GARMIN®

GETTING STARTED WITH THE GARMIN® SENSOR GATEWAY API

Table of Contents

Network and Service Information

To allow for easy hostname resolution, this device advertises its mDNS services, which you can use to access the device documentation, configuration, and API.

The device is also available for access at the static IPv4 address **172.20.1.1**.

Service Name	Protocol
_http._tcp	This service name represents the standard HTTP interface used to access the device documentation, configuration, and API.

HTTP API Information

The Garmin Sensor Interface (GSI) device hosts an HTTP-based API that you can use to interact with Garmin radar devices. This API uses several industry standards to facilitate communication between the Garmin device and connected client devices.

To interact with the API, clients communicate using standard HTTP over TCP. All communication uses the JSON interchange format to allow for a human-readable data exchange between the API and client devices.

The API is documented using the OpenAPI specification to ensure that the API is defined in an industry-accepted format and to provide additional tooling and support from the OpenAPI ecosystem.

The API uses HTTP resource identifiers to interact with the system in a logical hierarchy. All API resources are relative to the host name combined with the base API services path. This example shows a resource identifier used to interact with a "radar transmit" configuration.

garmin-gsi.local/garmin/gsi/v1/Radar/{RadarId}/Configuration/Transmit
To explain the previous example, this table outlines the three main path components required in each HTTP interaction with the API.

Host Name	Base API Service	API Resource
garmin-gsi.local/	garmin/gsi/v1/	Radar/{RadarId}/Configuration/Transmit

Accessing Additional Specification Documentation

This document provides only a high-level overview and basic use information for the Garmin Sensor Gateway API. Additional HTML-based specification documentation is hosted directly from the device.

1. If necessary, connect your computer to the GSI 10 device according to the installation instructions.
2. Configure your network connection to the 172.20.x.x subnet and access the device using either the advertised mDNS address or the static IPv4 address (172.20.1.1).
3. In a web browser, go to **garmin-gsi.local/garmin/gsi/Docs**.

API Interaction

Because the Garmin Sensor Gateway API uses the HTTP protocol, no additional software is required to interact or control the device. You can use common tools such as a web browser or cURL (curl.se) to initially test the API.

For example, when properly connected to the device with a computer, if you use a web browser to go to **garmin-gsi.local/garmin/gsi/v1/Radar**, the browser returns a list of the {RadarId} values for the radar scanners connected to the device, such as **["Fantom 256","GMR24XHD"]**.

This table shows the same request using cURL.

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar**
Output	["Fantom 256","GMR24XHD"]

NOTE: For consistency, examples throughout this document will be shown using cURL in tables similar to this.

System Resources

System resources allow you to interact at a network level using the API. You can use these resources to gather information about devices connected to the system. For example, the base "/Radar" resource is a system-level resource that returns a list of all of the radar scanners connected to the system.

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar**
Output	["Fantom 256","GMR24XHD"]

You should refer to the full API specification hosted on the device for the complete list of system-level resources (*Accessing Additional Specification Documentation*, page 1).

Configuration Resources

The configuration resources allow you to read, write, and modify the radar systems configuration using the API. Using the configuration resources, you can manage top-level resources and access granular, individual configuration sub-resources. These resources are parameterized so you can access various parts of the radar system.

To obtain or modify a configuration for a connected radar, you must provide a Radar ID. You can obtain a Radar ID for a connected radar from the "/Radar" system resource (*System Resources*, page 2). When referencing the examples in this document and in the full API specification hosted on the unit, you should substitute the appropriate Radar ID in place of the {RadarId} path parameter shown in the example or specification. For example, the configuration resource "/Radar/Fantom18/Configuration" returns the entire configuration for a connected Fantom™ 18 radar scanner with a Radar ID of "Fantom18."

Within the configuration, sub-resources are accessed directly. For example, if you want to adjust the parked position of the antenna for an open-array Fantom radar, you can post the change directly to the configuration resource "/Radar/{RadarId}/Configuration/Installation/ParkAdjust." For this example, the {RadarId} is "Fantom %20256."

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration**

Output	{
	"Installation": {
	"FrontOfBoatAdjust": 22.0,
	"ParkAdjust": 0.0,
	"AntennaSize": "6Foot"
	},
	"RangeChannels": [
	{
	"AutoGainMode": "High",
	"Gain": 74.0,
	"Range": 22224.0,
	"TargetSizeMode": "Normal",
	"Filters": {
	"SeaClutter": {
	"Gain": 50.0,
	"Mode": "Off"
	},
	"RainClutter": {
	"Enabled": false,
	"Gain": 50.0
	}
	}
	},
	{
	"AutoGainMode": "High",
	"Gain": 76.0,
	"Range": 926.0,
	"TargetSizeMode": "Normal",
	"Filters": {
	"SeaClutter": {
	"Gain": 50.0,
	"Mode": "Off"
	},
	"RainClutter": {
	"Enabled": false,
	"Gain": 50.0
	}

<table>
<tr><td></td><td><pre>
 }
 }
],
"CrossTalkRejection": {
 "DitherEnabled": true,
 "NoiseBlankingEnabled": true,
 "RejectionLevel": "Low",
 "TransmitChannel": {
  "Mode": "Manual",
  "Channel": 1
 }
},
"RangeMode": "Dual",
"RotationSpeedMode": "Low",
"Transmit": false
}
</pre></td></tr>
<tr><td>Get Command</td><td>curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/Installation/ParkAdjust</td></tr>
<tr><td>Output</td><td>0.0</td></tr>
<tr><td>Post Command</td><td>curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/Installation/ParkAdjust -d 90</td></tr>
<tr><td>Output</td><td>90.0</td></tr>
<tr><td>Get Command</td><td>curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/Installation/ParkAdjust</td></tr>
<tr><td>Output</td><td>90.0</td></tr>
</table>

You should refer to the full API specification hosted on the device for the complete list of configuration resources (*Accessing Additional Specification Documentation*, page 1).

Working with Range Channels

Some models of Garmin radar scanners support the ability to scan at more than one range using the same physical radar. These virtual range channels allow the radar to scan both short and long ranges simultaneously and produce two independent streams of data.

You can view the configuration details for each range using the **curl garmin-gsi.local/garmin/gsi/v1/Radar/{RadarId}/Configuration** command. For this example, the {RadarId} is "Fantom%20256."

<table>
<tr><td><pre>{
 "Installation": {
  "FrontOfBoatAdjust": 22.0,
  "ParkAdjust": 0.0,
  "AntennaSize": "6Foot"
 },
 "RangeChannels": [</pre></td><td>General radar information</td></tr>
<tr><td><pre>{
  {
   "AutoGainMode": "High",
   "Gain": 74.0,
   "Range": 22224.0,
   "TargetSizeMode": "Normal",
   "Filters": {
    "SeaClutter": {
    "Gain": 50.0,
    "Mode": "Off"
    },
    "RainClutter": {
     "Enabled": false,
     "Gain": 50.0
    }
   }
  },</pre></td><td>First range channel configuration information</td></tr>
<tr><td><pre>{
   "AutoGainMode": "High",
   "Gain": 76.0,
   "Range": 926.0,
   "TargetSizeMode": "Normal",
   "Filters": {
    "SeaClutter": {
     "Gain": 50.0,
     "Mode": "Off"
    },
    "RainClutter": {
     "Enabled": false,
     "Gain": 50.0
    }
   }
  }</pre></td><td>Second range channel configuration information</td></tr>
</table>

<pre>],
"CrossTalkRejection": {
"DitherEnabled": true,
"NoiseBlankingEnabled": true,
"RejectionLevel": "Low",
"TransmitChannel": {
"Mode": "Manual",
"Channel": 1
}
},
"RangeMode": "Dual",
"RotationSpeedMode": "Low",
"Transmit": false
}</pre>
	Additional radar information

You can view and change the configuration information for elements of each range channel separately by indicating the {Channel} (either 0 or 1) in the Get Command or Post Command. For example, if you want to adjust the gain of the radar for the first range channel of an open-array Fantom radar, you can first request the gain configuration for the channel using the configuration resource "/Radar/{RadarId}/Configuration/RangeChannels/{Channel}/Gain," and then post the change using the same configuration resource and providing the new value. For this example, the {RadarId} is "Fantom%20256," and the {Channel} is "0" for the first channel and "1" for the second channel.

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/RangeChannels/0/Gain**
Output	50.0
Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/RangeChannels/1/Gain**
Output	50.0
Post Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/RangeChannels/0/Gain -d 65**
Output	65.0
Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/RangeChannels/0/Gain**
Output	65.0
Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Configuration/RangeChannels/1/Gain**
Output	50.0

After completing the process in this example, the gain for the first channel (0) has been changed to 65, and the gain for the second channel (1) remains at 50.

Status Resources

The status resources allow you to read the status for various parts of the radar systems using the API. Using the status resources, you can read top-level resources and access granular, individual status sub-resources. These resources are parameterized so you can access various parts of the radar system.

To obtain a status for a connected radar, you must provide a Radar ID. You can obtain a Radar ID for a connected radar from the "/Radar" system resource (*System Resources*, page 2). When referencing the examples in this document and in the full API specification hosted on the unit, you should substitute the appropriate Radar ID in place of the {RadarId} path parameter shown in the example or specification. For example, accessing the status resource "/Radar/Fantom18/Status" returns the entire status for a connected Fantom 18 radar scanner with a Radar ID of "Fantom18."

To obtain only a portion of the status, you can read a sub-resource directly. For example, if you only want to read the state of the radar, you can request the "/Radar/{RadarId}/Status/State" resource. For this example, the {RadarId} is "Fantom%20256."

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Status**

Output	{
	"State": "Standby",
	"Rpm": 0.0,
	"Ranges": [
	232,
	463,
	926,
	1389,
	1852,
	2778,
	3704,
	5556,
	7408,
	11112,
	14816,
	22224,
	29632,
	44448,
	66672,
	88896,
	118528,
	133344,
	177792
	],
	"TransmitChannelMax": 4
	}

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Status/State**
Output	"Standby"

You should refer to the full API specification hosted on the device for the complete list of status resources (*Accessing Additional Specification Documentation*, page 1).

Stream Resources

Stream resources represent HTTP endpoints that deliver data continuously.

When using the system, configuration, and status resources, all API interactions are performed using a request/response transaction. For those transactions, the client establishes a TCP connection to the API service and makes an HTTP request to the API service. The API service processes the request, sends a response to the client, and closes the TCP connection. For stream resources, this method of communication cannot be used, because it creates excess overhead that makes it difficult to continuously deliver data from the API service to a client as the data becomes available.

Instead of using a request/response transaction, stream resources use chunked-transfer encoding. For stream transactions, the client establishes a TCP connection to the API service and makes an HTTP request to the API service. The API service processes the request and sends an initial response to the client indicating the use of chunked-transfer encoding. The API service then holds open the connection and sends individual JSON objects to the client as they become available. The JSON objects are pre-pended by the chunk length as outlined in section 4.1 of RFC 7230. As each chunk of data arrives at the client, the software processes the chunk payload as a JSON object.

Many networking software libraries support chunked-transfer encoding, so manual handling of chunked transfers is generally not necessary. Some software libraries do not provide a mechanism to service each chunk as it is received, and instead provide a result only upon completion of the chunk transfer. If your software library does not allow per-chunk processing, manual chunk handling is necessary. This will involve recording the chunk length, buffering incoming data up to the indicated chunk length, which may span multiple TCP transfers, and finally parsing that chunked JSON data.

NOTE: The API stream resources can transmit a very large amount of data to a client. It is important to use client hardware and software that can support a high load of data transfer when using the API stream resources.

As shown in this example using the cURL software library, you should use the “raw” command switch to allow for chunked-transfer encoding. If you request the stream resource in cURL without the "raw" command switch, the software does not display the result until after it receives all of the data from a chunk transfer. The “raw” command switch instructs cURL to display all of the data as it is received. Each raw data chunk represents a JSON spoke object and is preceded by a chunk length value. For this example, the {RadarId} is "Fantom %20256."

Get Command	**curl garmin-gsi.local/garmin/gsi/v1/Radar/Fantom%20256/Streams/SpokeData/0 –raw**

Output	698
	{
	"SampleCount": 640,
	"SpokeData": "AgMEBeQ……..==",
	"SpokeIndex": 502
	}
	698
	{
	"SampleCount": 640,
	"SpokeData": "CwaCgsM……..==",
	"SpokeIndex": 503
	}
	……

TIP: Due to the large amount of data sent from the API stream resource, the output from this example will quickly flood a user console. You should be prepared to use **CTRL-C** to shut down cURL quickly after starting the stream to limit the total amount of data printed to the console. You can send the output to a file for later viewing.

Simulator Mode

You can enable simulator mode to support development before a physical radar device is available to connect to the network. You can enable or disable the simulator endpoint by sending a Post command to /Simulator/Enable.

NOTE: For the radar simulator mode to work correctly, you must disconnect all radar devices from the system.

All radar endpoints are accessible using the simulated source, with two considerations.

- Simulator radar configuration changes do not affect the spoke stream data.

  Attempted modifications to a simulated radar's configuration are not reflected in the spoke stream. Each simulator source contains unique range channels and streams of radar data.
- Simulator requests for a radar set the simulator's context.

  For example, if a request is made to a radar endpoint for the “GRM Fantom 4,” the simulator operates in that simulated source's context, including its data streams. If a subsequent request is made to any radar endpoint using the “GMR 1224 xHD2” simulated radar source, the simulator context changes all simulated configurations, statuses, and streams to that radar source.