ROS s/w stack with many custom and 3rd-party packages. As a result, the complete list of ROS topics it publishes and subscribes to is quite large; more than 150 at latest count. Similarly, there is a large collection of ROS services. Many of these are neither useful nor safe for end-users to manipulate, while others are redundant with alternate topics. In the tables below, we detail just those topics and services that are designated specifically as part of the current NEPI external API. For details about unlisted topics or services, consult corresponding ROS documentation and/or contact a Numurus technical representative.

ROS includes powerful command-line and graphical system introspection tools. Standard applications like *rqt* (and accompanying plugins), *rostopic*, *rosservice*, and *rosparam* are very helpful when familiarizing yourself with the NEPI ROS ecosystem. These tools can be run from the NEPI device itself (via SSH or graphical terminal) or from a development host connected to the device and configured as a ROS client of the device.

By default, the *nepi* user account on the NEPI device does not source any ROS *setup.bash* file, so command line ROS tools and applications are not immediately available in an SSH or graphical session. You can manually source a *setup.bash* file with

$ source /opt/nepi/ros/setup.bash

to prepare a NEPI device terminal to launch ROS applications.

Configuration Files and the Parameter Server

The NEPI ROS components make full use of the ROS parameter server, including loading and storing configuration files in YAML format.

In general, config. files are located in node-specific subfolders of */opt/nepi/ros/etc*

The NEPI ROS config system makes use of three files for each configurable node: A fixed factory config file (with extension *.yaml.num_factory*), a user-modifiable config. file (with extension *.yaml.user*), and a symbolic link to quickly and easily switch between the two (with extension *.yaml*). It is the symbolic link files that are loaded by the NEPI ROS launch system on start-up. In general, the contents and relationship between these files is managed under-the-hood by the NEPI config_mgr, with NEPI ROS API topics and services to change parameter values on the fly, store updates to the user config., revert on request to the default factory or current user config., etc.

On occasion, it may be necessary for a user to modify the config. files directly – in some limited instances there is no ROS NEPI API to adjust NEPI or system parameters. Generally speaking, users should not modify the *yaml.num_factory* files, instead ensuring existence of a corresponding *.yaml.user* config file, proper *.yaml* symlink to that file, and making updates there.

NEPI does not make use of ROS *dynamic_reconfigure*, instead favoring topics and services to make runtime changes to parameters and behavior. In some cases, third-party ROS nodes may include a *dynamic_reconfigure* interface – refer to specific third-party documentation for usage.

NEPI ROS Base Namespace

All NEPI ROS components (nodes, parameters, topics, services) reside under the NEPI ROS base namespace, a two-component namespace which generally varies from device to device; it includes a device identifier to facilitate integration of multiple NEPI devices into the same ROS network, where otherwise there would be undesirable identically named ROS components between the systems.

The general format of the base namespace is

/nepi/<device_id>/

where the device_id is some user-configurable ROS namespace-legal string. For example, the Numurus S2X product has a default device_id of *s2x.*

The fact that the NEPI namespace is not fixed should pose no great problem for s/w integrators; the namespace can be obtained at runtime through standard ROS introspection tools (e.g., *ros topic list | grep nepi*), and then cached in variables that are used to generate complete namespaces at runtime.

Relationship to NEPI RUI

The NEPI Resident User Interface (RUI) is the webservice that provides a browser-based user-facing graphical environment for configuration, control, and real-time data view capabilities for NEPI-enabled devices.

The NEPI RUI communicates primarily to the back-end ROS nodes by way of the *rosbridge* intermediary, which encodes and decodes ROS message and service payloads as plain JSON, allowing systems without a native ROS installation or a language-appropriate ROS client library to interact with the rest of the ROS-enabled system.

The NEPI RUI provides just a subset of full NEPI functionality exposed by the programmatic ROS interface. That is, **anything a user can accomplish via RUI can also be accomplished via the ROS interface**. But the converse is not true in general – there are some capabilities exposed only through the ROS interface with no corresponding RUI feature.

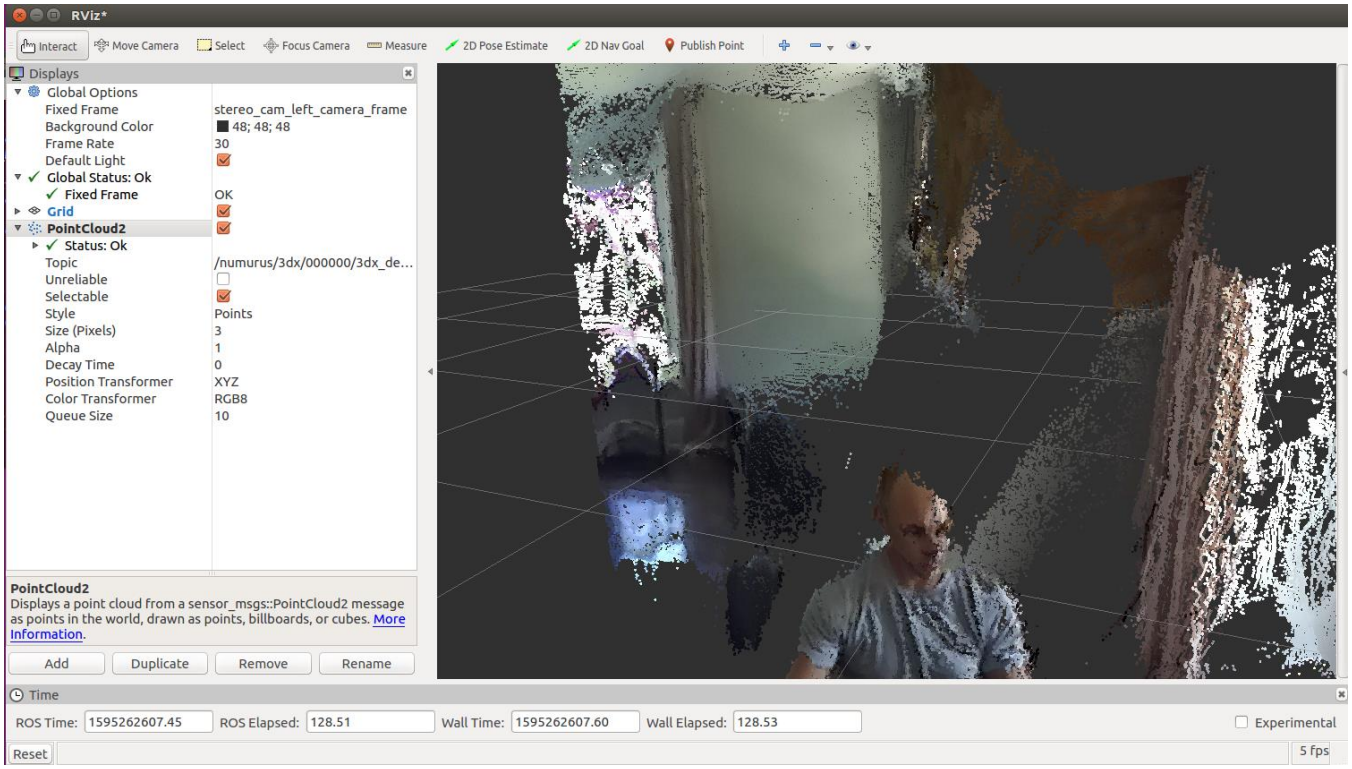
ROS Data Viewers

While the NEPI RUI provides an intuitive, web-based viewer for many of the 2D and 3D data products, not all data is viewable (currently) in the RUI. Moreover, as a browser-based application, the RUI suffers some inherent frame rate limitations and data latencies. This section details usage of various ROS GUI tools that provide an alternative to the RUI. It should be noted that these tools require a comprehensive ROS installation on the host; specifically, they are not available outside a Linux host, in contrast to the RUI, which as a browser application is multi-platform.

Viewing Point Clouds via RVIZ

At present, point cloud streaming is only available to ROS client systems, where the point cloud messages are streamed as a ROS topic stream. Point clouds can be viewed via the ROS *rviz* application. The screen

shot below shows the typical configuration, though note that your device's base namespace is likely to be different.



NEPI ROS API Tables

The following tables describe the most important ROS topics, services, and parameters that form the NEPI API. The contents are organized by namespace, and include message type(s), a short description of the purpose or contents, and the RUI equivalent section where applicable. The tables are not exhaustive; specific questions about unlisted topics should be directed to a Numurus technical representative.

The tables also include a column that specifies where in the RUI the equivalent functionality may be accessed and any details about RUI limitations compared to the direct ROS interface. Entries with nothing in this column represent functionality that is not accessible at all through the RUI.

NEPI ROS Topic Table

All topic names are with respect to the NEPI ROS Base Namespace and the node-specific namespace as identified in bold section rows.

TABLE 1 - NEPI ROS TOPICS

TOPIC	TYPE	PUBLISHERS	SUBSCRIBERS	RUI Access
		(top level)		
add_ip_addr	std_msgs/String		network_mgr	System:Admin:Network: Add
add_ntp_server	std_msgs/String		time_sync_mgr	
archive_inactive_rootfs	std_msgs/Empty		system_mgr	System:Software
clear_data_folder	std_msgs/Empty		system_mgr	Dashboard:Allow Data Deletion: Delete All Data
enable_dhcp	std_msgs/Bool		network_mgr	System:Admin:Network: DHCP Enabled
enable_gps_fix_override	std_msgs/Bool		nav_pose_mgr	Sensors:NavPose:Fixed Nav/Pose Data: Fixed Location
enable_script_autostart	nepi_ros_interfaces/AutoStartEnabled		automation_mgr	Applications:Automation: Control and Status: Auto Start
install_new_image	std_msgs/String		system_mgr	System:Software:Full System Update: Install
remove_ip_addr	std_msgs/String		network_mgr	System:Admin:Network: Delete
remove_ntp_server	std_msgs/String		time_sync_mgr	
reset	nepi_ros_interfaces/Reset		<configurable_node_name>	System:Admin:Configuration:Subsystem:All:UserReset
save_config	std_msgs/Empty		<configurable_node_name>	System:Admin:Configuration:Subsystem:All:Save
save_data	nepi_ros_interfaces/SaveData	system_mgr, nepi_link_ros_bridge	<savable_data_node_name>	Dashboard:Save Data
save_data_prefix	std_msgs/String		<savable_data_node_name>	Dashboard: File Name Prefix
save_data_rate	nepi_ros_interfaces/SaveDataRate	nepi_link_ros_bridge	<savable_data_node_name>	Dashboard: Save Freq. (Hz)
set_device_id	std_msgs/String		system_mgr	System:Admin:Advanced Settings Enable:Device ID
set_gps_fix	sensor_msgs/NavSatFix	gpsd_ros_client	nav_pose_mgr	
set_gps_fix_override	sensor_msgs/NavSatFix		nav_pose_mgr	Sensors:NavPose:Fixed Nav/Pose Data: Fixed Location: Lat/Lon/Altitude
set_op_environment	std_msgs/String	system_mgr	system_mgr	
set_periodic_sw_trig	nepi_ros_interfaces/PeriodicSwTrig		trigger_mgr	System:Admin:Trigger Settings:Auto Rate (HZ)
set_rosmaster	std_msgs/String		network_mgr	
set_time	std_msgs/Time		time_sync_mgr	Dashboard:System Clock: Sync Clocks
set_tx_bw_limit_mbps	std_msgs/Int32		network_mgr	System:Admin:Network:TX Rate Limit (Mbps)

start_classifier	nepi_ros_interfaces/ClassifierSelection		nepi_darknet_ros_mgr	Applications:AI:A/I Status:Start
stop_classifier	std_msgs/Empty		nepi_darknet_ros_mgr	Applications:AI:A/I Status:Stop
submit_system_error_msg	std_msgs/String	trigger_mgr, nav_pose_mgr	system_mgr	
sw_trigger	std_msgs/UInt32	trigger_mgr	trigger_mgr	
switch_active_inactive_rootfs	std_msgs/Empty		system_mgr	System:Software:A/B Partition Settings:Switch Active/Inactive
sys_time_updated	std_msgs/Empty	time_sync_mgr		
system_status	nepi_ros_interfaces/SystemStatus	system_mgr		Various Read/Only Fields throughout RUI
trigger_index_settings	nepi_ros_interfaces/TriggerIndexSettings		trigger_mgr	
/classifier/				
bounding_boxes	darknet_ros_msgs/BoundingBoxes	nepi_darknet_ros	nepi_darknet_ros_mgr	
detection_image	sensor_msgs/Image	nepi_darknet_ros	web_video_server	Applications:AI:Image Pane (classifier running)
found_object	darknet_ros_msgs/ObjectCount	nepi_darknet_ros		
/<savable_data_node_name>/				
save_data	nepi_ros_interfaces/SaveData		<savable_data_node_name>	Only global, not per node
save_data_prefix	std_msgs/String		<savable_data_node_name>	Only global, not per node
save_data_rate	nepi_ros_interfaces/SaveDataRate		<savable_data_node_name>	Only global, not per node
save_data_status	nepi_ros_interfaces/SaveDataStatus	<savable_data_node_name>		Only global, not per node
/<configurable_node_name>/				
reset	nepi_ros_interfaces/Reset		<configurable_node_name>	System:Admin:Configuration:Subsystem:<Node>:User Reset
save_config	std_msgs/Empty		<configurable_node_name>	System:Admin:Configuration:Subsystem:<Node>:Save
/<idx_sensor_node_name>/idx/				
bw_2d_image	sensor_msgs/Image	<idx_sensor_node_name>		Sensors:Imagery:Selected Image: B&W 2D
color_2d_image	sensor_msgs/Image	<idx_sensor_node_name>	sequential_image_mux	Sensors:Imagery:Selected Image:Color 2D
set_brightness	std_msgs/Float32		<idx_sensor_node_name>	Sensors:Imagery:Brightness

set_contrast	std_msgs/Float32		<idx_sensor_node_name>	Sensors:Imagery:Contrast
set_framerate_mode	std_msgs/UInt8		<idx_sensor_node_name>	Sensors:Imagery:Framerate
set_resolution_mode	std_msgs/UInt8		<idx_sensor_node_name>	Sensors:Imagery:Resolution
set_range	nepi_ros_interfaces/RangeWindow		<idx_sensor_node_name>	Sensors:Imagery:Range
set_thresholding	std_msgs/Float32		<idx_sensor_node_name>	Sensors:Imagery:Thresholding
status	nepi_ros_interfaces/IDXStatus	<idx_sensor_node_name>		Various fields throughout Sensors:Imagery page
/nav_pose_mgr/				
enable_attitude_override	std_msgs/Bool		nav_pose_mgr	Sensors:NavPose:Fixed Orientation
enable_heading_override	std_msgs/Bool		nav_pose_mgr	Sensors:NavPose:Fixed Heading
set_ahrs_offset	nepi_ros_interfaces/Offset		nav_pose_mgr	Sensors:NavPose:Enable Manual Nav/Pose Offsets:Text Fields
set_ahrs_out_frame	std_msgs/String		nav_pose_mgr	
set_ahrs_src_frame	std_msgs/String		nav_pose_mgr	
set_attitude_override	geometry_msgs/QuaternionStamped	gpsd_ros_client	nav_pose_mgr	Sensors:NavPose:Fixed Orientation: Text Fields
set_heading_override	nepi_ros_interfaces/Heading	gpsd_ros_client	nav_pose_mgr	Sensors:NavPose:Fixed Heading: Heading
set_imu_topic	std_msgs/String		nav_pose_mgr	
set_odom_topic	std_msgs/String		nav_pose_mgr	
/nepi_link_ros_bridge/				
connect_now	std_msgs/Empty		nepi_link_ros_bridge	Connect:Comms Link Settings: Connect Now
enable	std_msgs/Bool		nepi_link_ros_bridge	Connect:Enable NEPI Connect
enable_nepi_log_storage	std_msgs/Bool		nepi_link_ros_bridge	
hb/enable	std_msgs/Bool		nepi_link_ros_bridge	Connect:Enable High Bandwidth Comms
hb/set_auto_data_offloading	std_msgs/Bool		nepi_link_ros_bridge	
lb/create_data_set_now	std_msgs/Empty		nepi_link_ros_bridge	Connect:Low Bandwidth Data Config:Capture Data Now
lb/enable	std_msgs/Bool		nepi_link_ros_bridge	Connect:Comms Link Settings: Enable Low Bandwidth Comms
lb/select_data_sources	nepi_ros_interfaces/StringArray		nepi_link_ros_bridge	Connect: Low Bandwidth Data Config: Data Topics

lb/set_data_sets_per_hour	std_msgs/Float32		nepi_link_ros_bridge	Connect: Low Bandwidth Data Config: Data Rate per Hour
set_auto_attempts_per_hour	std_msgs/Float32		nepi_link_ros_bridge	Connect:Comms Link Settings:Connection Attempts per Hour
/rui_config_mgr/				
settings	nepi_ros_interfaces/RUISettings			RUI Internal Use
/sequential_image_mux/				
configure_mux_sequence	nepi_ros_interfaces/ImageMuxSequence		sequential_image_mux	Sensors:Sequencer:Sequences:Apply Changes
delete_mux_sequence	std_msgs/String		sequential_image_mux	Sensors:Sequencer:Sequences:Delete

NEPI ROS Service Table

All service names are with respect to the NEPI ROS Base Namespace and the node-specific namespace as identified in bold section rows.

TABLE 2 - NEPI ROS SERVICES

SERVICE	TYPE	RUI Access
/automation_mgr/		
get_running_scripts	nepi_ros_interfaces/GetRunningScriptsQuery	Applications:Automation:Running Scripts
get_script_status	nepi_ros_interfaces/GetScriptStatusQuery	
get_scripts	nepi_ros_interfaces/GetScriptsQuery	Applications:Automation:Automation Scripts
get_system_stats	nepi_ros_interfaces/GetSystemStatsQuery	Applications:Automation:Control and Status (Table values)
launch_script	nepi_ros_interfaces/LaunchScript	Applications:Automation:Control and Status:Start
stop_script	nepi_ros_interfaces/StopScript	Applications:Automation:Control and Status:Stop
/config_mgr/		
factory_reset	nepi_ros_interfaces/FileReset	System:Admin:Configuration:Factory Reset (Requires Advanced Settings Enable)
user_reset	nepi_ros_interfaces/FileReset	System:Admin:Configuration:User Reset
/<idx_sensor_node_name>/		
idx/capabilities_query	nepi_ros_interfaces/IDXCapabilitiesQuery	Sensors:Imagery:Control Panel and Selected Image list
/<savable_data_node_name>/		
<savable_data_node_name>/query_data_products	nepi_ros_interfaces/DataProductQuery	Dashboard:Save Data:Save Data Checkbox (limited to ALL data producers, no individual control)
/nav_pose_mgr/		
nav_pose_query	nepi_ros_interfaces/NavPoseQuery	Sensors:NavPose: Various Fields
nav_pose_status_query	nepi_ros_interfaces/NavPoseStatusQuery	Sensors:NavPose: Various Fields
/nepi_darknet_ros_mgr/		
img_classifier_list_query	nepi_ros_interfaces/ImageClassifierListQuery	Applications:AI:Settings:Image Classifier drop-down
img_classifier_status_query	nepi_ros_interfaces/ImageClassifierStatusQuery	Applications:A/I Status
/nepi_link_ros_bridge/		
nepi_link_status_query	nepi_ros_interfaces/NEPILinkStatusQuery	Connect:Various Fields
/network_mgr/		
bandwidth_usage_query	nepi_ros_interfaces/BandwidthUsageQuery	System:Admin:Network:TX Data Rate and RX Data Rate
ip_addr_query	nepi_ros_interfaces/IPAddrQuery	System:Admin:Network:Device IP Addresses
wifi_query	nepi_ros_interfaces/WifiQuery	
/sequential_image_mux/		

mux_sequence_query	nepi_ros_interfaces/ImageMuxSequenceQuery	Sensors:Sequencer: Various Fields
/system_mgr/		
op_environment_query	nepi_ros_interfaces/OpEnvironmentQuery	
sw_update_status_query	nepi_ros_interfaces/SystemSoftwareStatusQuery	System:Software:Full System Update:Status and Progress
system_defs_query	nepi_ros_interfaces/SystemDefsQuery	Dashboard: Device Info and System Status Various Fields
system_storage_folder_query	nepi_ros_interfaces/SystemStorageFolderQuery	
/time_sync_mgr/		
time_status_query	nepi_ros_interfaces/TimeStatusQuery	Dashboard: System Clock: Various Fields
/trigger_mgr/		
trigger_defs	nepi_ros_interfaces/TriggerDefs	
trigger_status_query	nepi_ros_interfaces/TriggerStatusQuery	

NEPI ROS Topic Descriptions

Following subsections describe each NEPI ROS Topic. Refer to tables in the previous section for payload details, RUI equivalent, etc.

Topic: add_ip_addr

Add an IP address alias to the main ethernet interface. Payload should be CIDR formatted (e.g., 192.168.100.103/24)

Topic: add_ntp_server

Add a new network time protocol (NTP) server. Payload is an IPv4 address with no netmask indicator or a resolvable hostname (if NEPI device is attached to a proper DNS server)

Topic: archive_inactive_rootfs

Create a complete binary image of the "inactive" rootfs partition copied to the user partition. Can be used in conjunction with *switch_active_inactive_rootfs* topic and system reboot to switch current active to inactive in order to archive the image.

Topic: clear_data_folder

Delete all contents of the Data folder on the user partition.

Topic: enable_dhcp

Enable NEPI device to act as DHCP client to set additional IP address alias for the primary Ethernet interface.

Topic: enable_gps_fix_override

Enable/disables NEPI GPS fix override. When enabled, NEPI uses and reports the override fix rather than any external GPS stream. See *set_gps_fix_override*.

Topic: enable_script_autostart

Enable/disable auto-start on boot for the specified automation script.

Topic: install_new_image

Install the currently detected NEPI complete ROOTFS image to the current "inactive" partition

Topic: remove_ip_addr

Remove a previously added IP address. Payload is a CIDR-formatted IPv4 address. Note: It is not possible to remove the default device IP address in this way, only user-configured or DHCP-assigned alias addresses.

Topic: remove_ntp_server
Remove a configured NTP server from the list of available servers.

Topic: reset
Execute a configuration reset for a particular node or all nodes simultaneously. The payload indicates what type of reset to perform. A “User” reset will revert any current changes that have not yet been stored to the device. A “Factory” reset will revert configuration back to the factory default settings. In the case of “Factory” reset, previous user settings are retained for archival purposes, but restoring them requires manual filesystem intervention. “Hardware” and “Software” reset effect depends on the particular node.

Topic: save_config
Store the current configuration parameters for a particular node or all nodes simultaneously (depending on topic namespace level). After configuration is stored, a subsequent system reboot will restore these parameters. See *reset* for details about reverting user-stored settings to factory defaults.

Topic: save_data
Start saving all configured data products for a particular node or all nodes simultaneously (depending on topic namespace level). Data products are saved at a pre-configured rate and with pre-configured filename prefix.

Topic: save_data_prefix
Set a prefix and/or subdirectory path for saved data filenames. If the prefix includes one or more forward-slash characters after strings the prefix will be treated as a subdirectory hierarchy (below the user partition Data folder).

Topic: save_data_rate
Set the rate at which a particular data product or all data products (depending on payload) for a particular node or all nodes (depending on topic namespace level) is saved to disk. This rate serves as a soft max – data products that are produced at a lower rate than this max will save at their full rate.

Topic: set_device_id
Set a new device ID, which is used in the NEPI ROS base namespace, /nepi/<device_id>. Requires a reboot to take effect.

Topic: set_gps_fix
ROS interface for GPS lat/lon/altitude solution. Published by the NEPI GPSD ROS client that interfaces GPSD server to ROS, but can also be published externally to update the Nav/Pose solution

Topic: set_gps_fix_override

Set the override (typically static/fixed) GPS solution. If GPS fix override is enabled (see *enable_gps_fix_override*), this value will be used instead of any value streamed via the *set_gps_fix* live topic.

Topic: set_op_environment

Set the system "operating environment," which controls various aspects of the system. Valid operating environment variable vary system by system. For example, a submersible system with cameras may include "water" and "air" operating environments to adjust camera calibrations depending on whether the systems are submerged or not.

Topic: set_periodic_sw_trig

Some NEPI sensors and actions are s/w-triggerable and have a mask value associated with them so that a single numeric trigger message can control multiple trigger events simultaneously and selectively. This topic sets NEPI to provide a particular s/w trigger value at a specified rate. Additional triggers with different trigger values can be set to run complex trigger scenarios.

Topic: set_rosmaster

Set the NEPI device ROS_MASTER_URI environment variable for networking scenarios with an external rosmaster in the network. Requires reboot after setting. **Warning: This topic can render ROS API unusable if the specified rosmaster does not exist or is not network reachable**. NEPI device attempts to ping the new rosmaster before accepting this setting to ensure that it exists and is reachable, but there is no fallback mechanism if that condition is not met on subsequent power cycles.

Topic: set_time

Sets the NEPI device system clock to the specified value for scenarios where NTP is not available.

Topic set_tx_bw_limit_mbps

Sets an upper limit on the network transmit throughput of the NEPI device's main interface. Useful in bandwidth-constrained networks to avoid swamping the network with large data traffic from NEPI.

Topic: start_classifier

Start an AI classifier inference model with specified input image stream, confidence threshold, and network model. If a classifier is already running, this topic will force it to stop and restart with the specified payload parameters.

Topic: stop_classifier

Stop a currently-running classifier

Topic: submit_system_error_msg
Typically used only internally by NEPI, this topic is also available to be published on by custom nodes. Primary use is to push a message into the system_msgs list that is displayed in RUI, etc.

Topic: sw_trigger
Send a s/w trigger with numeric value payload. The payload is a bitmask that indicates which triggerable nodes (sensors, events, etc.) should be activated.

Topic: switch_active_inactive_rootfs
Set NEPI device to boot to the currently inactive ROOTFS **on next boot**. Useful for forcing a s/w version fallback or in preparation for archiving or overwriting the current active ROOTFS (which must be made inactive before either of these operations can be performed).

Topic: sys_time_updated
This topic is published whenever the NEPI device system clock is updated via initial NTP synchronization or the *set_time* topic. Primarily used by NEPI internally, but can be helpful for guarding against unexpected timestamp jumps or failed s/w timeouts, etc. in custom nodes.

Topic: system_status
Top-level system status published by the NEPI system manager. Includes current time, temperature(s), user partition disk status, etc.

Topic: trigger_index_settings
NEPI triggerable nodes publish to this topic to inform the NEPI Trigger Manager (and any other nodes that care) their individual trigger index/mask value. Typically only used internally.

Topic: classifier/bounding_boxes
Output data topic from classifier that provides a small payload readable-text list of current detections, including identifying details about the source image and the image boundaries of a bounding box that contains the detected object.

Topic: classifier/detection_image
Output data topic from classifier that is a copy of the classifier input image with bounding box and object labels overlaid.

Topic: classifier/found_object
Output data topic from classifier that provides the count of detected objects in the most recent input image

Topic: idx/bw_2d_image
Grayscale 2D image produced by a NEPI IDX sensor. Only advertised if available for that particular sensor.

Topic: idx/color_2d_image
Full color 2D image produced by a NEPI IDX sensor. Only advertised if available for that particular sensor.

Topic: idx/set_brightness
Topic to adjust sensor “brightness.” Actual effect depends on sensor including any config. file-based parameter remapping. Only advertised if available for that particular sensor.

Topic: idx/set_contrast
Topic to adjust sensor “contrast.” Actual effect depends on sensor including any config. file-based parameter remapping. Only advertised if available for that particular sensor.

Topic: idx/set_framerate_mode
Topic to adjust sensor “framerate” between 4 distinct values. Actual effect depends on sensor including any config. file-based parameter remapping. Only advertised if available for that particular sensor.

Topic: idx/set_range
Topic to adjust sensor “start range” and “stop range.” Actual effect depends on sensor including any config. file-based parameter remapping. Only advertised if available for that particular sensor.

Topic: idx/set_resolution_mode
Topic to adjust sensor “resolution” between 4 distinct values. Actual effect depends on sensor including any config. file-based parameter remapping. Only advertised if available for that particular sensor.

Topic: idx/set_thresholding
Topic to adjust sensor “thresholding” value. Actual effect depends on sensor including any config. file-based parameter remapping. Only advertised if available for that particular sensor.

Topic: idx/status
Published by NEPI IDX sensors to inform of their current IDX parameter values.

Topic: nav_pose_mgr/enable_attitude_override
Enables/disables the attitude override value to take precedence over any live attitude received by the device

Topic: nav_pose_mgr/enable_heading_override
Enables/disables the heading override value to take precedence over any live heading received by the device

Topic: nav_pose_mgr/set_ahrs_offset
Sets the attitude/heading reference 6-DOF offset value. This represents the transform from the AHRS sensor to the NEPI center frame of reference.

Topic: nav_pose_mgr/set_ahrs_out_frame
Sets the desired output frame of reference (by name) for AHRS data. Frame must be defined in the ROS *tf* subsystem.

Topic: nav_pose_mgr/set_ahrs_src_frame
Sets the input frame of reference (by name) for AHRS data, typically the frame for the AHRS sensor, itself. Frame must be defined in the ROS *tf* subsystem.

Topic: nav_pose_mgr/set_attitude_override
Sets the attitude override values, which take precedence over any dynamic streaming attitude input whenever *enable_attitude_override* is set true.

Topic: nav_pose_mgr/set_heading_override
Sets the heading override value, which takes precedence over any dynamic streaming heading input whenever *enable_heading_override* is set true.

Topic: nav_pose_mgr/set_imu_topic
Specifies the ROS topic (full namespace) for IMU data (as input to nav_pose_mgr). This only has an effect when the AHRS type is set to ROS.

Topic: nav_pose_mgr/set_odom_topic
Specifies the ROS topic (full namespace) for Odometry data (as input to nav_pose_mgr). This only has an effect when the AHRS type is set to ROS.

Topic: nepi_link_ros_bridge/connect_now
Starts a synchronous NEPI Connect session to the configured cloud endpoint. This connection attempt will occur right away as long as another Connect session is not already in process. It has no impact on any configured periodic sessions (as configured by *set_auto_attempts_per_hour*).

Topic: nepi_link_ros_bridge/enable
Enable or disable all NEPI Connect features. When disabled, no scheduled LB data sets will be collected, no scheduled Connect sessions will be started, and no synchronous attempts at either will be honored.

Topic: nepi_link_ros_bridge/enable_nepi_log_storage
When enabled, NEPI Connect session logs are stored to the Logs folder in the user partition and can be uploaded to Cloud endpoint as part of an HB data upload session.

Topic: nepi_link_ros_bridge/hb/enable
Enable/disable HB session during NEPI Connect sessions. Only when enabled will HB features execute, including *auto_data_offloading*.

Topic: nepi_link_ros_bridge/hb/set_auto_data_offloading
Enable/disable automatic Data folder offloading during NEPI Connect sessions. Be aware that when enabled, all Data folder contents will be uploaded to cloud (via *rsync*, so redundancy is avoided) on each Connect session, which can require significant network bandwidth and cloud-side storage.

Topic: nepi_link_ros_bridge/lb/create_data_set_now
This topic will synchronously launch LB data collection as configured independent of any scheduled data set construction (via *set_data_sets_per_hour*).

Topic: nepi_link_ros_bridge/lb/enable
Enable/disable LB data collection and LB session during NEPI Connect sessions. Only when enabled will LB features execute.

Topic: nepi_link_ros_bridge/lb/select_data_sources
Set the ROS topics that are subscribed to during LB data collection, and any built-in or custom scripts that convert those topics to appropriate LB data files. This sets up the data snippets for LB sessions.

Topic: nepi_link_ros_bridge/lb/set_data_sets_per_hour
Configure the schedule that controls when automatic LB data sets are collected (according to the *select_data_sources* configuration).

Topic: nepi_link_ros_bridge/set_auto_attempts_per_hour
Configure the schedule that controls automatic NEPI Connect session attempts.

Topic: sequential_image_mux/configure_mux_sequence
Create or update a complete image mux sequence to generate a round-robin type image sequence topic for downstream consumers

Topic: sequential_image_mux/delete_mux_sequence
Delete an existing image mux sequence from the list of available mux sequences. Note, this is distinct and more extreme than disabling an existing image mux sequence (as is possible with *configure_mux_sequence*).

NEPI ROS Service Descriptions

Following subsections describe each NEPI ROS Service. Refer to tables in the previous section for payload details, RUI equivalent, etc.

Service: automation_mgr/get_running_scripts

Query for the list of automation scripts currently running.

Service: automation_mgr/get_scripts

Query for the list of all automation scripts currently known to NEPI Automation Manager

Service: automation_mgr/get_system_stats

Query for the running statistics for a single script (identified by name)

Service: automation_mgr/launch_script

Synchronously start an automation script (identified by name). Note: this call attempts to start an additional instance of the script even if there is already an instance running.

Service: automation_mgr/stop_script

Force a currently running automation script to stop via a kill signal

Service: config_mgr/factory_reset

NEPI Internal use: Not part of standard public API. See the *reset* NEPI ROS topic, instead.

Service: config_mgr/user_reset

NEPI Internal use: Not part of standard public API. See the *reset* NEPI ROS topic, instead.

Service: <idx_sensor>/idx/capabilities_query

Query for the list of IDX supported features for a particular NEPI IDX sensor

Service: <data_producer>/query_data_products

Query for the list of data products that are produced and can be saved to Data folder by a given data producer NEPI node.

Service: nav_pose_mgr/nav_pose_query

Query for the nav/pose solution with a given timestamp. Timestamp must be within last 5 seconds and will be interpolated if it does not correspond to an exact timestamp at which nav pose was computed. Use timestamp 0.0 to request the latest solution.

Service: nav_pose_mgr/nav_pose_status_query
Query for the current status of the nav/pose system to determine if AHRS and/or GPS input data is flowing.

Service: nepi_darknet_ros_mgr/img_classifier_list_query
Query for the list of Darknet image classifiers currently stored on the NEPI device

Service: nepi_darknet_ros_mgr/img_classifier_status_query
Query for the current status of the Darknet classifier; specifically, if it is running, the input image stream, and the current detection threshold value.

Service: nepi_link_ros_bridge/nepi_link_status_query
Query for the current status of the NEPI Connect subsystem, including whether enabled, currently running, and any scheduling parameters.

Service: network_mgr/bandwidth_usage_query
Query for current network bandwidth consumption and limits on the main NEPI device ethernet port

Service: network_mgr/wifi_query
Query for the current Wifi availability and status for the NEPI device

Service: sequential_image_mux/mux_sequence_query
Query for the list and status of all configured image mux sequences.

Service: system_mgr/op_environment_query
Query for the currently configured operating environment (e.g., air, water)

Service: system_mgr/sw_update_status_query
Query for the detection of a s/w update (in nepi_full_sys_img) folder and the current update execution status/progress.

Service: system_mgr/system_defs_query
Query for a litany of NEPI device system definitions including firmware revision, device ID and serial number, disk capacity and usage, etc.

Service: system_mgr/system_storage_folder_query
Query for the full path to the NEPI storage folder (i.e., user partition). Mostly just for NEPI internal use, but could be helpful for custom nodes that must gather files from or store data to user partition.

Service: time_sync_mgr/time_status_query
Query for the current NEPI device system time and NTP status

Service: trigger_mgr/trigger_defs
Query for the list of triggerable nodes indexed by their individual trigger bitmask offset values

Service: trigger_mgr/trigger_status_query
Query for basic status about the trigger manager; current auto trigger setup (desired and achieved rate), most recent trigger time, etc.

Sensor-specific APIs
Individual sensors, actuators, etc. may provide socket-based interfaces with a more complete set of controls, data streams, etc. than are provided directly by the NEPI APIs. NEPI makes these servers directly available to clients by bridging between disparate subnetworks. For example, a NEPI-connencted ONVIF-enabled security camera with IP address 192.168.0.41 running an RTSP server at URL
rtsp://192.168.0.41:554/stream1
can be reached by a client on the NEPI main subnetwork 192.168.179.0/24. Generally this requires setting a default route on the client system that uses the NEPI device as a gateway to the sensor network.

For example, on many modern Linux distributions this can be achieved with the shell command of the form

```
sudo ip route add 192.168.0.0/24 192.168.179.103 dev enp0s1
```

Where the specific IP subnetwork and addresses and specific Ethernet device identifier should be set according to the user's system.

Similarly, on Windows a route can be established via a shell command of the form

```
route add 192.168.0.0 MASK 255.255.255.0 192.168.179.103
```

For details about specific sensor APIs, refer to sensor manufacturer documentation.

ROS2 API

Coming Soon!

Interim ROS2 API available via *ros1_bridge*:
https://github.com/ros2/ros1_bridge

Direct Image and Pointcloud Socket APIs

Coming Soon!