

Deviceworx Technologies Inc.

xGATWEAY User's Guide

Table of Contents

Revisions.....	5
Document Overview.....	6
Functionality.....	6
xGATEWAY Models.....	7
xGATEWAY PURE.....	7
Specifications.....	7
Connections.....	7
Mounting.....	8
xGATEWAY HaLow/LTE.....	9
Specifications.....	9
Connections.....	10
Mounting.....	10
xGATEWAY 3.0 Details.....	11
Connections.....	12
Mounting.....	12
Quick Start.....	13
Linux OS Support.....	14
SSH Access.....	14
SFTP Access.....	15
Package Management.....	15
xGATEWAY Daemon Software.....	15
Startup and Process Management.....	15
systemd daemon Management.....	16
Wiring Diagrams.....	18
Point to Point RS-232 Full Duplex Wiring.....	18
Multi-Drop RS-485 Half Duplex Wiring.....	19
Multi-Drop RS-485/422 Full Duplex Wiring.....	20
Cloud Dashboard.....	21
Device Setup.....	21
Getting Started.....	21
Logging In.....	21
Overview Screen.....	22
Changing your Password.....	22

Management Pages.....	23
Sites.....	23
xGATEWAYS.....	25
xGATEWAY Radius Modes.....	27
xTAGs.....	28
API Keys.....	32
Users.....	33
Reports.....	36
Timeframe & Date.....	36
Scope.....	37
Temperature Unit.....	37
Threshold.....	37
Plot Controls.....	37
xGATEWAY Custom App Dev Support.....	38
.NET Framework Support.....	45
xGATEWAY Socket Interface.....	47
xtagusb and xtaglmod Interface Differences.....	48
xTAG Sensor Streaming Vs Sampling.....	48
xTAG Sensor Calibration.....	49
Command Formatting.....	49
Command Response and Unsolicited Message Formatting.....	49
Error Responses.....	50
xtagusb Communications.....	50
Typical xtagusb Command Sequences.....	50
Socket Interface Commands Specific to xtagusb (USB xTAGs).....	51
List xTAGs (0x02).....	51
xTAG Connect (0x03).....	52
xTAG Disconnect (0x04).....	52
xTAG Metadata Read (0x05).....	53
xTAG Metadata Write(0x06).....	54
HaLow Communications.....	54
Typical xtaglmod Command Sequences.....	55
HaLow Config and Sample Data Store and Forward.....	55
Socket Interface Commands Specific to xtaglmod (HaLow xTAGs).....	57
xTAG HaLow Metadata Heartbeat (0x0A).....	57

xGATEWAY HaLow Metadata Heartbeat (0x1A).....	58
xTAG Metadata Read (0x0D).....	59
xTAG Metadata Write (0x0E).....	60
xTAG FOTA Update (0x0F).....	61
Socket Interface Command Supported by All Daemons.....	62
xGATEWAY-Specific Commands.....	62
xGATEWAY Metadata Read (0x1B).....	62
xGATEWAY Metadata Write (0x1C).....	63
Shutdown (0xFF).....	64
xTAG-Specific Commands.....	65
xTAG Sensor Acq Start (0x18).....	65
xTAG Sensor Acq Stop (0x19).....	65
xTAG Accelerometer Config Read (0x23).....	66
xTAG Accelerometer Config Write (0x24).....	67
Accelerometer Acquisition Stream Start (0x26).....	69
Accelerometer Acquisition Stream Data (Responses Only) (0x2A).....	72
Accelerometer Acquisition Stream Stop (0x27).....	73
xTAG Environmental Config Read (0x40).....	74
xTAG Environmental Config Write (0x41).....	75
xTAG Environmental Sample Read (0x4A).....	76
xTAG CO2 Config Read (0x50).....	77
xTAG CO2 Config Write (0x51).....	78

Revisions

Rev	Author	Notes	Date
1.0	Mark Janke	Original Draft.	Aug 13, 2020
1.1	Mark Janke	Added clarifications on return values and notes per customer Q's.	Aug 25, 2020
1.2	Mark Janke	Updated for xtagd socket connections and disconnection changes.	Aug 27, 2020
1.3	Mark Janke	Changed stream start cmd to support start after motion or no-motion. Added wiring diagrams for RS-232/422/485 comms.	Sept 17, 2020
1.4	Mark Janke	Updated switch pics to new default of slew rate unlimited.	Sept 28, 2020
1.5	Mark Janke	Updated to include xtagbled and xtagusbd support details.	Oct 6, 2020
1.6	Mark Janke	Updated to support release of xTAG firmware and xGATEWAY firmware and daemons.	Nov 12, 2020
1.7	Mark Janke	Removed typo within xTAG Acc Acquisition Config (0x14).	Nov 18, 2020
1.8	Mark Janke	Added plugged stream msg details (0x17)	Nov 19, 2020
1.9	Mark Janke	Clarified ODR <= 100 s/s on delayed stream.	Dec 14, 2020
1.10	Mark Janke	Updated SSH login details.	April 4, 2021
2.1	Mark Janke	Updated to Include API Changes for HaLow, Environmental and Gas Sensor API Elements	Feb 2, 2023
2.2	Nathan Janke	Update Dashboard elements (internal release only).	Aug 2, 2023
2.3	Mark Janke	Update Socket Interface and clean up dashboard elements.	Sept 1, 2023
2.4	Mark Janke	Updated Socket Interface message details and general cleanup.	Sept 17, 2023
2.5	Mark Janke	Misc updates to the cloud doc, added diagrams.	Sept 19, 2023
2.6	Mark Janke	Misc updates after client review.	Sept 21, 2023
2.7	Mark Janke	Misc protocol updates.	June 17, 2024

Document Overview

This document serves to provide xGATEWAY user's with information required to setup and use xGATEWAY devices. Note that this same document covers all revisions of xGATEWAY products.

Functionality

The xGATEWAY is an Industrial-grade Internet of Things (IIoT) Gateway device. It is used to collect data from Deviceworx xTAGs via WiFi HaLow (1000's of xTAGs) or USB (up to 20 xTAGs). It can push xTAG data to the Deviceworx Cloud or provide access to it on a LAN via a Socket Interface or API. Additionally, the xGATEWAY can send and receive data from up to 32 legacy industrial devices that utilize industry standard interfaces including RS232, RS422 and RS485. This connectivity can provide legacy industrial devices with Cloud-based status, reporting and even control to retrofit IoT functionality into legacy hardware. Lastly, the xGATEWAY provides IoT "Edge" functionality (i.e. it can be used to condition data including statistical analysis, AI model-based data predictions, alarming and local control).



xGATEWAY Models

xGATEWAY PURE

The xGATEWAY PURE is the first xGATEWAY developed by Deviceworx. It includes core IoT functionality, but is limited to USB communications. It has been replaced by newer, more capable 2nd and 3rd generation products.

Specifications

- 340 g or 12 oz
- 110mm (4.33") wide, 90mm (3.5") long, 50mm (2") tall
- Quad Core ARM (Cortex-A53) Processor at 1.8 GHz
- 2 GB RAM and 16 GB Flash Memory
- Debian 10 with
 - gcc, gdb, dpkg (apt), npm, pm2 ssh and sftp, npt utilities and tools
 - .NET 5.0 and Visual Studio Support
- Wi-Fi and Ethernet Support for Cloud Connections
- USB Host at 12 Mbps with 5m cable length (increase to 100m with USB extension)
- 5 Vdc Power (3A) or 12 Vdc power (1.5A)
- RJ45 10/100/1000 Ethernet Connection
- Panel mount enclosure
- -35°C to 85°C operating temperature

Connections

Device connections are shown within the picture below.



Mounting

The xGATEWAY is typically mounted within a field enclosure on that enclosure's back panel. To facilitate this, accessory mount tabs are provided with each xGATEWAY upon customer request.



To install the tabs, simply remove the top end cap of the enclosure that holds the antenna, along with its plastic cover. This is done by removing the 4 screws holding both the plastic cover and end cap. Then, simply slide the tabs into the enclosure side channels (each side) and replace the top end cap. Note that the antenna lead should not be removed while the end cap and plastic cover have been removed. The tabs will be loose and slide up and down within the side channels until the enclosure is mated to a back panel and secured by screwing the tabs to the back panel.

xGATEWAY HaLow/LTE

Second generation xGATEWAY models include a mini PCI express (mPCIe) slot. This slot typically houses either a WiFi HaLow module (xGATEWAY HaLow) or LTE module (xGATEWAY LTE).

xGATEWAY HaLow supports the wireless connection of 1000's of xTAG HaLow Sensors over long distances (2+ km). See the xTAG Sensors page at www.deviceworx.com for more information on xTAG HaLow Sensors. Note that xGATEWAY HaLow can be connected through an LTE network to the cloud, but an external LTE Modem is required. Deviceworx can sell these external modems with xGATEWAY HaLow devices when required, along with supporting cabling, power supplies and other accessories.

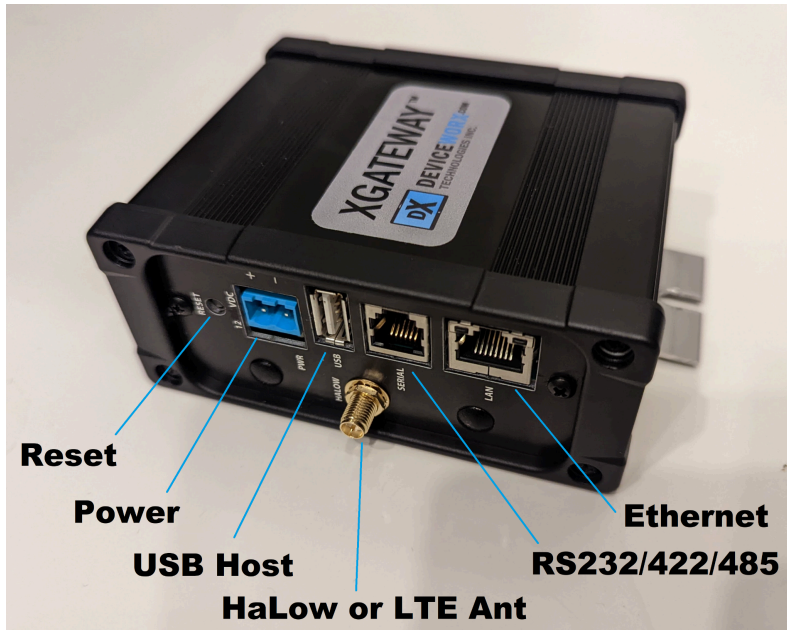
xGATEWAY LTE can be used whenever xTAG USB Sensors are deployed with an xGATEWAY and a clean LTE cloud connection is required. Deviceworx packages an mPCIe LTE modem within the xGATEWAY to support built-on LTE cloud connections (no external LTE modem is required).

Specifications

- 340 g or 12 oz
- 110mm (4.33") wide, 90mm (3.5") long, 50mm (2") tall
- Quad Core ARM (Cortex-A53) Processor at 1.8 GHz
- 2 GB RAM and 16 GB Flash Memory
- Debian 10 with
 - gcc, gdb, dpkg (apt), npm, pm2 ssh and sftp, npt utilities and tools
 - .NET 5.0 and Visual Studio Support
- LTE and Ethernet Support for Cloud Connections
- USB Host at 12 Mbps with 5m cable length (increase to 100m with USB extension)
- 12 Vdc Power (2.0 A) or 24 Vdc power (1.0 A)
- RJ45 10/100/1000 Ethernet Connection
- RJ11 RS-232/485
- CAN Bus Support
- TPM Module with TPM 2.0 Support
- DIN Rail mount enclosure
- -35°C to 85°C operating temperature

Connections

Device connections are shown within the picture below.



Mounting

The xGATEWAY HaLow/LTE is typically mounted within a field enclosure on that enclosure's back panel. To facilitate this, accessory mount tabs are provided with each xGATEWAY HaLow/LTE upon customer request. See an example tab within the pic above.

To install the tabs, simply remove the top end cap of the enclosure that holds the antenna, along with its plastic cover. This is done by removing the 4 screws holding both the plastic cover and end cap. Then, simply slide the tabs into the enclosure side channels (each side) and replace the top end cap. Note that the antenna lead should not be removed while the end cap and plastic cover have been removed. The tabs will be loose and slide up and down within the side channels until the enclosure is mated to a back panel and secured by screwing the tabs to the back panel.

xGATEWAY 3.0 Details

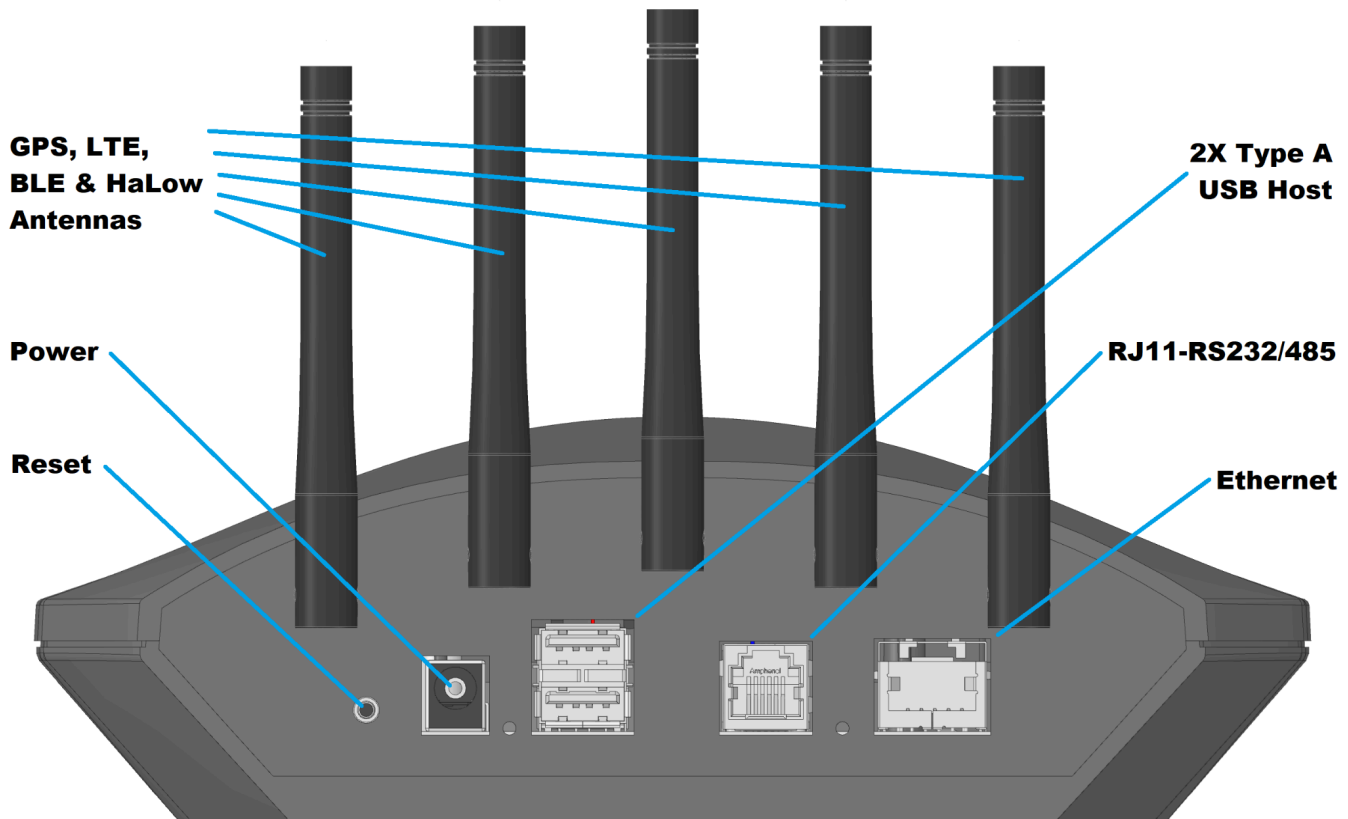
The new third generation xGATEWAY (xGATEWAY 3.0) further improves upon the second gen xGATEWAY LTE and xGATEWAY HaLow designs. It includes a mini PCI express (mPCIe) slot but does not require use of this slot to support WiFi HaLow. WiFi HaLow support has been moved onto the main printed circuit assembly. This slot can be used to support an LTE module giving operators the ability to cleanly support both WiFi HaLow and LTE connections without the need to install an external LTE modem. Additionally, the xGATEWAY 3.0 supports enhanced security (latest TPM module tech), and built-in GPS for telematics applications. The only feature removed from the second generation xGATEWAY is a CAN bus transceiver. To connect CAN bus peripherals to the xGATEWAY 3.0, a CAN bus mPCIe module may be used.

Specifications

- 454 g or 16 oz
- 195mm (7.7") wide, 160mm (6.3") long, 1100mm (4.3") tall
- Quad Core ARM (Cortex-A53) Processor at 1.8 GHz
- 2 GB RAM and 16 GB Flash Memory
- Debian 10 with
 - gcc, gdb, dpkg (apt), npm, pm2 ssh and sftp, npt utilities and tools
 - .NET 5.0 and Visual Studio Support
 - AI Support via TensorFlow Lite, Onnx, Python, Java ...
- Wi-Fi Classic (to 6e), LTE and Ethernet Support for Cloud Connections
- Wi-Fi 802.11ah HaLow @ 1 MBps with FEM
- GNSS (GPS, GLONASS, Galileo, BeiDou) with Real Time Clock (RTC) battery backup
- USB Host 2.0 at 480 Mbps with 5m cable length (increase to 100m with USB extension)
- 12 Vdc Power (3.0 A) or 24 Vdc power (1.5 A)
- RJ45 10/100 Ethernet Connection
- RJ11 RS-232/485
- TPM Module with TPM 2.0 Support
- Desktop and DIN Rail mount enclosure for panel and waterproof installations
- -40°C to 85°C operating temperature

Connections

Device connections are shown within the picture below.



Mounting

The xGATEWAY 3.0 has 4 mount holes on its back supporting the attachment of 2 standard DIN rail mounts. These DIN rail mounts can be pre-installed on xGATEWAY 3.0 devices by Deviceworx as an option. Reach out to Deviceworx Sales to order this option with xGATEWAYs or to retrofit this option to xGATEWAY 3.0 devices that have already been purchased. Note that the xGATEWAY 3.0 is configured to sit on a desktop when these rail mounts are not installed.

Quick Start

To install and start an xGATEWAY device, follow these steps:

1. Mount the xGATEWAY as discussed in previous sections for xGATEWAY PURE [here](#), xGATEWAY HaLow/LTE [here](#) or xGATEWAY 3.0 [here](#).
2. Install xGATEWAY antennas provided with the xGATEWAY. Note that antenna labels are provided on the xGATEWAY enclosure. Ensure that antennas are only “hand tight”. If possible point antennas straight up or straight down as 90 degrees to their length is the direction of max transmission. Keep antennas at least 6” from metal objects including structural steel, metal walls, etc. Note that, for desktop testing where xTAG HaLow Sensors are closer than 3 m or 10 feet from xGATEWAYS, HaLow antennas should be left disconnected as their strength may be too great for this short distance.
3. Plug in any interconnect cabling including an Ethernet cable to an external LTE modem or LAN, a serial cable to any RS-232 or RS-485 connected equipment or USB cable to any xTAG USB Sensors or external GPS devices. As serial connections can require custom cabling and (in some cases) termination resistance, a [separate Wiring Diagrams section](#) details serial connection support.
4. Plug in an xGATEWAY power supply (included with the xGATEWAY). Watch the xGATEWAY LEDs flash a few times when Deviceworx software starts automatically after Linux boots on the xGATEWAY. This typically takes approximately 10 seconds after power is cycled to the xGATEWAY or its reset button is pressed.
5. Interact with the xGATEWAY and nearby xTAG Sensors using the Deviceworx [Cloud Dashboard](#) or [Socket Interface](#). xGATEWAY devices are pre-configured, before shipping to customers, for either cloud dashboard support or exposure of a Socket Interface. This setup can, however, be changed at any time by customers by simply changing how the xGATEWAY daemon software, pre-installed on xGATEWAYS, is started. This is detailed within the [Startup and Process Management](#) section that follows.

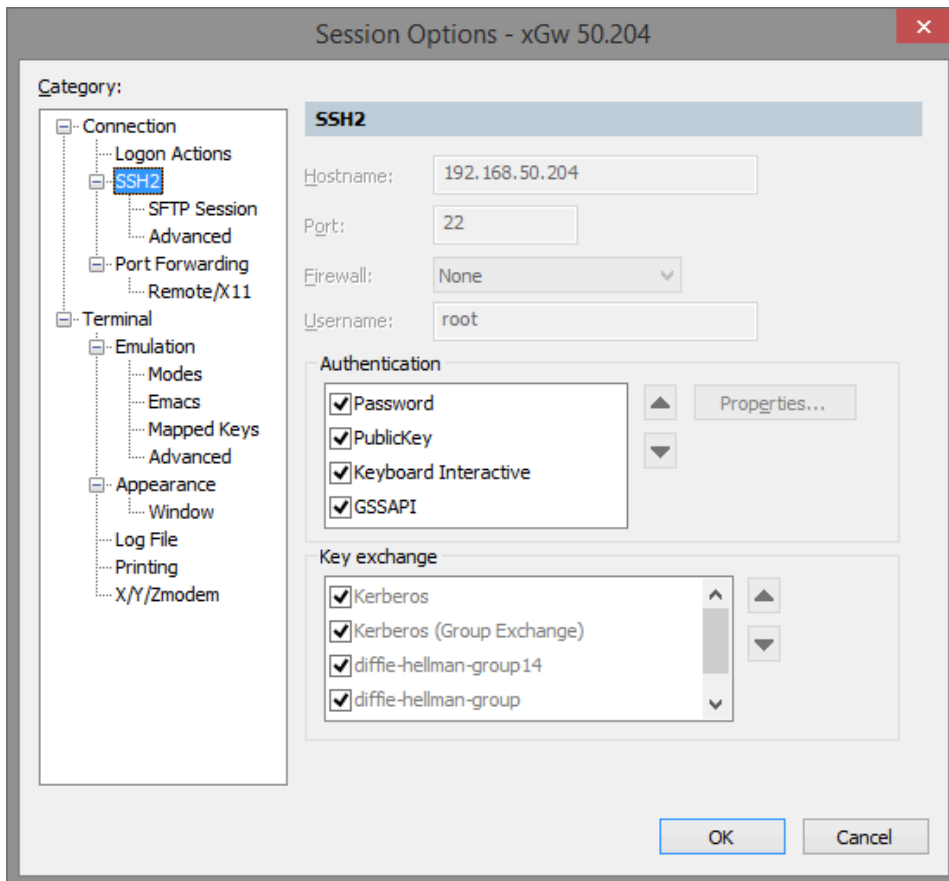
Linux OS Support

The xGATEWAY runs Debian Linux (version 10). xGATEWAYS are often used by operators without the need for Linux access (i.e. when using only pre-installed Deviceworx software on the device). If required, however, operators may access Linux and run their own apps on the xGATEWAY.

SSH Access

To connect to an xGATEWAY using SSH, ensure that the xGATEWAY has a good Ethernet LAN connection. Next, determine the IP address of the xGATEWAY by reviewing DHCP allocated or statically allocated addresses on your LAN. DHCP servers typically list assigned IP addresses, along with a device label. xGATEWAYS are labeled as “imx8mm-var-dart”.

Setup a SSH client connection to the xGATEWAY (SSH2 to be specific) using IP address, standard SSH port 22, username “root”, password “root” and authentication features as shown in the example SecureCRT setup below:



After saving authentication, no user and password entry will be required upon SSH re-connection. Ensure that the SSH client is setup to utf-8 character encoding (sometimes called “locale”). This is required by some Linux terminal apps such as “top”.

SFTP Access

Any industry standard SFTP client can be used to send/receive files to/from an xGATEWAY. The best option is the widely available FileZilla client. Ensure that port 22 is used when setting up an SFTP client and that user “root” and password “root” are used.

Package Management

“Out of the Box”, xGATEWAY devices support the apt Debian/Ubuntu package manager. For more information on how to use apt, consult [this online reference](#). Note that no “sudo” precursor to “apt” is required when logged in as root over SSH. Please ensure that the command “apt update” is issued before installing any additional packages.

Additional package managers for Debian can be installed by apt. For example, install the Node Package Manager (NPM) using “apt install npm”. NPM (and other package managers) can complement the packages supported by apt.

xGATEWAY Daemon Software

xGATEWAYS are responsible for communicating with xTAGs locally and linking those xTAGs to the Deviceworx cloud-based Dashboard or client application via a Socket Interface. Separate xTAG daemon apps support each xTAG communication option. “xtagusb” supports xTAG USB Sensor comms. “xtaghlod” supports xTAG HaLow Sensor comms. Multiple daemons can be run at the same time to simultaneously support different xTAG connection types. Daemons are started automatically as discussed within the [systemd daemon Management](#) section that follows. Deviceworx customers that use the Deviceworx Dashboard will never have to directly interact with the daemons and can even ignore sections that follow discussing their startup and management. Deviceworx customers using the Socket Interface will not have to change daemons, but understanding how they start and support the Socket Interface gives users of the interface useful context.

Startup and Process Management

If xGATEWAY operators choose to deploy their own app(s) to the device, there are options to support automatic app launch and management.

The /etc/rc.local file supports startup commands that will execute at the end of the device startup cycle. Edit this file using vi or install another text editor if required. Add any commands as necessary.

systemd daemon Management

Deviceworx pre-installed daemon software is managed by systemd as “service” units. Systemd starts daemon software when the device starts (after prerequisite software elements are running), restarts daemon software if it stops, and sets environment variables used by daemon software. To support this functionality, Deviceworx deploys a .service file for each daemon. An example service file supporting the xtaghlod daemon (xtaghlod.service) is below.

```
[Unit]

Description=xtaghlod service

After=network.target sockets.target time-sync.target chrony.service

Wants=network-online.target


[Service]

Restart=always

RestartSec=10

TimeoutStopSec=5

Type=simple

ExecStartPre=chronyc waitsync

#Local interface only - ExecStart=/root/xtaghlod

#Additional cloud interface - ExecStart=/root/xtaghlod cloud

ExecStart=/root/xtaghlod cloud

#Example keys follow (only required when a cloud interface is used).

# Test Client Key:
Environment='DW_CLOUD=aadfadfafafaFxG2VRor9CK93D2Llj36UNp70RtL6c'


[Install]

WantedBy=multi-user.target
```

Note the following for daemon .service files:

1. This file is installed, along with other service files, at /etc/systemd/system/
2. The ExcStart variable must be set to start the daemon with Deviceworx Cloud support (include "cloud" daemon argument) or without this support (do not include "cloud"). See the example file above.
3. Deviceworx will provide each customer with their own secure and private Dashboard Client Key and preset .service files to set this key as an environment variable. This key is used (by pre-installed daemon software) to securely connect to the Deviceworx Dashboard (i.e. it reads the DWCLOUD Environment variable). Note that even customers using only the local Socket Interface can use a private Client Key - supporting their access to xTAG Firmware Over The Air (FOTA) updates.

To setup the service to start automatically, execute: `systemctl enable xtaghlod`

Restart the xGATEWAY after this command has been issued (softly using `reboot now`). After restart and login, the xtaghlod daemon should be observed as automatically started.

To disable systemd autostart or restart, execute: `systemctl disable xtaghlod`

Wiring Diagrams

The xGATEWAY primary serial device can be connected to a single secondary serial device via RS-232, or to a “chain” of up to 32 devices via RS-485/422. Separate sections below outline how to configure the xGATEWAY for distinct serial communication configurations as well as how to wire up serial connection within these configurations.

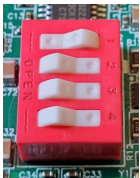
Note that color coding of each diagram matches a common 6 core RJ-11 standard. This standard is implemented within a low-cost cable sold by Digikey (Part Number: A3662R-07-ND) and accessible here:

[ASSMAN AT-S-26-6/6/B-7-OE Cable](#)

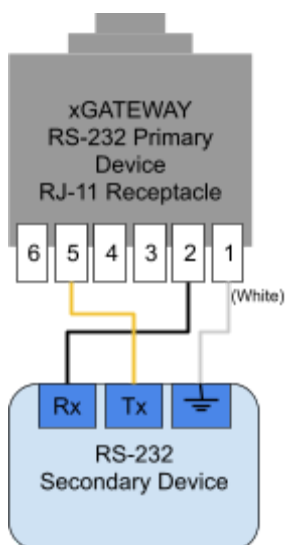
Note that diagrams that follow show the receptacle pinout on the xGATEWAY. The pinout for the plug on the cable will be numbered in reverse order (Pin 1 on the left).

Point to Point RS-232 Full Duplex Wiring

To configure the xGATEWAY to support full duplex RS-232, set on board switches to match that in the picture below (POS1 OPEN, POS2, POS3 & POS4 CLOSED).

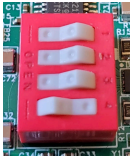


The diagram below shows how to connect a single RS-232 secondary device to the xGATEWAY's serial RJ-11 Connector.

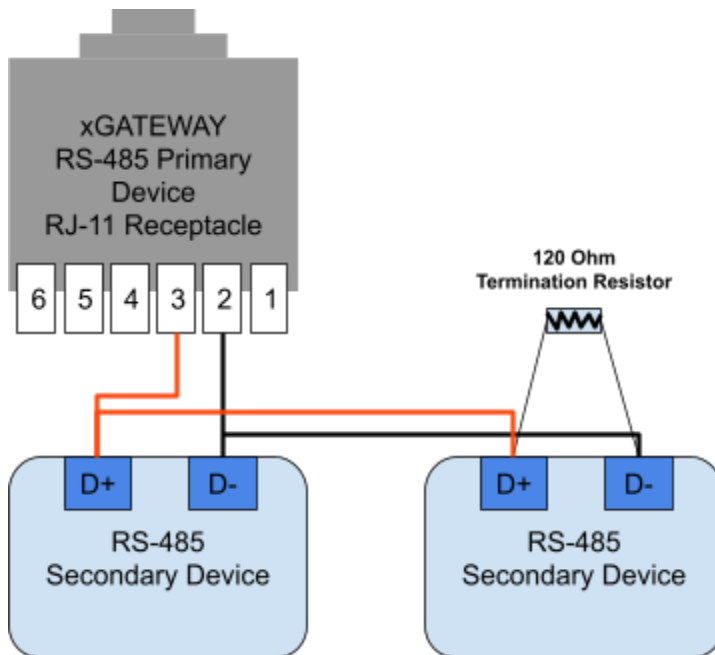


Multi-Drop RS-485 Half Duplex Wiring

To configure the xGATEWAY to support half duplex RS-485, set on board switches to match that in the picture below (POS1-3 OPEN, POS4 CLOSED).



Many RS-485 secondary devices including panel controllers, Programmable Logic Controllers (PLCs) etc, utilize simple half-duplex wiring. The diagram below shows how to connect 1 or more RS-485 secondary devices (up to 32) to the xGATEWAY's serial RJ-11 Connector.

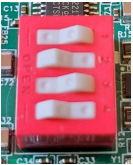


Note that the 120 Ohm termination resistor shown above is only required for very long connections or to help limit EMI (a shielded cable grounded at one end may help with EMI as well). When this termination resistance is added to the last multi-drop connection, also enable the 120 Ohm line termination resistance within the xGATEWAY by installing J4 on the board (installed by default behind the RJ-11 connector) as shown here:

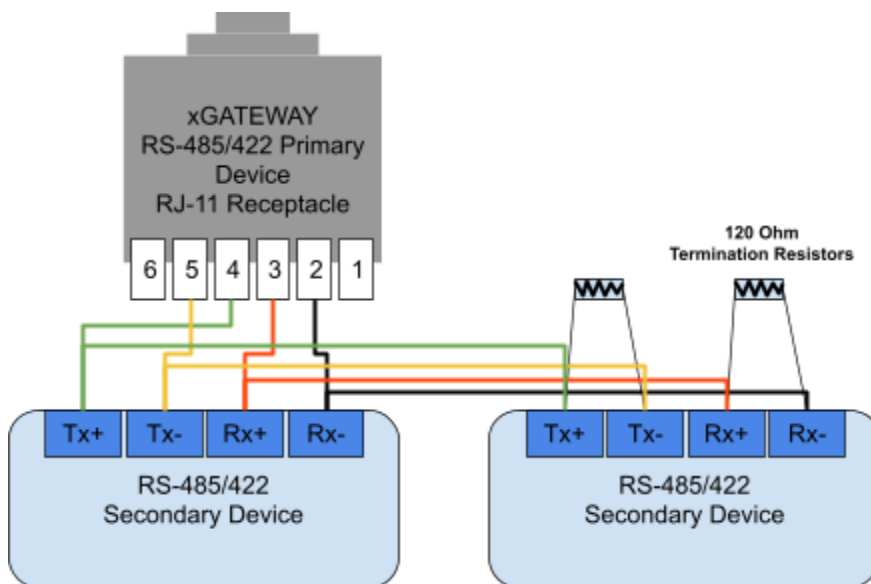


Multi-Drop RS-485/422 Full Duplex Wiring

To configure the xGATEWAY to support full duplex RS-485/422, set on board switches to match that in the picture below (POS1 OPEN, POS 2 CLOSED, POS3 OPEN, POS4 CLOSED).



The diagram below shows how to connect 1 or more RS-485/422 secondary devices (up to 32) to the xGATEWAYs serial RJ-11 Connector.



Note that the 120 Ohm termination resistors shown above are only required for very long connections or to help limit EMI (a shielded cable grounded at one end may help with EMI as well). When this termination resistance is added to the last multi-drop connection, also enable the 120 Ohm line termination resistances within the xGATEWAY by installing **J4 and J6** on the board (installed by default behind the RJ-11 connector) as shown here:



Cloud Dashboard

Monitoring, Configuration, and Management of xGATEWAYSs and xTAGs is possible through the Deviceworx Cloud Dashboard. Customers can leverage this Dashboard to get xGATEWAYSs and xTAG Sensors up and running and collecting meaningful data in a few minutes.

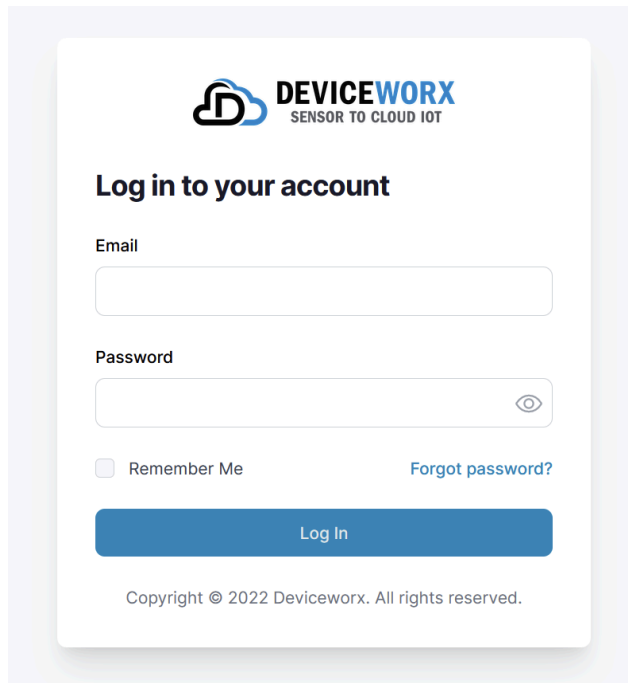
Device Setup

No customer setup is required on xGATEWAYSs or xTAGs for Dashboard use whenever customer requests Dashboard access when purchasing hardware. Deviceworx presets devices under customer accounts so that they can “see” devices after installation - providing that a cloud connection is available to xGATEWAYSs. This provision is as simple as an internet connection supporting secure browsing (i.e. web browsing over port 443).

Getting Started

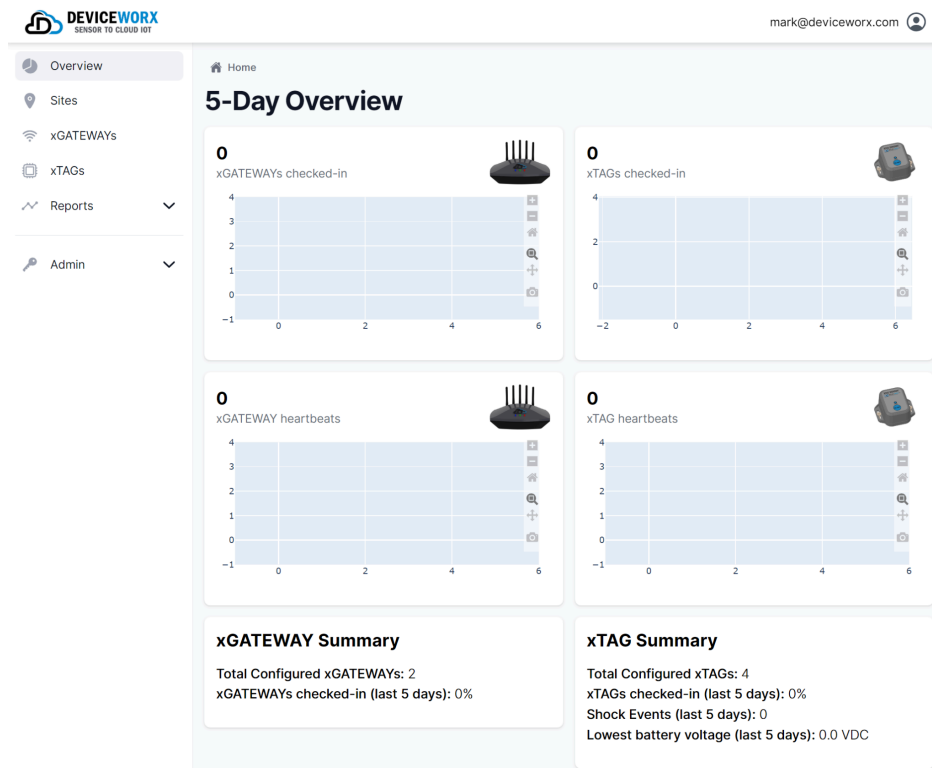
Logging In

The dashboard can not be accessed without valid login credentials. Deviceworx will provide credentials when devices ship to customers. Once you have been provided with login credentials, use them to access the dashboard at <https://dash.deviceworx.com>.

The image shows a login form for the DeviceWorx Cloud Dashboard. At the top is the DeviceWorx logo with the tagline "SENSOR TO CLOUD IOT". Below the logo is the heading "Log in to your account". There are two input fields: "Email" and "Password". The "Password" field has a toggle icon (an eye) to the right of it. Below the "Email" field is a checkbox labeled "Remember Me". To the right of the checkbox is a link labeled "Forgot password?". Below these elements is a blue "Log In" button. At the bottom of the form is a copyright notice: "Copyright © 2022 Deviceworx. All rights reserved."

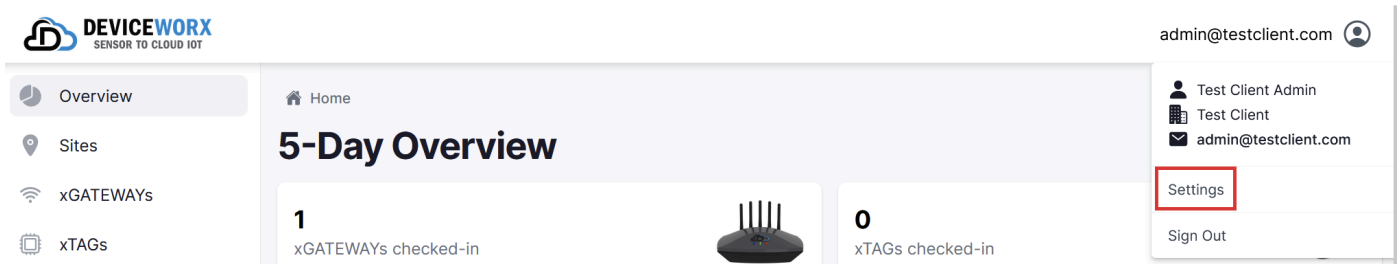
Overview Screen

After logging in, an Overview page will be displayed. This page shows general information about xGATEWAYS and xTAGs including how many devices have “checked-in” or communicated with the Dashboard in the last few days and sent “heartbeat” data to the Dashboard. Additionally, the number of heartbeats sent is shown. This data provides users with a general sense of device connectivity to the Dashboard.



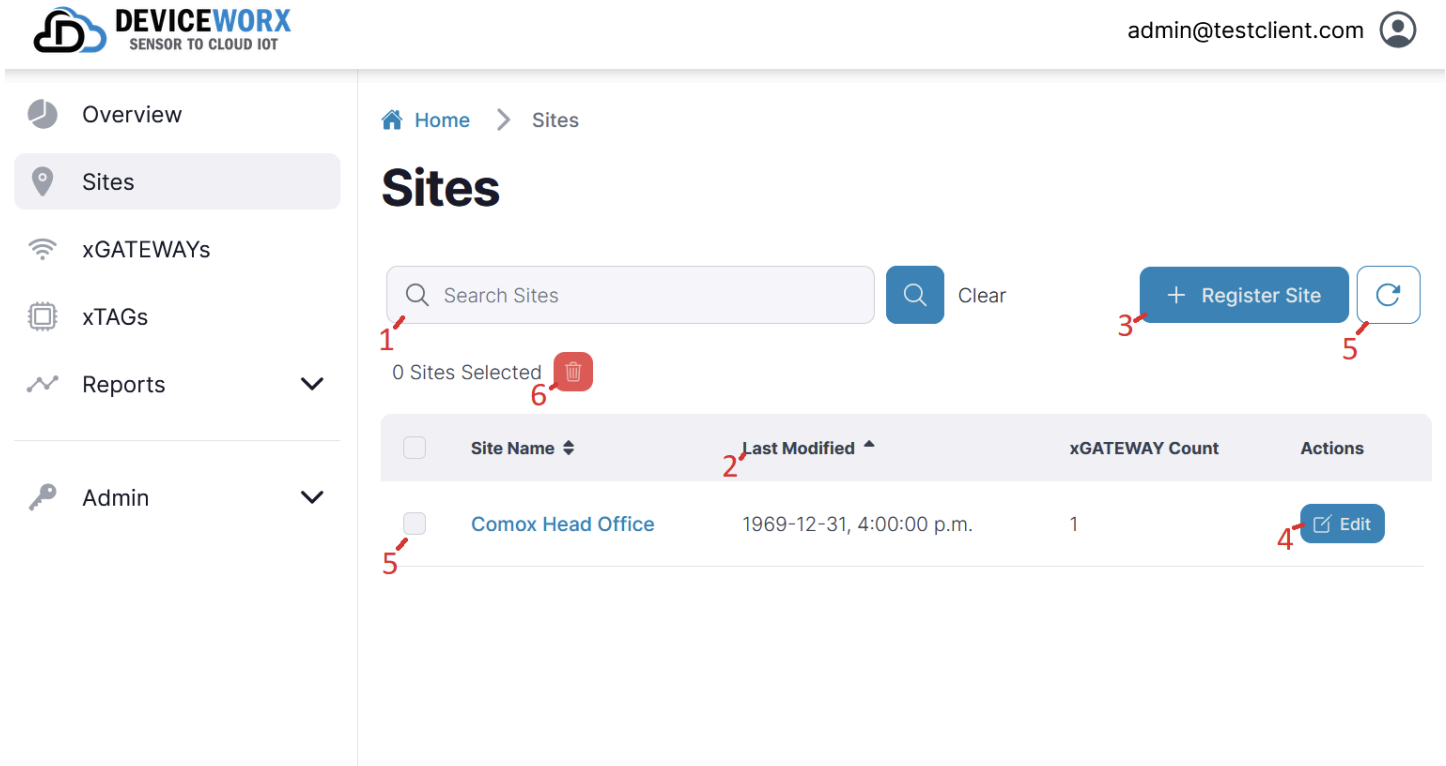
Changing your Password

After successfully logging in for the first time, it is strongly recommended that you change your password. To do this, select the user icon on the top-right of the screen and click “Settings”. Change your password on the page that appears.



Management Pages

Similar management pages exist within the Dashboard for managing Sites, xGATEWAYS, and xTAGs. Additionally, organization administrators have access to management pages for API Keys and Users.



An example of a management page (“Sites”) is shown above. Management pages provide a sortable, searchable data table as well as controls to edit/delete items. Rows can quickly be filtered by entering text in the search bar (1). Certain columns can be sorted by clicking anywhere on the column header (2). If the authenticated user has modification permissions, new entries can be added by clicking on the Add/Register button (3) (e.g. “Add Site” or “Register Site”), or edited by clicking on the “Edit” button within the row (4). On certain pages, multiple rows can be selected at once using the checkboxes at the beginning of each row (5). For example, with multiple rows selected, multiple items are deleted at once using the batch delete button (6).

Sites


The “Sites” management page allows registration, modification, and deletion of sites. A site represents a geographical location in which many xGATEWAYS are located. xGATEWAYS can be assigned to sites upon xGATEWAY registration, or by editing an xGATEWAY’s details on the xGATEWAY management page.


When a site is removed, all xGATEWAYS associated with the site will no longer be associated with any site.

Sites provide several features to users.

1. Filter xGATEWAY listings. To see only xGATEWAY devices at a site, a site filter can be selected within the xGATEWAYS screen. For example, if you only want to view xGATEWAYS linked to the *Comox Head Office* site, enter *Comox* within the Search edit box and only xGATEWAYS linked to the *Comox Head Office* will be shown.
2. Filter Report values. To only see report values for data acquired by xGATEWAYS that are linked to a specific site, select the *Site* option under the Report screen *Scope* selection. A *Site* list box will appear and support selection of a Site within that list box. Report data will be limited to xGATEWAYS linked to the selected Site.
3. Site mapping. The Deviceworx Dashboard supports the explicit location of anchored (permanently mounted) xGATEWAYS by specifying a latitude and longitude value for each xGATEWAY. Additionally, coverage details (based on where an xGATEWAY is mounted within a building) can be specified for each xGATEWAY (See radius definitions within xGATEWAY registration and edit controls below). When xGATEWAYS with anchor and coverage details specified are linked to a site and a map file is available for that site (with known latitude and longitude values for each corner of the map), xGATEWAYS and their coverage can be added to the map. Note that mapping of xGATEWAY locations and coverages is not supported by the Dashboard (it may be added in the future), but data supporting this mapping can be pulled from the Dashboard via a web service. This supports mapping within 3rd party screens. Consult Deviceworx sales (sales@deviceworx.com) for details on how this data can be acquired through a web service.

xGATEWAYS


DEVICEWORX
 SENSOR TO CLOUD IOT

admin@testclient.com 

Overview

Sites

xGATEWAYS





xTAGs


Reports

Admin

Home > xGATEWAYS

xGATEWAYS

0 xGATEWAYS Selected    

<input type="checkbox"/>	Device ID ^	Name ^	Heartbeat ^	Lat, Lon	Command	Last Edited ^	Actions
<input type="checkbox"/>	f8:dc:7a:59:1f:64	Comox xGw 1	2023-08-01, 1:16:46 p.m.	49.6749, -124.89336	 2023-05-03, 10:55:19 a.m.	2023-05-03, 9:31:29 a.m.	<input type="button" value="Edit"/>

The “xGATEWAYS” page shows all xGATEWAYS currently registered within the organization. Columns in the table show details about each xGATEWAY such as the latest heartbeat and status of the last dispatched command. Clicking on the first column of a row (Device ID) will show more details about the selected xGATEWAY. Clicking on the Lat/Lon coordinates will reveal the location of the coordinates on a Google map.

When one or more xGATEWAYS are selected, a configuration or firmware update can be issued via the appropriate button above the table on the left.

The command status will show a yellow icon when a command has been dispatched and no confirmation has yet been received. The status will show a green icon if a successful confirmation has been received, and red if an error confirmation has been received.

When an xGATEWAY is deleted, all associated heartbeat records are deleted from the database.

To edit an xGATEWAY, use the Edit button.

Edit xGATEWAY
 ×

Device ID <input type="text" value="f8:dc:7a:7a:15:36"/>	Name <input type="text" value="xGATEWAY - Cab Connect Tester"/>
Site <input type="text" value="Lucerne Valley"/>	
Commission Date <input type="text" value="2023-08-17, 10:28 a.m."/>	Channel <input type="text" value="7 (BW: 1 MHz)"/>
Transmit Power (dBm) <input type="text" value="15"/>	Timezone <input type="text" value="Not Specified"/>
User Group ID (optional) <input type="text"/>	User Device ID (optional) <input type="text"/>
Anchor Latitude <input type="text" value="34.4068607"/>	Anchor Longitude <input type="text" value="-116.8637422"/>
<input checked="" type="radio"/> Simple Radius <input type="radio"/> Complex Radii ?	
Circular Radius (m) <input type="text"/>	
Notes (optional) <input type="text"/>	
<input type="button" value="Apply Changes"/>	

The “Edit xGATEWAY” form provides controls for modifying xGATEWAY configurations and metadata:

- **Device ID:** The ID (MAC address) of the device.
- **Name:** A name assigned to the device for internal reference.
- **Site:** An optional site association for organization and report grouping.
- **Commission Date:** The date the device was commissioned.
- **Channel:** A WiFi HaLow channel to use. Channels can be 1 through 11 excluding 4. Use 7 as a good default value.
- **Transmit Power:** Transmit power can be set between 10 and 18 dbm. Set this value to 10 when xTAG connections will be < 50m and select 18 when connections are greater than 200m Otherwise, 15 is a good power setting to support a 50-200m range.
- **Timezone:** The time zone the device resides within.
- **User Group ID:** An optional numeric ID for grouping.
- **User Device ID:** An optional numeric ID for reference.

- **Anchor Latitude/Longitude:** The device's fixed latitude and longitude (in degrees with decimal values (no min or second values)).
- **Simple Radius / Complex Radii:** See "xGATEWAY Radius Modes" below.
- **Notes:** Optional notes for reference.

xGATEWAY Radius Modes

☐ Simple Radius
 ☒ Complex Radii ?

North Offset (degrees)

North-West Radius (m)

North-East Radius (m)

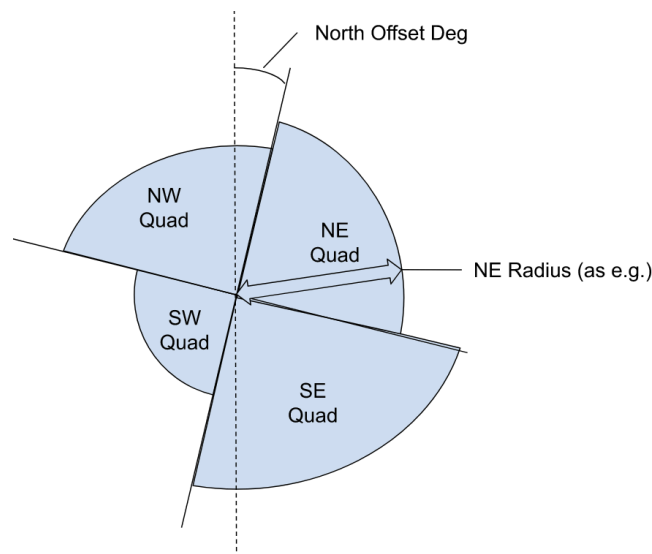
South-West Radius (m)

South-East Radius (m)


Notes (optional)

Register

When configuring an xGATEWAY, one of two radius modes can be used to specify expected coverage of the xGATEWAY. The first "Simple Radius", in which a radius defines a perfect circle area around the xGATEWAY. The second "Complex Radii", in which north offset, and 4 quadrant radii, are given (see the diagram below).



xTAGs


admin@testclient.com

- Overview
- Sites
- xGATEWAYS
- xTAGs**
- Reports
- Admin

Home > xTAGs

xTAGs

Clear

+ Register xTAG
↻

0 xTAGs Selected

<input type="checkbox"/>	Device ID ^	Nearest to	Heartbeat	Lat, Lon	RSSI	Voltage	Type	Connection	Command	Last Edited	Actions
<input type="checkbox"/>	20:12:52:01:00:02	Comox xGw 1	2023-08-01, 11:06:41 a.m.	49.6749, -124.89336	-17 dBm	3.0 VDC	Gas-CO2	HaLow	2023-08-01, 10:56:42 a.m. !	2023-08-01, 10:51:48 a.m.	Edit
<input type="checkbox"/>	20:12:52:01:00:08	Comox xGw 1	2023-08-01, 10:56:36 a.m.	49.6749, -124.89336	-15 dBm	3.0 VDC	Env TPH	HaLow	2023-08-01, 10:56:39 a.m. ✓	2023-08-01, 10:52:04 a.m.	Edit

The xTAG management page is similar to the xGATEWAY management page. Clicking on the first column of a row (device ID) will show more details about the selected xGATEWAY. Clicking on the Lat/Lon coordinates will reveal the location of the coordinates on a Google map.

When one or more xTAGs are selected, a configuration or firmware update can be issued via the appropriate button above the table on the left, similar to the xGATEWAYS page.

The command status will show a yellow icon when a command has been dispatched and no confirmation has yet been received. The status will show a green icon if a successful confirmation has been received, and red if an error confirmation has been received.

Removing an xTAG will remove all associated heartbeats, environmental records, shock events, etc.

To edit an xTAG, use a table row's edit button. The available fields of the form differ depending on the sensor type selected. Three example Edit forms are shown below for 3 sensor types.

Vibration Sensor (Accelerometer) Edit Form

Edit xTAG ×

Device ID

Sensor Type

Acc/Pos

Transmit Power (dBm)

15

Heartbeat Frequency (minutes)

10

Heartbeat Retries

5

Shock Threshold (G)

0

Apply Changes

Enter the following information:

- **Device ID:** The device ID (MAC address) of the device (read only unless registering the xTAG).
- **Sensor Type:** The xTAG sensor type.
- **Transmit Power:** The transmit power in dBm. Leave this value at 15 dbm (default) unless very long range of 200m or more is required - then select 18 dbm.
- **Heartbeat Frequency:** The heartbeat frequency in minutes. This is the frequency at which xTAGs wake from a sleep, connect to a nearby xGATEWAY, get configuration updates and upload previously sampled data.
- **Heartbeat Retries:** Leave the default value - unless optimizing xTAG power (then reduce to 2 retries).
- **Shock threshold:** If triggering an opportunistic short 5 sec acquisition on the xTAG, select the G force threshold breach required to trigger those acquisitions.

Environmental Sensor Edit Form

Edit xTAG
 ×

Device ID

Sensor Type

Transmit Power (dBm)

Heartbeat Frequency (minutes)

Heartbeat Retries

Shock Threshold (G)

Sample Frequency (seconds)

Env Sample IIR Filter

Temperature Oversample Rate

Pressure Oversample Rate

Humidity Oversample Rate

Apply Changes

Enter the following information:

- **Device ID:** The device ID (MAC address) of the device (read only unless registering the xTAG).
- **Sensor Type:** The xTAG sensor type. Note that Environmental sensors measuring Temperature, Pressure and Humidity (TPH) and those just measuring Temperature and Humidity (TH) are supported by the same form (Pressure Oversample Rate is ignored for TH style sensors).
- **Transmit Power:** The transmit power in dBm. Leave this value at 15 dbm (default) unless very long range of 200m or more is required - then select 18 dbm.
- **Heartbeat Frequency:** The heartbeat frequency in minutes. This is the frequency at which xTAGs wake from a sleep, connect to a nearby xGATEWAY, get configuration updates and upload previously sampled data.
- **Heartbeat Retries:** Leave the default value - unless optimizing xTAG power (then reduce to 2 retries).
- **Shock threshold:** If triggering a shock event - select a threshold value in G's. If the sensor is shocked greater than this force, a shock event will be created.
- **Sample Frequency:** Sample recording frequency in seconds. How often the sensor records a sample.
- **Env Sample IIR Filter:** TPH sensors can filter incoming data using an IIR filter. Given how slow temperature, pressure and humidity values change, this value should not be used (as typical). Set it to 0 unless measuring a very quickly changing temperature or pressure.

- **Temperature Oversample Rate:** Temperature oversampling is supported, but should never be used. Deviceworx recommends leaving this rate at 1x (1).
- **Pressure Oversample Rate (Environmental only):** Minimal pressure oversampling is helpful and Deviceworx recommends oversampling at a rate of 2x (2). This is based on empirical test results.
- **Humidity Oversample Rate (Environmental only):** Nominal humidity oversampling greatly improves humidity measurements. Deviceworx recommends oversampling at a rate of 4x (4).

Gas (CO2) Sensor Edit Form

Edit xTAG
 ×

Device ID

Sensor Type

Transmit Power (dBm)

Heartbeat Frequency (minutes)

Heartbeat Retries

Shock Threshold (G)

Sample Frequency (seconds)

CO2 Altitude (m)

CO2 Pressure (kPa)

CO2 Temperature offset (°C)

FRC (next config update)

FRC Measured PPM

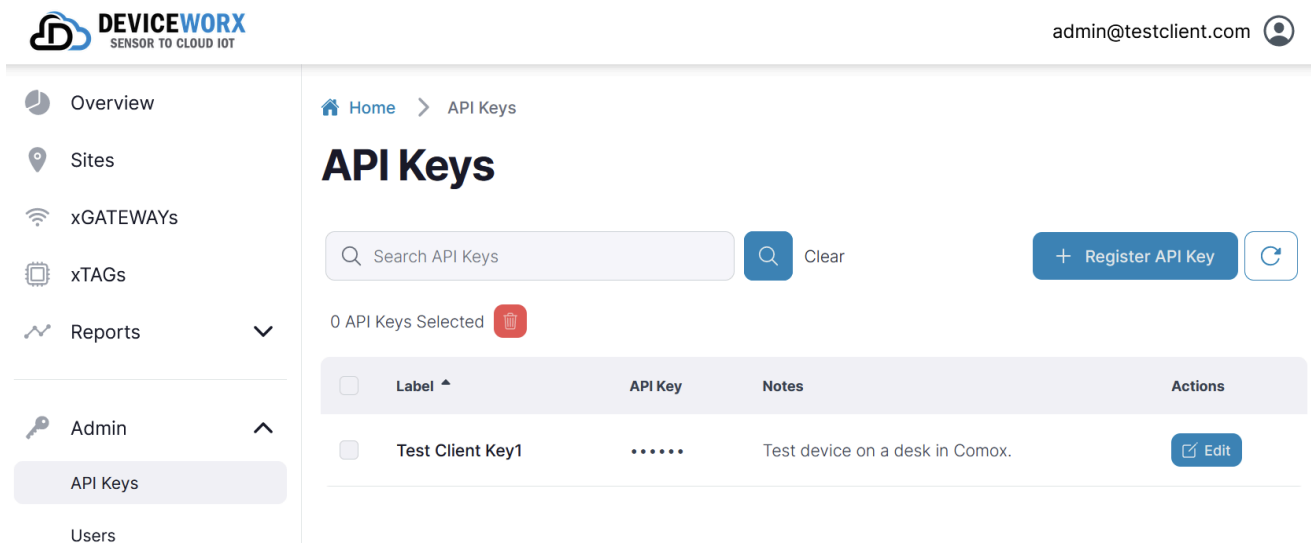
Apply Changes

Enter the following information:

- **Device ID:** The device ID (MAC address) of the device (read only unless registering the xTAG).
- **Sensor Type:** The xTAG sensor type. Note that Environmental sensors measuring Temperature, Pressure and Humidity (TPH) and those just measuring Temperature and Humidity (TH) are supported by the same form (Pressure Oversample Rate is ignored for TH style sensors).
- **Transmit Power:** The transmit power in dBm. Leave this value at 15 dbm (default) unless very long range of 200m or more is required - then select 18 dbm.
- **Heartbeat Frequency:** The heartbeat frequency in minutes. This is the frequency at which xTAGs wake from a sleep, connect to a nearby xGATEWAY, get configuration updates and upload previously sampled data.
- **Heartbeat Retries:** Leave the default value - unless optimizing xTAG power (then reduce to 2 retries).

- **Shock threshold:** If triggering a shock event - select a threshold value in G's. If the sensor is shocked greater than this force, a shock event will be created.
- **Sample Frequency:** Sample recording frequency in seconds. How often the sensor records a sample.
- **CO2 Altitude (Gas-CO2 only):** CO2 sensor accuracy can be affected by atmospheric pressure as it relates to altitude. Set this value to an altitude (in meters) above sea level.
- **CO2 Pressure (Gas-CO2 only):** For cloud-connected sensors, this value is not often used. Set it (in lieu of an altitude) to get an even more accurate measurement.
- **CO2 Temperature offset (Gas-CO2 only):** This value directly offsets the measured temperature from the sensor. It is provided - given that CO2 temperature values are not very accurate. Ignore this value noting that it does not affect CO2 accuracy.
- **FRC (Gas-CO2 only):** This value supports future cloud-based calibration of sensors. Note that Deviceworx pre-calibrates CO2 sensors before they ship, so no calibration should be required.
- **FRC Measured PPM (Gas-CO2 only):** This value supports entering a known CO2 value for calibration. As calibration is not currently supported, this field can be ignored.

API Keys



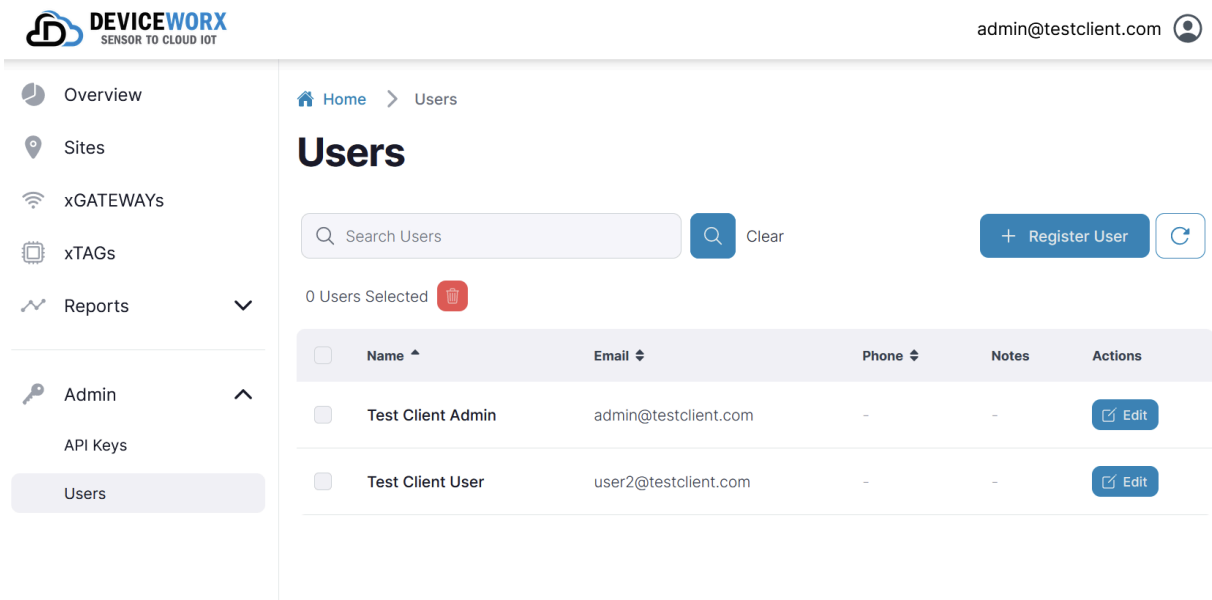
The screenshot shows the DeviceWorx web interface. The top navigation bar includes the DeviceWorx logo and the user email 'admin@testclient.com'. The left sidebar contains a menu with 'API Keys' highlighted. The main content area is titled 'API Keys' and features a search bar, a 'Register API Key' button, and a table of existing API keys. The table has columns for 'Label', 'API Key', 'Notes', and 'Actions'. One key is listed: 'Test Client Key1' with a masked API key and a note 'Test device on a desk in Comox.'.

The API Keys page can be accessed by Admins and is used to register, modify, and delete API keys. API keys have a specified access level (role) and must be used to authenticate API calls.

For security reasons, API keys are shown to the user once upon creation and can not be shown again once the dialog is closed. For API keys that have already been registered, only the label and description can be changed, not the key itself.

Critically - API Keys will be preset by Deviceworx for customers when their account is setup within the Dashboard. Deviceworx does not expect that customers will need to utilize API Keys screens. Their access is provided for completeness.

Users



The screenshot shows the 'Users' management interface in the DeviceWorx application. On the left is a sidebar with navigation links: Overview, Sites, xGATEWAYS, xTAGs, Reports, Admin, API Keys, and Users (highlighted). The top right shows the user 'admin@testclient.com' with a profile icon. The main content area has a breadcrumb 'Home > Users' and a title 'Users'. Below the title is a search bar labeled 'Search Users' with a 'Clear' button. To the right of the search bar are two buttons: '+ Register User' and a circular refresh icon. Below these is a status bar indicating '0 Users Selected' with a trash icon. A table lists the users with columns for Name, Email, Phone, Notes, and Actions. Two users are listed: 'Test Client Admin' and 'Test Client User', both with 'Edit' buttons in the Actions column.

<input type="checkbox"/>	Name ^	Email ^	Phone ^	Notes	Actions
<input type="checkbox"/>	Test Client Admin	admin@testclient.com	-	-	<button>Edit</button>
<input type="checkbox"/>	Test Client User	user2@testclient.com	-	-	<button>Edit</button>

The user management page allows admins to register, modify, and remove users, as well as change the password of existing users without the need to provide their existing password. Newly registered users will be created with the password provided in the user registration form, and they should be advised to change their password as soon as possible.

New users can be created with the “Register User” button.

Register User ×

Name

Email

Password

Confirm Password

Password must contain at least 8 characters, a mix of upper and lower case and at least one symbol or number.

Role

Select Role... ▼

Contact Phone (optional)

Notes (optional)

Register

Enter the following information for each user.

- **Name:** The user’s name.
- **Email:** The user’s email address.
- **Password:** A temporary password the user will initially use to sign-in.
- **Confirm Password:** The confirmation of the temporary password.
- **Role:** The role of the new user.
- **Contact Phone:** The user’s phone number for contact.
- **Notes (optional):** Optional notes pertaining to the user.

Existing users can also be modified with the “edit” button in their corresponding table row. The form is similar, with an optional password change option as shown below:

Edit User ×

Edit User Details

Name

Email

Role

User ▼

Contact Phone (optional)

Notes (optional)

Apply Changes

Change User Password

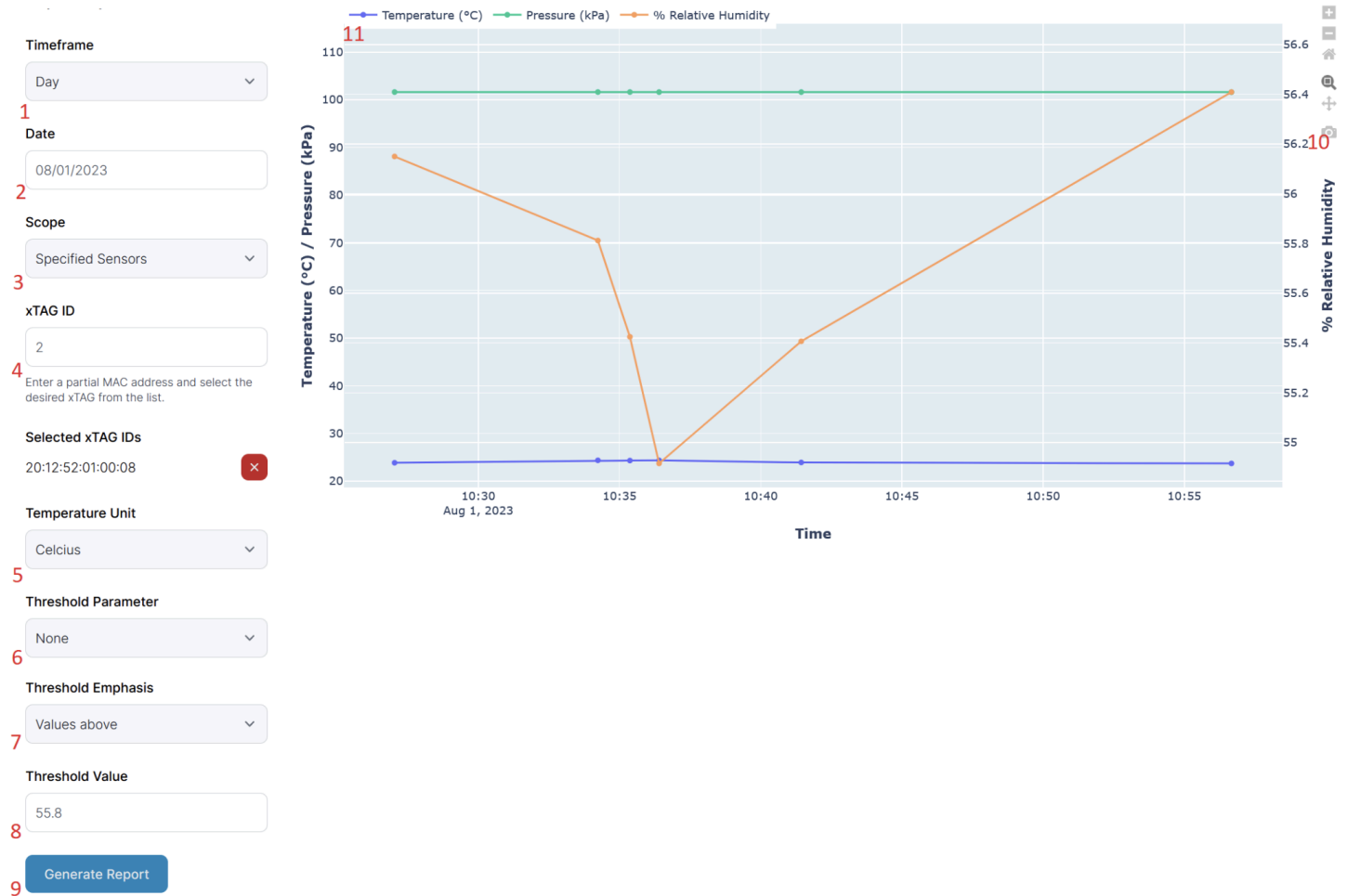
Password
👁

Confirm Password
👁

Password must contain at least 8 characters, a mix of upper and lower case and at least one symbol or number.

Change Password

Reports



The reports interface generates sensor record data reports.

Timeframe & Date

In the Timeframe drop-down (1), day, month, year, or custom range can be specified. Use the Date picker (2) to set the date, month, year, or range for the report.

Scope

The Scope drop-down **(3)** specifies whether data is shown for all sensors, data acquired at a specific site, or individual sensors. For individual sensors, all data is plotted for each sensor. For “all sensors” or “specific site”, data is displayed as hourly/daily/monthly averages depending on the length of the timeframe.

Temperature Unit

The temperature unit drop-down **(5)** can be used to specify whether temperature values are shown in degrees Celsius or degrees Fahrenheit, where applicable.

Threshold

The Threshold Parameter **(6)** drop-down allows a report parameter to be selected to be emphasized (highlighted in red). Whether the emphasis is on values greater than or less than the parameter is controlled by the emphasis drop-down **(7)**. The min/max value is set in the Threshold Value input **(8)**.

Plot Controls

The plot toolbar **(10)** includes controls to (top to bottom) increase/decrease zoom, reset zoom, use the mouse to select an area to zoom, pan the graph area, and export a PNG image of the plot.

The legend **(11)** can be used to show/hide plot data lines by clicking on the line color you wish to hide from the graph. The axes scale may be adjusted to fit the new bounds.

xGATEWAY Custom App Dev Support

The xGATEWAY device is intended to be “open” and support execution of customer applications or apps. To facilitate this, customers can develop their apps on xGATEWAYS using standard dev tools.

Support for some app dev is pre-installed by default. For example, test Node.js (Javascript support and Node.js install) by executing “node -v” when connected to an xGATEWAY through SSH to read the version of Node.js installed. Or, confirm python support by issuing “python -v”.

A wide variety of tools and documentation is available online supporting JS and Python development and will not be included within this doc.

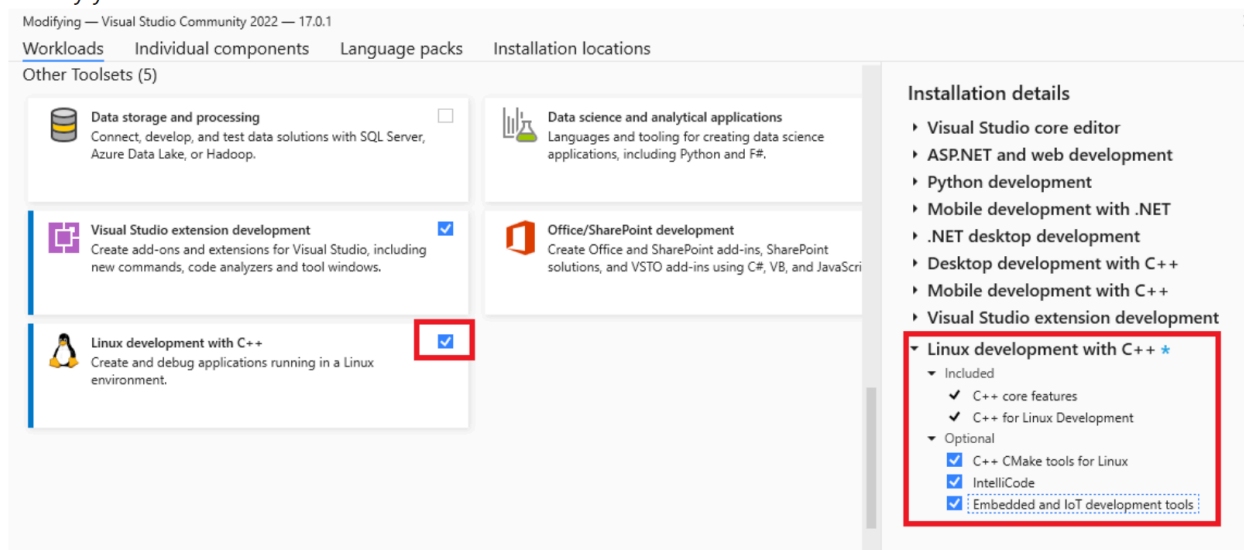
To develop “native” C/C++, Visual Studio Community (or any VS version compatible with VS Community) is the best option. See [this link](#) for VS Community details. Deviceworx can assist in pre-installation of the following elements supporting VS Community:

- rsync - supports host to xGATEWAY target comms.
- gdb - supports on target debugging with breakpoints, etc.
- zip - a prerequisite for VS Community target agent software.

... simply request that Deviceworx sales (sales@deviceworx.com) provide VS Community support on xGATEWAY devices when they are purchased and the required VS Community elements will be pre-installed.

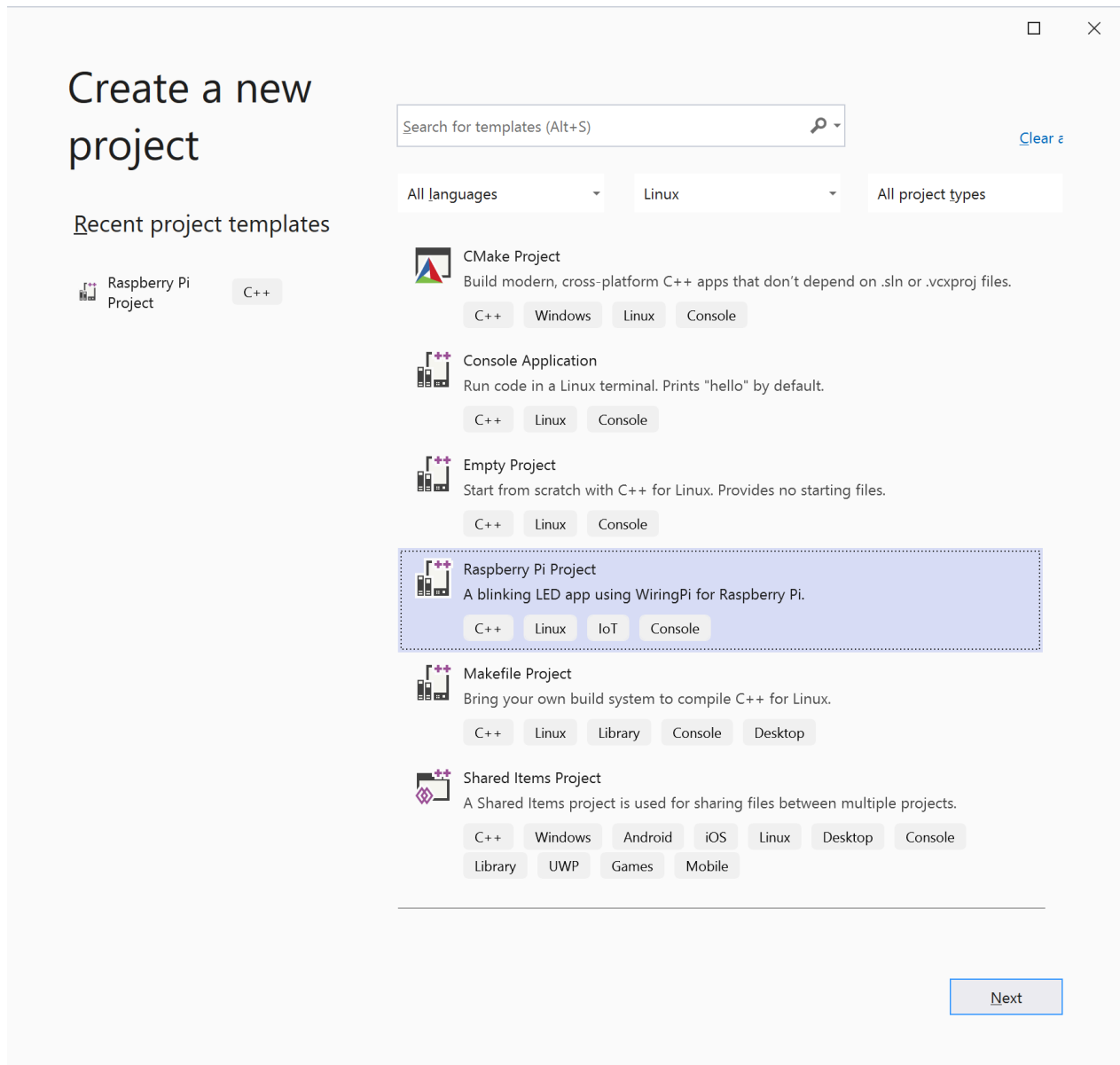
Follow VS Community installation instructions and during installation, ensure that support for Linux target devices like the Raspberry Pi is selected. Specifically - look for “Linux development with C++”. Or, modify an existing installation to include Linux target support (see example screenshot below).

Modify your Visual Studio.

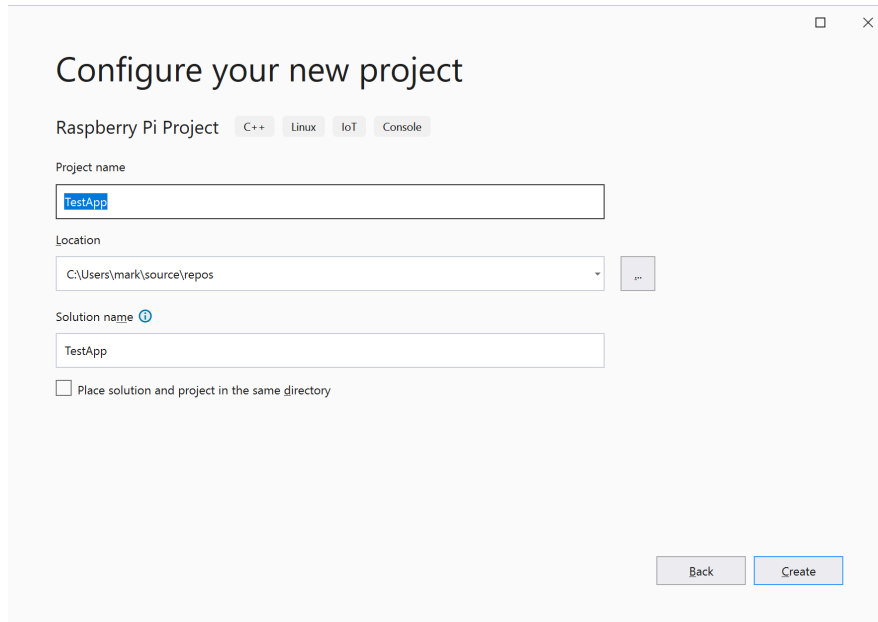


Once VS Community has been setup with “Linux development with C++”, a project can be created to support app dev using the steps below.

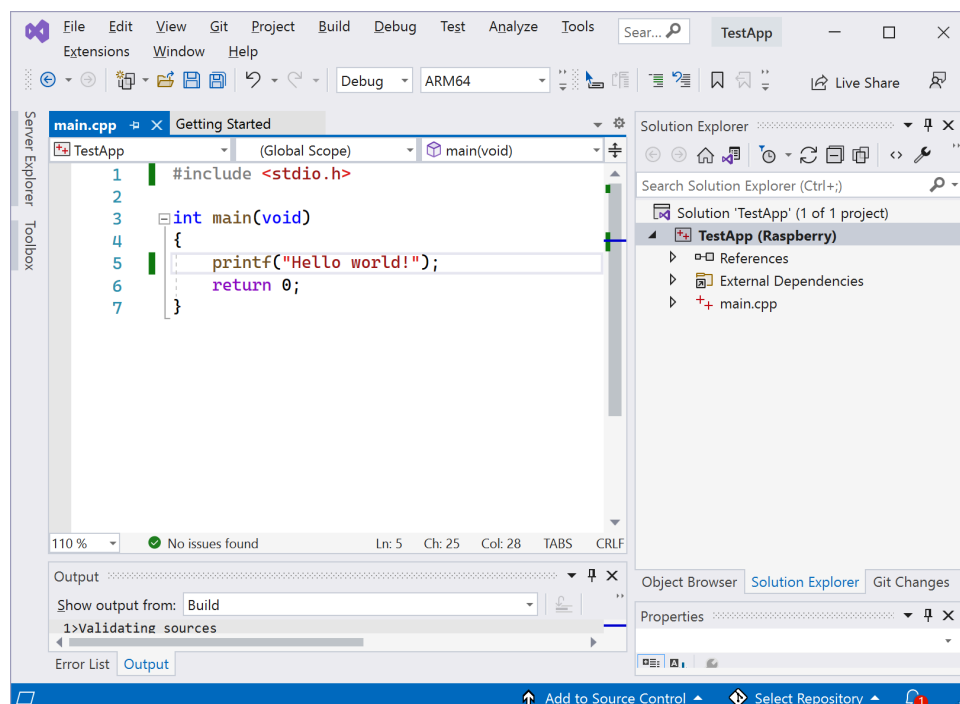
1. Within VS Community, select *File->New->Project* and a dialog supporting creation of a project will be displayed as shown below. Select the *Raspberry Pi Project* and then *Next*.



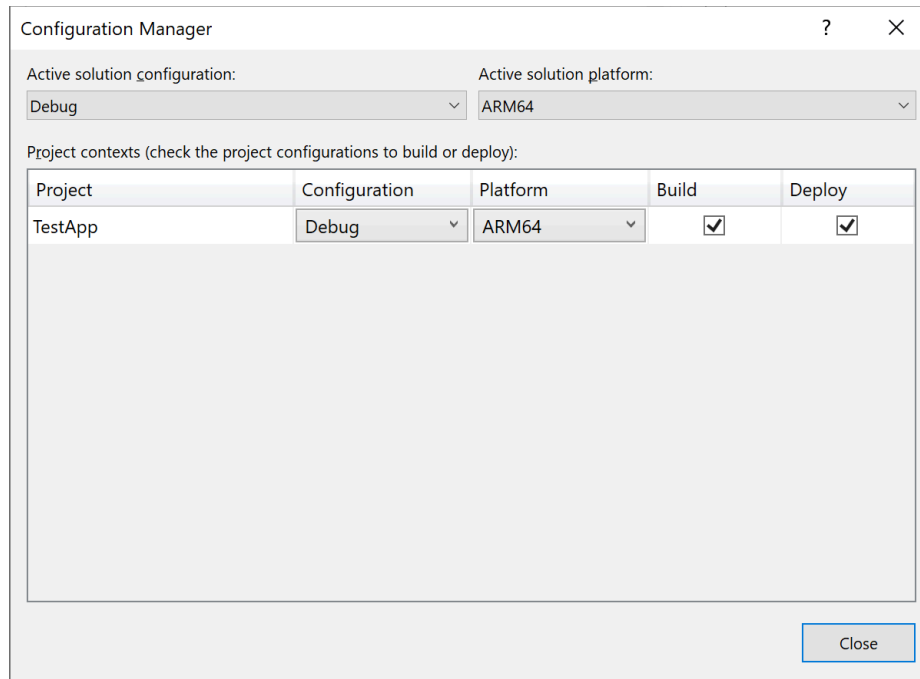
2. Give the app a name and then select *Create* as shown below.



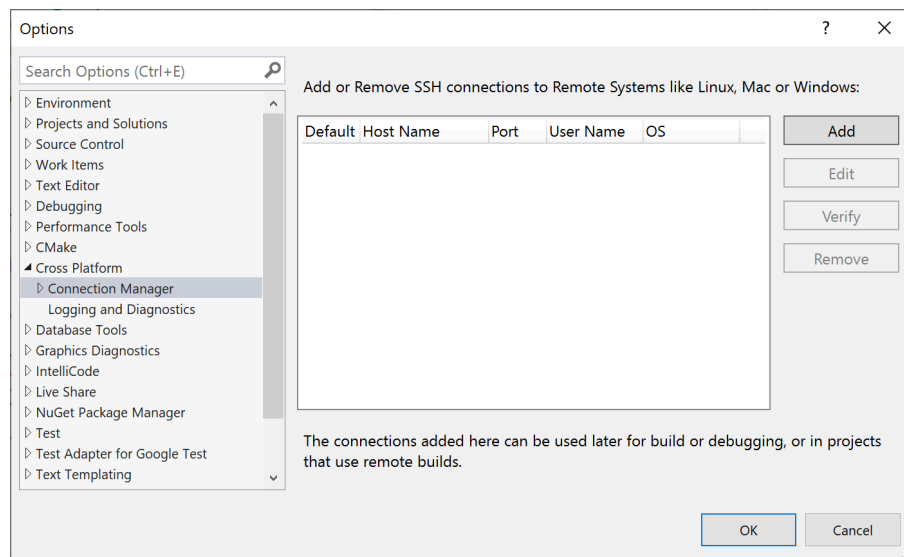
3. A new project will be created with some Raspberry Pi test code. Update this code as shown below (simple Hello World output). Note that the target *ARM* will also have to be changed to *ARM64*.



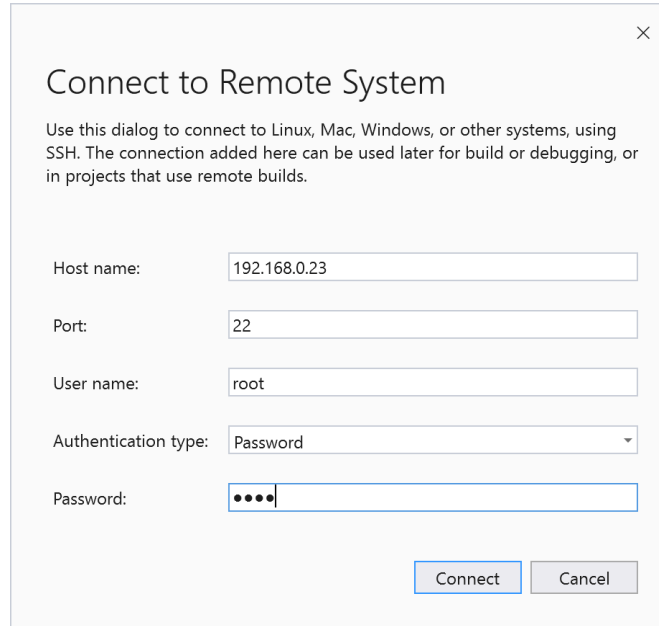
4. Select the drop down configuration option (down triangle to the right of “ARM”) to open the *Configuration Manager* as shown below. Select an *Active Solution platform* of ARM64 for both *Debug* and *Release* configurations before selecting the *Close* button.



5. Next, the *Connection Manager* will have to be configured to point at the target Linux device. Select the *Tools->Options* menu item and look for the *Cross Platform -> Connection Manager* branch on the left side of the *Options* dialog box as shown below.



6. Click *Add* to Add the xGATEWAY target.
7. Enter the IP address of the xGATEWAY. Leave the *Port* and *Authentication Type* at default values of 22 and *Password* respectively and enter the value of *root* for both *User name* and *Password*. Press ok to connect to the xGATEWAY. This may take a few seconds.



Connect to Remote System

Use this dialog to connect to Linux, Mac, Windows, or other systems, using SSH. The connection added here can be used later for build or debugging, or in projects that use remote builds.

Host name:

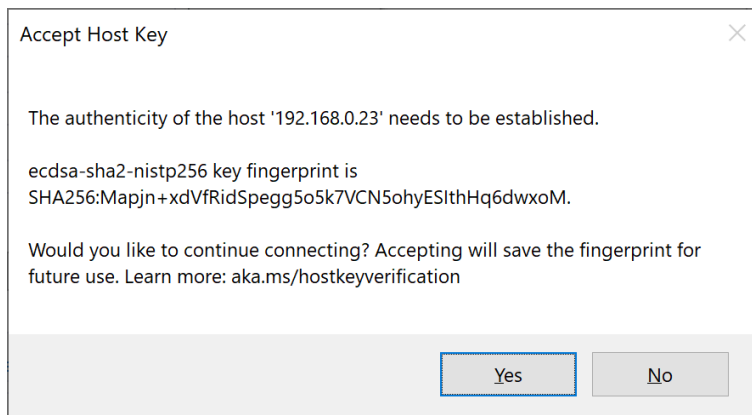
Port:

User name:

Authentication type:

Password:

8. VS Community will ask you to *Accept a Host Key* so that reconnections are automatic. Select *Yes*. See below



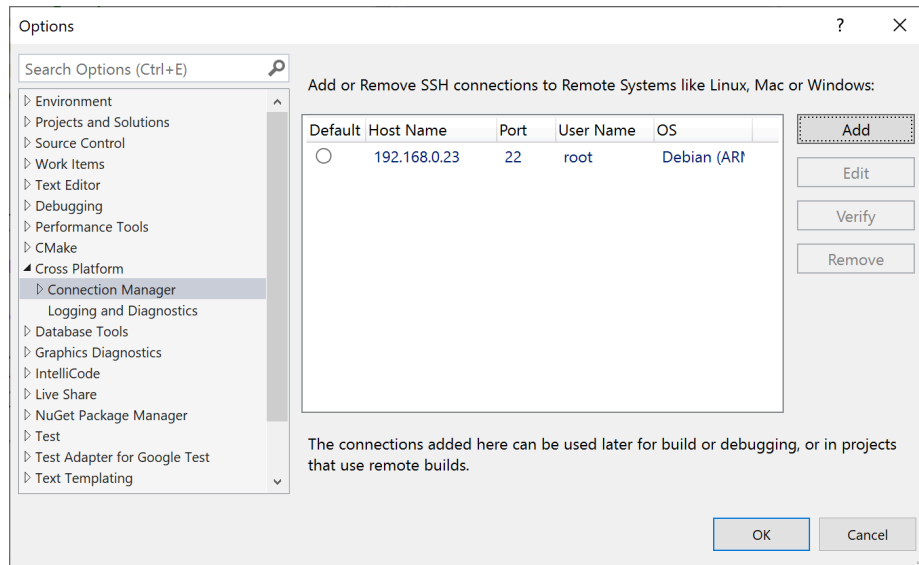
Accept Host Key

The authenticity of the host '192.168.0.23' needs to be established.

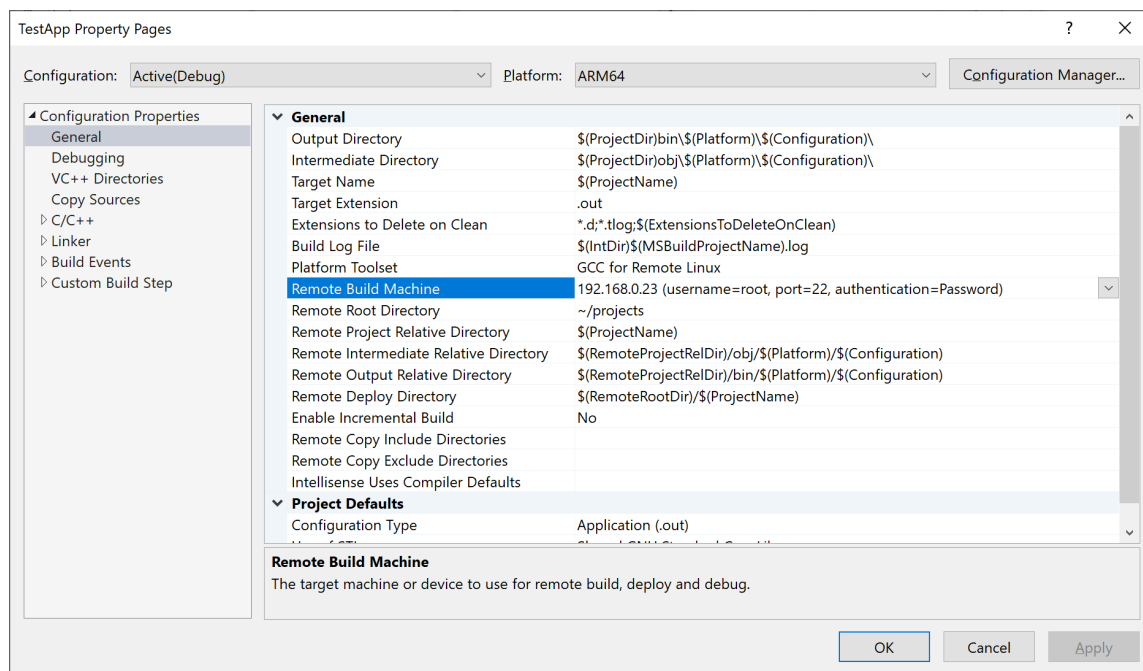
ecdsa-sha2-nistp256 key fingerprint is
SHA256:Mapjn+xdVfRidSpegg5o5k7VCN5ohyESlthHq6dwxoM.

Would you like to continue connecting? Accepting will save the fingerprint for future use. Learn more: aka.ms/hostkeyverification

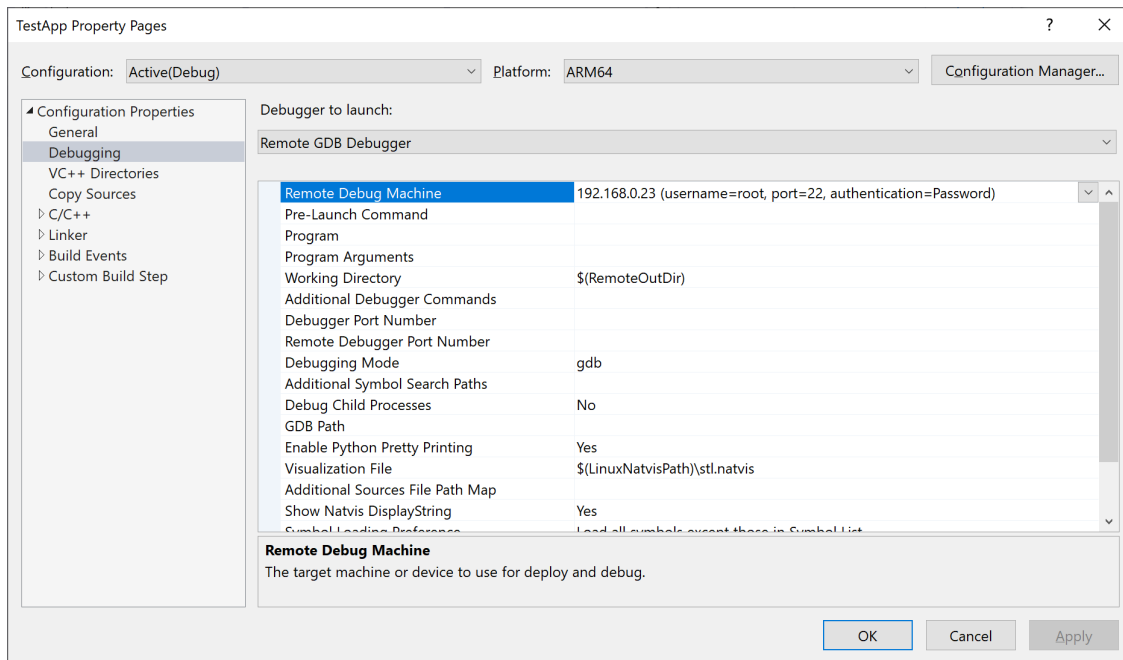
9. The connection (shown below) is now ready for use after selection within the project. Note that the above step, importantly, transfers VS Community support elements to the target.



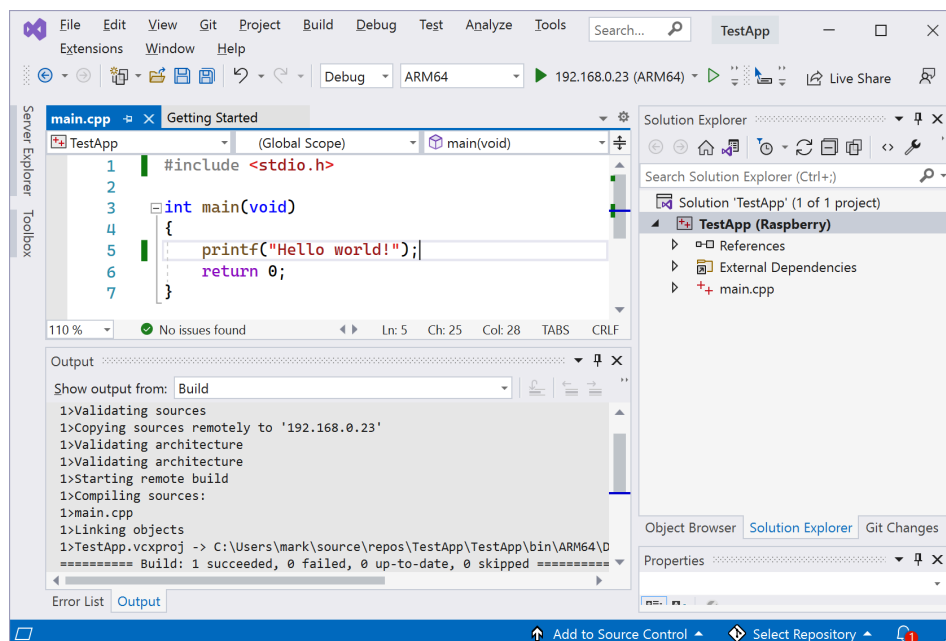
10. Right click on the project and select *Properties* to open the <Project> Property Pages. Select the *Remote Build Platform* option matching the target IP as shown below.



11. Make the same selection within the *Debugging - Remote Debug Machine* selection as shown below.



12. Select the *Build -> Build Solution* menu option to build the app on the xGATEWAY target. VS Community output should be similar to what is shown below (see Output window in the lower left corner of the screen).



13. Select the *Debug -> Start Debugging* menu option to run the app on the target debugger. It will run within a console window and terminate very quickly. Add code, if required, to pause execution after the `printf()` call to witness this console for longer.

.NET Framework Support

Follow steps below to download and use .NET 5.0. Newer versions of .NET should be installable using the same or very similar steps.

1. Download the ARM64 SDK (inc runtime) to /root/ (\$HOME) using sftp. File: `dotnet-sdk-5.0.100-linux-arm64.tar.gz`
2. Extract it:

- a. `mkdir -p "$HOME/dotnet"`
- b. `tar xzf dotnet-sdk-5.0.101-linux-arm64.tar.gz -C "$HOME/dotnet"`

3. Setup paths by adding the following lines to `~/.bashrc` using emacs or vi:

```
# .net additions below
export DOTNET_ROOT=$HOME/dotnet
export PATH=$PATH:$HOME/dotnet
```

4. Restart the xGATEWAY to set the environment variables set within the `~/.bashrc` file. Confirm `PATH` and `DOTNET_ROOT` env vars are set by issuing: `env`
5. Issue `info` cmd to confirm versions, etc.: `dotnet --info`

```
.NET SDK (reflecting any global.json):
Version: 5.0.100
Commit: 5044b93829
Runtime Environment:
OS Name: debian
OS Version: 10
OS Platform: Linux
RID: debian.10-arm64
Base Path: /root/dotnet/sdk/5.0.100/
Host (useful for support):
Version: 5.0.0
Commit: cf258a14b7
.NET SDKs installed:
5.0.100 [/root/dotnet/sdk]
.NET runtimes installed:
Microsoft.AspNetCore.App 5.0.0 [/root/dotnet/shared/Microsoft.AspNetCore.App]
Microsoft.NETCore.App 5.0.0 [/root/dotnet/shared/Microsoft.NETCore.App]
```

6. Create a simple .NET console app that runs to prove .NET installation is ok. Execute the following commands:

- a. *mkdir ~/dotnet/dotnet5test*
- b. *cd ~/dotnet/dotnet5test*
- c. *dotnet new console*
- d. *dotnet run*

... and see the "Hello World!" output. A project (*~/dotnet/dotnet5test/dotnet5test.csproj*) is created from a "console" template. This includes a single c# file (*~/dotnet/dotnet5test/Program.cs*) that contains code.

This is a simplistic test. For more elaborate apps, more config, libs, etc are typically required.

xGATEWAY Socket Interface

A local (same subnet or device) Socket Interface is supported by xTAG Daemon apps running on xGATEWAY devices. Two xTAG daemons are available for and pre-installed on the xGATEWAY. The first (“xtagusb”d”) supports interacting with USB-connected xTAGs. The second, xtaghlod supports interacting with WiFi HaLow connected xTAGs. Daemon client connections will be via standard Berkeley Socket techniques and will not be described within this document. Following socket connections, daemon clients will send commands to the daemons and handle responses.

Within this document, “xtagxxd” is used to describe functionality associated with all daemons.

Note that all daemons can be run simultaneously as they use different ports.

For all daemon types, 2 socket parts are used. A primary port handles incoming synchronous client commands with responses to these commands sent back to the client on the same port. A secondary stream port sends asynchronous messages to clients. For example, start data streaming by issuing a command over the primary port and continuously read streamed data messages over the stream port (until a stop streaming command is issued on the primary port). To preserve the possibility of supporting multiple daemons running concurrently, ports are different for each daemon type (listed below). See [Accelerometer Acquisition Stream Data \(Responses Only\) \(0x2A\)](#) for more detail on streaming and an example. Daemon clients must connect to a Primary port before issuing any synchronous commands and must connect to a Stream port before data streaming is started. Note that if no streaming is required, only a connection to a Primary port is required.

Daemon	Primary Port	Secondary Streaming Port
xtagusb	3242	3243
xtaghlod	3244	3245

It is strongly recommended that xtagxxd clients disconnect all sockets before restarting xtagxxd or reconnecting to it (if ever required). Failure to release socket connections can require a socket timeout (approx 1 min) before re-binding to these sockets is allowed (handled within xtagxxd)..

To serve as an example to client app developers, Devicworx has created an xtagxxd client test app (“xGwTestApp”). This app can be referenced at any time as a public GitHub repo:<https://github.com/deviceworx/xGwTestApp/tree/main>. There is a readme.md file in this repo with details on the app including its structure.

xtagusbd and xtaghlod Interface Differences

While Deviceworx xTAG USB and xTAG HaLow Sensors support the same core functionality (including support for a variety of membranes, each with different sensor options), the way that they communicate with xGATEWAYS through xtagxxxd daemons is fundamentally different.

xGATEWAYS commonly connect to xTAG USB Sensors (i.e. a connection is initiated from xGATEWAYS whenever an xTAG USB Sensor configuration change occurs or samples are read).

Conversely, xTAG HaLow Sensors periodically connect to xGATEWAYS. xGATEWAYS never connect to xTAG HaLow Sensors. This is due to high battery drain on xTAG HaLow Sensors whenever they act as a WiFi access point. It is much more power efficient for xTAG HaLow Sensors to periodically wake from a deep sleep, connect to a xGATEWAY, acquire any pending configuration updates, transmit any acquired samples and then go back to sleep.

Because USB and HaLow communications are structurally different, the API supporting their configuration, monitoring and data acquisition has differences. These differences are described within the detailed commands, responses and unsolicited messages described below.

xTAG Sensor Streaming Vs Sampling

Because of data acquisition speed differences, some sensors collect and forward data onto xGATEWAYS differently.

Vibration sensors, for example, sample data as fast as 1600 times per second. Data from Vibration sensors is streamed to xGATEWAYS over USB whenever an xGATEWAY connection to them is made and they are signaled to start acquiring data. When they are signaled to stop acquisition, the data stream stops. Of course incoming stream data is forwarded over the Socket Interface stream port. Vibration sensor data can also stream to xGATEWAYS over HaLow after these sensors wake and connect to the xGATEWAY (i.e. A delayed stream). Importantly, stream data is sent over the Socket Interface asynchronously after stream start.

Environmental sensors, GAS sensors, etc collect sample data much slower, typically every few min or every few hours. When signaled to start sampling, these sensors collect data internally and then periodically send this data to xGATEWAYS when xGATEWAYS connect (via USB) or when they connect to xGATEWAYS (via HaLow). Importantly, xGATEWAYS synchronously read samples from these sensors. Their data is not streamed. Likewise, the Socket Interface supports read commands for each of these sensors - supporting direct read of a sample data cache within xTAGs (connected via USB) or of a cache within the xGATEWAY (for xTAG HaLow sensors that have transferred their sample data opportunistically after each xGATEWAY connection).

xTAG Sensor Calibration

Some xTAG Sensors (e.g. Vibration sensors, CO2 sensors, etc) require calibration after manufacturing to ensure accurate measurement. Deviceworx calibrates all xTAG sensors before they ship to customers and their calibration data is persisted within sensor chipsets.. For that reason, no xTAG Calibration messaging is described in this doc (even though calibration commands exist).

Command Formatting

All commands (cmds) will have the following format for xtagusbd and xtaglmod comms.

<cmd byte>,<cmd len byte>,<cmd param byte 1>..

Notes

1. <cmd len byte> stipulates total cmd length including the <cmd byte> and all bytes following it.
2. <cmd param byte 1>...<cmd param byte n> are optional and will only be used in commands where passing data to xtagxxd is required.

Command Response and Unsolicited Message Formatting

Command responses will have the following format.

<resp/msg byte>,<resp/msg len byte>,<resp error>,<resp/msg data byte 1>...<resp/msg data byte n>

Notes

1. A resp byte is sent in response to a cmd byte and always matches that byte value. (e.g. resp byte value 0x06 is sent at the start of the response to cmd 0x06).
2. A msg byte is sent asynchronously to cmd bytes and does not match any cmd byte. Stream messages, for example, have a byte value that does not match any cmd byte.
3. The len byte stipulates total resp/msg length including the resp/ msg byte and all bytes following it.
4. resp error will pass an error value or status value indicating successful command execution.
5. resp/msg data byte 1...resp/msg data byte n may only be returned for some commands but is not required when a response is simply passing back status via the resp error byte.

Error Responses

Error responses will be consistent for all commands described in this document. Specifically, they will be 3 bytes long and match this format:

<resp byte>,<len byte=0x03>,<Error Byte>

Valid Error Byte values will be described for each command.

xtagusbd Communications

Typical xtagusbd Command Sequences

The following command order of execution applies to xtagxxxd client applications (applicable port in brackets for usb).

1. Connect to the xGATEWAY via sockets.
2. Configure an xGATEWAY (3242) or ...
3. List available xTAGs (3242) ...
 - a. Connect to an xTAG (3242).
 - b. Start acquisition (3242).
 - c. Stream fast data (e.g. from Vibration sensors) ((3243)) or periodically read slow data (e.g. from Environmental sensors) (3242).
 - d. Stop acquisition and disconnect the xTAG (3242).
4. Disconnect sockets.

Each of these steps is demonstrated within the daemon client test app discussed above.

Socket Interface Commands Specific to xtagusbd (USB xTAGs)

Certain commands (and their responses) are only supported by xtagusbd daemon for xTAG USB comms. For example, only these types of xTAGs are connectable, so only xtagusbd daemon support the [xTAG Connect \(0x03\)](#) and [xTAG Disconnect \(0x04\)](#) commands.

List xTAGs (0x02)

Return a list of connected and connectable xTAGs. Response will be quick as device IDs will be stored when they are discovered upon their connection.

Command Format <Cmd=0x02>,<Len Byte = 0x03>,<Timeout Byte>

Response Format<Cmd=0x02>,<Len Byte>,<Error Byte>,<xTAG1 Con Status>,<xTAG1 MS Byte>,<xTAG1 2nd MS Byte>,<xTAG1 3rd MS Byte>,<xTAG1 4th MS Byte>,<xTAG1 5th MS Byte>,<xTAG1 LS Byte><xTAGn Con Status>,<xTAGn 6 byte address>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length.
- 0x02: Error processing the command.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Value of success will be returned - even if no xTAGs could be found (i.e. none connected on USB).
3. Regardless of physical layer connection type, xTAG IDs will be in the form of MAC addresses with 6 bytes (e.g. 11:22:33:44:55:66 represented in Hex data format).
4. The< xTAGn Con Status> byte will be non-zero when connected and zero when disconnected.
5. Max returned xTAG IDs is 20 (USB) requiring a response len of $(20*(6+1))+3=143$ bytes.
6. <Timeout Byte> valid range is 5-20 sec with default of 10 sec if a value outside of this range is specified (e.g. including 0x00). When a physical USB connection is used, this value is ignored but must be set (cmd len must always be 3 bytes for consistency).

xTAG Connect (0x03)

Connect to a specified xTAG. Multiple connections are supported at once.

Command Format <Cmd=0x03>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x03>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x02: Connection failure. USB connect should be immediate.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. USB will take less than 1 sec.
3. If an xTAG is already connected, calling this does nothing (no reconnect) and returns "Success" quickly.
4. The response does not include the 6-byte xTAG Address. It should not be required when this command is executed sequentially for multiple tags (i.e. wait for a response from each connect command before issuing a connect command for another xTAG).

xTAG Disconnect (0x04)

Disconnect from a specified xTAG.

Command Format <Cmd=0x04>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x04>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Disconnect time is dependent on Physical Link. For USB, this will be < 1 sec.
3. If the xTAG was already disconnected, this command will do nothing and Success will be returned.

4. The response does not include the 6-byte xTAG Address. It should not be required when this command is executed sequentially for multiple tags (i.e. wait for a response from each disconnect command before issuing a disconnect command for another xTAG).

xTAG Metadata Read (0x05)

Read attributes of a connected xTAG.

Command Format <Cmd=0x05>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x05>,<Len Byte = 0x12>,<Error Byte>,<Type Byte>,<Battery Byte>,<SUP Byte>,<RunSec MS Byte>,<RunSec 2nd MS Byte>,<RunSec 3rd MS Byte>,<RunSec LS Byte>,<UnixSec MS Byte>,<UnixSec 2nd MS Byte>,<UnixSec 3rd MS Byte>,<UnixSec LS Byte>,<HW Rev MS Byte>,<HW Rev LS Byte>,<SW Rev MS Byte>,<SW Rev LS Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. xTAG 6 byte address will be returned when error byte is Success or any error. Nothing will follow the xTAG address on error.
3. Type Byte values: 16-Accelerometer, 17-Beacon, 18-Enviromental(TPH), 19-Enviromental(TH), 32-CO2
4. Battery Byte: 0x00-0xFF denoting range 2.2 Vdc to 3.0 Vdc. Always 0xFF for USB-connected xTAGs.
5. SUP Byte is unused.
6. RunSec Bytes make up an unsigned double-word int (UIN32) denoting the number of sec since xTAG firmware start. Can be used for remote troubleshooting as xTAG restarts only occur upon xTAG firmware failures.
7. UnixSec Bytes make up an unsigned double-word int (UIN32) denoting the number of sec since midnight on Jan 1, 1970 GMT (as per Unix timestamp standard). The xTAG does not store this value between boot cycles, so if an xTAG is restarted (for any reason) a client with a valid clock must set the xTAGs Unix time if this time is to be used (i.e. future timestamping). If unix time is not set, these read values won't be valid. IMPORTANTLY - the UnixSec is only updated every sec from an internal processor timer that will be minimally aperiodic. For this reason, expect that the unix time will have to be updated daily to remain within +/- 1-2 sec accuracy.
8. HW Bytes denote a raw unsigned word with implied decimals indicating xTAG hardware revision. E.g. 0x2711 = 10001 denoting rev 1.00.01.
9. SW Bytes denote a raw unsigned word with implied decimals indicating xTAG software or firmware revision. E.g. 0x2775 = 10101 denoting rev 1.01.01.

xTAG Metadata Write(0x06)

Write attributes of a connected xTAG. In effect, this is really used to update the Unix time within sensors so that they can timestamp data.

Command Format <Cmd=0x06>,<Len Byte = 0x0D><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<SUP Byte>,<UnixSec MS Byte>,<UnixSec 2nd MS Byte>,<UnixSec 3rd MS Byte>,<UnixSec LS Byte>

Response Format<Cmd=0x06>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad length.
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. SUP Byte - Unused for USB-connected xTAGs (will be ignored).
3. UnixSec Bytes make up an unsigned double-word int (UINT32) denoting the number of sec since midnight on Jan 1, 1970 GMT (as per Unix timestamp standard). The xTAG does not store this value between boot cycles, so if an xTAG is restarted (for any reason) a client with a valid clock must set the xTAGs Unix time if this time is to be used (i.e. timestamping). If unix time is not set, these read values won't be valid. IMPORTANTLY - the UnixSec is only updated every sec from an internal processor timer that will be minimally aperiodic. For this reason, expect that the unix time will have to be updated daily to remain within +/- 1-2 sec accuracy.

HaLow Communications

Typical xtaghlod Command Sequences

The following command order of execution applies to xtaghlod client applications (applicable port in brackets).

1. Connect to the xGATEWAY via sockets (3244/3245).
2. Configure an xGATEWAY (3244) or ...
3. Configure an xTAG (3244) ...
 - a. Signal acquisition to start whenever the next xTAG connection to the xGATEWAY is made (3244).
 - b. Stream fast data (e.g. from Vibration sensors) or periodically read slow data whenever xTAGs connect (e.g. from Environmental sensors) (3245).
 - c. Signal acquisition stop upon the next xTAG connection (3244).
4. Disconnect sockets (3244/3245).

HaLow Config and Sample Data Store and Forward

Critically - xTAG HaLow Sensor interactions with xGATEWAYS are completely different from USB sensor interactions in that xTAG HaLow sensors connect to xGATEWAYS. **xGATEWAYS never connect to xTAG HaLow Sensors**. More typically - IoT Gateways connect to sensors and not the other way around. xGATEWAY to xTAG USB Sensor connections, for example, follow this more typical paradigm (xGATEWAYS connect to xTAG USB sensors). The reason HaLow is different - HaLow sensors cannot function as connectable WiFi Access Points without drawing a lot of battery power. This is a limitation of WiFi HaLow. To address this, xGATEWAYS store config updates for xTAG HaLow sensors so that they can fetch these updates when they next wake up from a low-power sleep state and connect. Importantly - if any xTAG HaLow Sensor has the opportunity to connect to multiple xGATEWAYS, each must store a pending config update for that xTAG HaLow Sensor. Because xGATEWAYS support a massive, local, secure persistent database (over 10 GB possible size), there is no practical limitation to this approach. Socket Interface client applications must ensure that all xGATEWAYS near an xTAG HaLow Sensor receive any pending configuration updates.

Whenever an xTAG to xGATEWAY connection is made, the following steps occur.

1. xTAG wakes from a sleep state.
2. The xTAG scans for nearby xGATEWAYS (each acting as a WiFi HaLow AP) over WiFi HaLow.
3. The xTAG connects to the xGATEWAY with the highest scanned RSSI.
4. The xTAG sends a [xTAG HaLow Metadata Heartbeat \(0x0A\)](#) message to the xGATEWAY outlining its current time, config data timestamp, battery level, last RSSI scan ...
5. If the timestamp for the xTAG's config data within the xGATEWAY is newer than the timestamp in the heartbeat, it sends a config update. This process will also start sensor sampling if a sample rate changes from 0 to non-zero. If the sample rate transitions to 0, sampling is stopped.
6. If a stream start is signaled on the xTAG (e.g. vibration sample stream), the stream is started and runs for a few seconds - passing sample data in real time up to the xGATEWAY.

7. Any slowly acquired sample data (e.g. environmental samples taken every few min or hours) is forwarded to the xGATEWAY.
8. If a FOTA update for the xTAG has been saved in the xGATEWAY, a FOTA update is performed.
9. If the xTAG sensor current time is inaccurate, an xTAG clock update occurs.
10. The xTAG disconnects and goes back to sleep.
11. The xTAG may wake up multiple times before the next xGATEWAY connection to sample data at the rate provided (e.g. Environmental or GAS sensors).

Socket Interface Commands Specific to xtaglmod (HaLow xTAGs)

Certain commands (and their responses) and asynchronous messages are only supported by the xtaglmod daemon for xTAG HaLow comms. For example, only these types of xTAGs send heartbeats, so only xtaghold supports the [xTAG HaLow Metadata Heartbeat \(0x0A\)](#) asynchronous message.

xTAG HaLow Metadata Heartbeat (0x0A)

Metadata attributes of an xTAG are sent automatically over HaLow after each xGATEWAY connection.

Command Format : NA (Always unsolicited)

Message Format<ID=0x0A>,<Len Byte = 0x1E>,<Error Byte>,<Type Byte>,<ID MS Byte>,<ID 2nd MS Byte>,<ID 3rd MS Byte>,<ID 4th MS Byte>,<ID 5th MS Byte>,<ID LS Byte>,<Battery Byte>,<RSSI MS Byte>,<RSSI LS Byte>,<RunSec MS Byte>,<RunSec 2nd MS Byte>,<RunSec 3rd MS Byte>,<RunSec LS Byte>,<UnixSec MS Byte>,<UnixSec 2nd MS Byte>,<UnixSec 3rd MS Byte>,<UnixSec LS Byte>,<CfgUnixSec MS Byte>,<CfgUnixSec 2nd MS Byte>,<CfgUnixSec 3rd MS Byte>,<CfgUnixSec LS Byte>,<HW Rev MS Byte>,<HW Rev LS Byte>,<SW Rev MS Byte>,<SW Rev LS Byte>,<Tx Power Byte>

Error Byte values:

- 0x00: Success

Notes

1. Type Byte values: Accelerometer with Position=0x10, Beacon=0x11, Env (temp, press, humid)=0x12, Env (temp, humid)=0x13, Gas (CO2)=0x20
2. ID Byte values denote the MAC address of the xTAG.
3. Battery Byte: 0x00-0xFF denoting range 0 Vdc to 3.3 Vdc (0 - 255 for 0-3.3 Vdc).
4. RSSI Bytes make up a signed word (INT16) denoting rssi with no implied decimals.
5. RunSec Bytes make up an unsigned double-word int (UINT32) denoting the number of sec since xTAG firmware start. Can be used for remote troubleshooting as xTAG restarts only occur upon xTAG firmware failures.
6. UnixSec Bytes make up an unsigned double-word int (UINT32) denoting the number of sec since midnight on Jan 1, 1970 GMT (as per Unix timestamp standard). The xTAG does not store this value between boot cycles, so if an xTAG is restarted (for any reason) a client with a valid clock must set the xTAGs Unix time if this time is to be used (i.e. future timestamping). If unix time is not set, these read values won't be valid.
7. CfgUnixSec Bytes make up an unsigned double-word int (UINT32) denoting the unix timestamp associated with the last configuration dataset downloaded and used by the xTAG Sensor. This is used by the xGATEWAY to check if a new dataset has been provided by the operator(with newer timestamp) and, if so, to update the xTAG with new configuration data.
8. HW Bytes denote a raw unsigned word with implied decimals indicating xTAG hardware revision. E.g. 0x2711 = 10001 denoting rev 1.00.01.

9. SW Bytes denote a raw unsigned word with implied decimals indicating xTAG software or firmware revision. E.g. 0x2775 = 10101 denoting rev 1.01.01.
10. Tx Power Byte set as dbm (8-18 dbm eg in NRC Docs).

xGATEWAY HaLow Metadata Heartbeat (0x1A)

Many Metadata attributes of an xGATEWAY are sent automatically through a stream port after connection. This supports clients monitoring things like xGATEWAY position (i.e. in telematics applications), xGATEWAY health (e.g. how long it has been running and its current clock) and current config params like current software version.

Command Format : NA (Always unsolicited)

Message Format<ID=0x1A>,<Len Byte = 0x1E>,<Error Byte>,<xGATEWAY Type Byte>,<ID MS Byte>,<ID 2nd MS Byte>,<ID 3rd MS Byte>,<ID 4th MS Byte>,<ID 5th MS Byte>,<ID LS Byte>,<RunSec MS Byte>,<RunSec 2nd MS Byte>,<RunSec 3rd MS Byte>,<RunSec LS Byte>,<UnixSec MS Byte>,<UnixSec 2nd MS Byte>,<UnixSec 3rd MS Byte>,<UnixSec LS Byte>,<HW Rev MS Byte>,<HW Rev LS Byte>,<SW Rev MS Byte>,<SW Rev LS Byte>,<Lat MS Byte>,<Lat 2nd MS Byte>,<Lat 3rd MS Byte>,<Lat LS Byte>,<Lon MS Byte>,<Lon 2nd MS Byte>,<Lon 3rd MS Byte>,<Lon LS Byte>

Error Byte values:

- 0x00: Success

Notes

- xGATEWAY Type Byte values: xGATEWAY Pure=0X00, xGATEWAY LTE=0X01, xGATEWAY HaLow=0X02, xGATEWAY 3.0=0X03, xGATEWAY Virtual (i.e. Tablet or Phone App)=0X04
- ID Byte values denote the MAC address of the xGATEWAY.
- RunSec Bytes make up an unsigned double-word int (UINT32) denoting device run time in sec.
- UnixSec Bytes make up an unsigned double-word int (UINT32) denoting the number of sec since midnight on Jan 1, 1970 GMT (as per Unix timestamp standard). This value is the xGATEWAYs internal clock value.
- HW Bytes denote a raw unsigned word with implied decimals indicating xTAG hardware revision. E.g. 0x2711 = 10001 denoting rev 1.00.01.
- SW Bytes denote a raw unsigned word with implied decimals indicating xTAG software or firmware revision. E.g. 0x2775 = 10101 denoting rev 1.01.01.
- Lat and Lon signed values (S32) include 7 implied decimals to represent Lat and Lon values in standard "Decimal Degrees" format with fractional components (i.e. no min or seconds). -1800000000 to 1800000000 denotes -180.0000000 to 180.0000000.

xTAG Metadata Read (0x0D)

Read back common config attributes written into the xGATEWAY for individually connecting xTAGs. The xTAG(s) will be updated upon their next connection to the xGATEWAY. CRITICALLY - if the xTAG is near multiple xGATEWAYS, each xGATEWAY must be sent the xTAG config to ensure that it is available to the xTAG regardless of which xGATEWAY it connects to.

Command Format: <Cmd=0x0D>,<Len Byte = 0x08><xTAG ID MS Byte>,<xTAG ID 2nd MS Byte>,<xTAG ID 3rd MS Byte>,<xTAG ID 4th MS Byte>,<xTAG ID 5th MS Byte>,<xTAG ID LS Byte>

Response Format: <Cmd=0x0D>,<Len Byte = 0x0C>,<Error Byte>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<Type Byte>,<Tx Power Byte>,<Heartbeat Freq MS Byte>,<Heartbeat Freq LS Byte>,<Heartbeat Retries Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument or Bad length.
- 0x02 - No Metadata exists within the xGATEWAY for the xTAG ID provided.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.
2. See [xTAG Metadata Write \(0x0E\)](#) below for details on additional message components.

xTAG Metadata Write (0x0E)

Write common attributes for individually connecting xTAGs. The xTAG(s) will be updated upon their next connection to the xGATEWAY. CRITICALLY - if the xTAG is near multiple xGATEWAYS, each xGATEWAY must be sent the xTAG config to ensure that it is available to the xTAG regardless of which xGATEWAY it connects to.

After metadata is written into xGATEWAYS for each xTAG, specific config data parameters (based on sensor type) must be written using messaging that is specific to each sensor type.

Command Format: <Cmd=0x0E>,<Len Byte = 0x11><xTAG ID MS Byte>,<xTAG ID 2nd MS Byte>,<xTAG ID 3rd MS Byte>,<xTAG ID 4th MS Byte>,<xTAG ID 5th MS Byte>,<xTAG ID LS Byte>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<Type Byte>,<Tx Power Byte>,<Heartbeat Freq MS Byte>,<Heartbeat Freq LS Byte>,<Heartbeat Retries Byte>

Response Format: <Cmd=0x0E>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument or Bad length.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.
2. Config UnixSec Byte values denote the unix time when the metadata was set. This will be used on the xGATEWAY to determine when a config data update is required on target devices. Note that this value will also be set on xGATEWAYS when sensor specific configurations are set (e.g. Acc config data or Env config data, etc ... see [xTAG Accelerometer Config Write \(0x24\)](#) for an example).
3. Type Byte values: Accelerometer with Position=0x10, Beacon=0x11, Env (temp, press, humid)=0x12, Env (temp, humid)=0x13, Gas (CO2)=0x20
4. Valid Tx Power byte values are 10 to 18 dbm.
5. Heartbeat frequency values are specified in min.
6. To avoid long xTAG wake cycles, set Heartbeat Retries to a value of 3 or fewer. This is the number of xGATEWAY retries attempted when trying to send a heartbeat msg. This value should never be higher than 5. For best power consumption - use 1. If more than 3 retries are required, there is very likely to be no xGATEWAY nearby that is connectable.

xTAG FOTA Update (0x0F)

Signal an xGATEWAY to apply an xTAG Firmware Over The Air or FOTA update to an xTAG when it next connects to the xGATEWAY. Note that a firmware update binary file will be downloaded from the Deviceworx cloud to support this update (if it has not already been downloaded for another xTAG). Therefore, an xGATEWAY - Deviceworx cloud connection must be available before this command can be executed.

xTAG firmware update files are stored on xGATEWAYS within the /root/cfg_halow/ folder after they have been downloaded from the Deviceworx cloud. Files have the following format:

xtaghlo_nnnnn-mmmmm...m.bin where nnnnn represents a 5 digit rev number with 0 padding and mmmmm...m denotes a CRC value

For example, "xtaghlo_10204_03AF422B.bin denotes firmware rev 1.02.04 or 1.2.4 and CRC 3AF422B (always 8 chars with "0" padding when required - represented by 4 bytes).

Command Format: <Cmd=0x0F>,<Len Byte = 0x0E><xTAG ID MS Byte>,<xTAG ID 2nd MS Byte>,<xTAG ID 3rd MS Byte>,<xTAG ID 4th MS Byte>,<xTAG ID 5th MS Byte>,<xTAG ID LS Byte>,<firmware MS Byte>,<firmware LS Byte>,<CRC byte 1>,<CRC byte 2>,<CRC byte 3>,<CRC byte 4>

Response Format: <Cmd=0x0F>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument or Bad length.
- 0x02 - No firmware file exists within the xGATEWAY matching the name provided.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.
2. If the xGATEWAY has a connection to the Deviceworx cloud, it will fetch a firmware binary from the cloud. If no such connection is feasible (i.e. the xGATEWAY is on a private network), the firmware binary will have to be copied into the xGATEWAY /root/cfg_halow/ folder using xGATEWAY SFTP support.
3. Firmware binaries, along with their CRC values, can be provided by Deviceworx for local copy to the xGATEWAY or for download from the Deviceworx cloud. Consult Deviceworx support to obtain any available binaries.

Socket Interface Command Supported by All Daemons

xGATEWAY-Specific Commands

xGATEWAY Metadata Read (0x1B)

Write attributes of the xGATEWAY for immediate use.

Command Format: <Cmd=0x1B>,<Len Byte = 0x02>

Response Format: <Cmd=0x1B>,<Len Byte = 0x12>,<Error Byte>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<Channel MS Byte>,<Channel LS Byte>,<Tx Pwr Byte>,<Lat MS Byte>,<Lat 2nd MS Byte>,<Lat 3rd MS Byte>,<Lat LS Byte>,<Lon MS Byte>,<Lon 2nd MS Byte>,<Lon 3rd MS Byte>,<Lon LS Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad Len Byte

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Config Unix time (in sec) denotes when the config data was updated within the host app (before last write).
3. Lat and Lon signed values include 7 implied decimals to represent Lat and Lon values in standard "Decimal Degrees" format with fractional components (i.e. no min or seconds). -1800000000 to 1800000000 denotes -180.0000000 to 180.0000000.

xGATEWAY Metadata Write (0x1C)

Write attributes of the xGATEWAY for immediate use.

Command Format <Cmd=0x1C>,<Len Byte = 0x11>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<Channel MS Byte>,<Channel LS Byte>,<Tx Pwr Byte>,<Lat MS Byte>,<Lat 2nd MS Byte>,<Lat 3rd MS Byte>,<Lat LS Byte>,<Lon MS Byte>,<Lon 2nd MS Byte>,<Lon 3rd MS Byte>,<Lon LS Byte>

Response Format<Cmd=0x1C>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Config Unix time (in sec) denotes when the config data was updated within the host app. A time that is newer than that for stored config data signals the xGATEWAY to use the new config parameters in the message. This new time will then be stored and compared to subsequent message times.
3. Valid channel values are 1 through 11 excluding 4.
4. Valid tx power byte values are 11 through 18 dbm.
4. The xGATEWAY will restart after this command is executed (to facilitate use of the new vars)
5. Lat and Lon 32 bit signed values (each) include 7 implied decimals to represent Lat and Lon values in standard "Decimal Degrees" format with fractional components (i.e. no min or seconds). -1800000000 to 1800000000 denotes -180.0000000 to 180.0000000.

Shutdown (0xFF)

Cleanly shutdown the xGATEWAY daemon and support disconnection of any xTAGs, etc before the socket connection is closed. Use this command to reset a daemon (if ever required) as it will be automatically restarted after shutting down.

Command Format <Cmd=0xFF>,<Len Byte = 0x02>

Response Format<Cmd=0xFF>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x7F: General Error. Check things like correct Len Byte.

Notes

1. Ensure that a 20 sec delay is observed after this command is executed before reconnecting to a restarted daemon as shutdown and restart tasks won't be immediate.

xTAG-Specific Commands

xTAG Sensor Acq Start (0x18)

Start non-streaming, periodic acquisition from environmental, CO2 and similar sensors. Note that other sensors (accelerometer-based vibration sensors) may have their own, specific start/stop commands.

Command Format <Cmd=0x18>,<Len Byte = 0x0D><xTAG ID MS Byte>,<xTAG ID 2nd MS Byte>,<xTAG ID 3rd MS Byte>,<xTAG ID 4th MS Byte>,<xTAG ID 5th MS Byte>,<xTAG ID LS Byte>,<Shock Thresh Byte>,<Sample Freq MSB>,<Sample Freq 2nd MSB>,<Sample Freq 3rd MSB>,<Sample Freq LSB>

Response Format<Cmd=0x18>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. The Shock Thresh value denotes a shock level in G's. This is only used with some, specialized Environmental sensors. Set to 0.
3. The Sample Freq byte values denote a sample frequency in seconds. Typically this is 10 sec or higher.

xTAG Sensor Acq Stop (0x19)

Stop non-streaming, periodic acquisition from environmental, CO2 and similar sensors. Note that other sensors (accelerometer-based vibration sensors) may have their own, specific start/stop commands.

Command Format <Cmd=0x19>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x19>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.

xTAG Accelerometer Config Read (0x23)

Read the accelerometer config on a connected xTAG.

Command Format <Cmd=0x23>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x23>,<Len Byte = 0x07>,<Error Byte>,<G Range Byte>,<G ODR Byte><G US & BWP Byte><Stream Sec Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. See the [xTAG Accelerometer Config Write \(0x24\)](#) command for details on the <G Range Byte>,<G ODR Byte> and <G US & BWP Byte> including their default values.

xTAG Accelerometer Config Write (0x24)

Config the accelerometer on a connected xTAG.

Command Format <Cmd=0x24>,<Len Byte = 0x10><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<G Range Byte>,<G ODR Byte><G US & BWP Byte><Stream Sec Byte>

Response Format<Cmd=0x24>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.
2. Config UnixSec Byte values denote the unix time when the accelerometer config data was set. This will be used on the xGATEWAY to determine when a config data update is required on target devices. Note that this value will also be set on xGATEWAYS when metadata is written (i.e. either a Metadata or accelerometer config write will set the value).
3. The <G Range Byte> values are written directly into the ACC_RANGE register (0x41) in the BMI160:
 - a. 0x03: 2 G
 - b. 0x05: 4 G
 - c. 0x08: 8 G
 - d. 0x0C: 16 G
4. The <G ODR Byte> values are written directly into the least significant nibble of the ACC_CONF register (0x40) in the BMI160:
 - a. 0x06: 25 samples/sec
 - b. 0x07: 50 samples/sec
 - c. 0x08: 100 samples/sec
 - d. 0x09: 200 samples/sec
 - e. 0x0A: 400 samples/sec
 - f. 0x0B: 800 samples/sec
 - g. 0x0C: 1600 samples/sec
5. The <G US & BWP Byte> values are written directly into the most significant nibble of the ACC_CONF register (0x40) in the BMI160. The only supported values are.
 - a. 0x02: Undersampling disabled - normal filter mode. No oversampling. See table 12 in the BMI160 guide for 3db cutoff freq in this mode.
 - b. 0x01: Undersampling disabled - OSR2 filter mode. Oversampling rate 2.
 - c. 0x00: Undersampling disabled - OSR4 filter mode. Oversampling rate 2.
6. The <Stream Sec Byte> value denotes how long in sec (0-120) to stream data after threshold is broken or upon the next HaLow heartbeat (cleared after streaming). Note that this value is only used when

supporting HaLow xTAGs and is ignored on other types as explicit connect, start, stop sequences are possible on these types.

7. See [Socket Interface](#) for details on how to set a Phy value of 0 (125 kbps), 1 (1 Mbps) or 2 (2 Mbps). No phy value is required when starting xtagusbd for usb comms to xTAGs.

Accelerometer Acquisition Stream Start (0x26)

Start accelerometer FIFO reads.

Command Format <Cmd=0x26>,<Len Byte = 0x0A><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<On Thresh Byte>,<Off Thresh Byte>

Response Format<Cmd=0x26>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length.
- 0x02: Processing Error. Non-zero <On/Off Thresh Byte> specified with ODR over 100 samples/s.
- 0x03: No Connection Error.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Starting (if successful) will automatically initialize the return of multiple stream data responses.
3. Issue a stream stop cmd to stop stream data responses.
4. If already streaming, this command will be ignored.
5. The On and Off Thresh Byte arguments support delayed start of the stream. Streaming will continue until stopped through this API. To immediately start streaming, simply set On Thresh Byte and Off Thresh Byte values to zero. A detailed description of delayed streaming follows in a separate paragraph.
6. Only the On Thresh Byte or Off Thresh Byte should be non-zero for any given call, but not both. If both On and Off Thresh Byte values are non-zero, the Off Thresh Byte will be treated as being zero.
7. On and Off Thresh Byte values range from 0-255 (UINT8). This value denotes a G value (within 50 percent of the selected G range). To select a top-range threshold value, use 255 (255/255 * 0.5 of G Range ... eg 1G on a +/-2G range). To select a mid-range threshold value, use 127 (127/255 * 0.5 of G Range ... eg 1G on a +/-4G range). To select a lower-range threshold value, use 63 (63/255* 0.5 of G Range ... eg approx 1G on a +/-8G range). Note that these mid and lower range values are provided as examples only. Any value between 0 - 255 is valid.

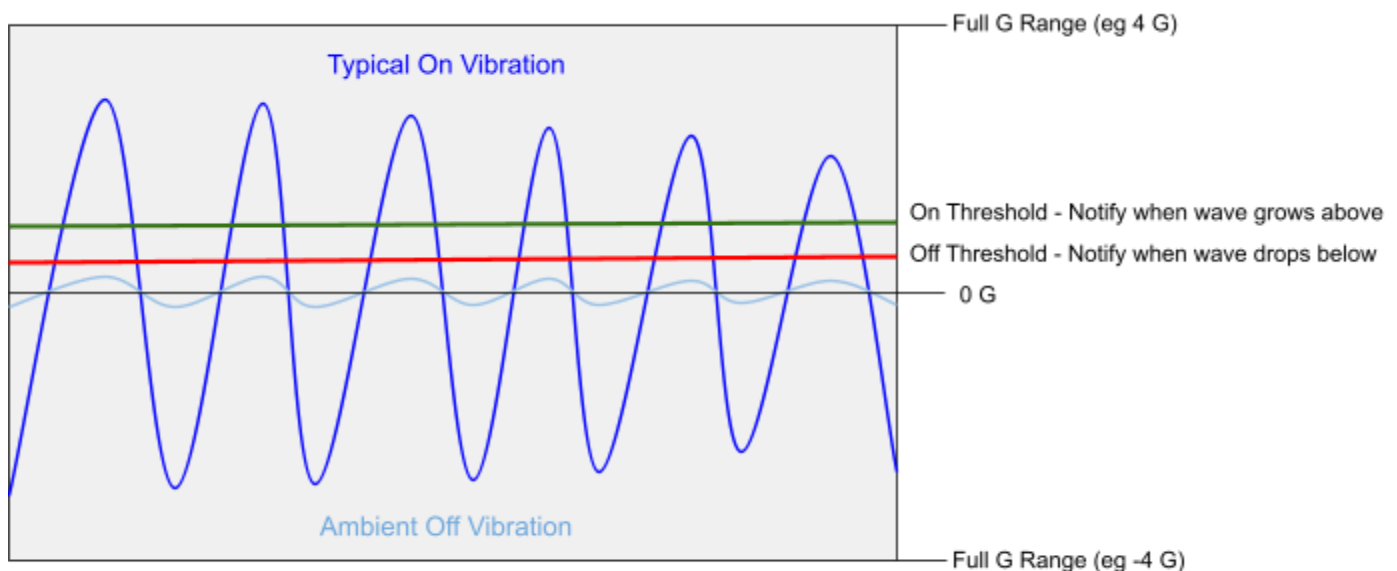
Delayed Stream Start Until Threshold Cross

To simplify xGATEWAY monitoring of machine On or Off status, and dramatically reduce required xTAG battery draw, the xTAG accelerometer's ability to watch for Significant Motion to see when a machine is On (operating or started) or to watch for No Motion to see when a machine is Off (idle or stopped) is leveraged. This is done by setting up the xTAG accelerometer to notify the xTAG microcontroller when it senses a machine is on or off. After getting notified (and not until), the xTAG microcontroller will stream data until a stream stop command is received. The xTAG accelerometer notifies the xTAG microcontroller that a machine is On whenever measured G values increase past a threshold. It notifies the xTAG that a machine is Off whenever measured G values

decrease below a threshold. Only On or Off notification is supported at this time. Typical xGATEWAY client steps for using this feature are:

1. The API client Issues a start stream command (0x16) with On Thresh Byte non-zero to watch for a machine start.
2. The client then waits for the stream to start (0x17 responses returned) indicating that the machine has come On. xTAGs can watch for this state change using minimal battery power. Clients process stream response as required.
3. The client issues a stop stream command (0x18) to stop streaming (incoming 0x17 responses).
4. The API client issues a start stream command (0x16) with Off Thresh Byte non-zero to watch for the machine stop.
5. The client then waits for the stream to start indicating that the machine is Off. xTAGs can watch for this state change using minimal battery power. Read stream data as required.
6. Issue stop stream command (0x18) to stop streaming (incoming 0x17 responses).

On and Off Thresh bytes must be selected with machine vibration characteristics in mind. These characteristics include typical On vibration magnitude and typical ambient vibration magnitude (eg. vibration from machinery nearby). When watching for a machine to start or come On, the On threshold must be well above ambient vibration magnitude (to avoid false positives) and well below machine On vibration magnitude (to be reliable). When watching for a machine to stop or turn Off, the Off threshold must still be above ambient vibration (to avoid falsely reporting a machine Off event). The chart below shows candidate threshold values within an example vibration signal.



Note On and Off Threshold Max is 50% of G Range (eg 2 G)

Contact Deviceworx support (support@deviceworx.com) as required for assistance in setting up delayed stream start for On or Off machine state monitoring.

NOTE - the goal of this feature is to dramatically reduce xTAG battery draw when monitoring a machine state that changes infrequently. For best efficiency, set up the xTAG sample rate at 25 or 50 samples per sec. 100 samples per sec is the max supported rate. Increasing the sample rate above this range will increase battery draw as this sample rate is used by the accelerometer to determine how frequently to sample acceleration values and notify the xTAG microcontroller when a threshold is crossed.

The stream start (due to accelerometer notification) is not immediate. Expect a 0.5 to 2 sec delay before the xTAG sends the first stream message after a threshold crossing has been determined.

Accelerometer Acquisition Stream Data (Responses Only) (0x2A)

Whenever data is received from the BMI160 accelerometer (acc) on the sensor (as a result of a Stream Start cmd), this data will be automatically sent to the host as stream data responses. Formatting of these data responses is described here.

Stream Data Response (no error): <Cmd=0x2A><Resp Len>,<Error Byte=0x00>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<X1 LSB><X1 MSB><Y1 LSB><Y1 MSB><Z1 LSB><Z1 MSB><X2 LSB><X2 MSB><Y2 LSB><Y2 MSB><Z2 LSB><Z2 MSB>... <Xn LSB><Xn MSB><Yn LSB><Yn MSB><Zn LSB><Zn MSB>

ATYPICAL DATA LSB BEFORE MSB - THIS IS TO MATCH DATA RETRIEVAL FROM THE ACC CHIP.

Ongoing Stream Data Response (plugged stream): <Cmd=0x2A><Resp Len=0x04>,<Error Byte=0x05>,<Removed Samples Byte>

Error Byte values:

- 0x00: Cmd Success
- 0x05: Plugged Stream

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. The max number of samples that can be returned from a Sensor within each msg is 40 samples (40x6=240 bytes). Note that multiple msgs will be returned per second to satisfy the sample/sec selected. For example, if a sample rate of 200 samples / sec is selected 5 msgs, with 40 samples each, will be returned each second from that Sensor.

Accelerometer Acquisition Stream Stop (0x27)

Stop accelerometer FIFO stream reads.

Command Format <Cmd=0x27>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format <Cmd=0x27>,<Len Byte = 0x03>,<Error Byte>

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. This command will simply stop the data stream responses (0x2A). Expect these responses to stop and then receive the Stream Stop Cmd (0x2A) response described here.
3. Sensor streaming can stop if its stream backs up too much due to buffering overflow when the wireless connection does not support the sample rate on the stream.

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x04: Bad state. Not streaming when the stop command is sent.
- 0x7F: General Error.

xTAG Environmental Config Read (0x40)

Read the environmental sensor config on a connected xTAG.

Command Format <Cmd=0x40>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x40>,<Len Byte = 0x07>,<Error Byte>,<TP IIR Filter Byte>,<Temp Oversample Byte>,<Press Oversample Byte>,<Humidity Oversample Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. See the [xTAG Environmental Config Write \(0x41\)](#) command for details on <TPH Filter Byte>,<Temp Oversample Byte>,<Press Oversample Byte>,<Humidity Oversample Byte> details including their recommended values.

xTAG Environmental Config Write (0x41)

Read the environmental sensor config on a connected xTAG.

Command Format <Cmd=0x41>,<Len Byte = 0x10><xTAG ID MS Byte>,<xTAG ID 2nd MS Byte>,<xTAG ID 3rd MS Byte>,<xTAG ID 4th MS Byte>,<xTAG ID 5th MS Byte>,<xTAG ID LS Byte>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<TP IIR Filter Byte>,<Temp Oversample Byte>,<Press Oversample Byte>,<Humidity Oversample Byte>

Response Format<Cmd=0x41>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.
2. Config UnixSec Byte values denote the unix time when the accelerometer config data was set. This will be used on the xGATEWAY to determine when a config data update is required on target devices. Note that this value will also be set on xGATEWAYS when metadata is written (i.e. either a Metadata or accelerometer config write will set the value).
3. The TP IIR Filter value stipulates whether to apply IIR filtering on Temperature and Pressure (not Humidity) samples. A value of 0-127 can be applied ('0' disables IIR). Deviceworx recommends disabling IIR (setting this value to 0) as its intended use (to filter values changing quickly between successive samples) does not make sense when sampling over longer periods of time (e.g. 10+ sec). Temperature and atmospheric pressure will not change that quickly.
4. The Temperature, Pressure and Humidity OverSample Rate or OSR values are intended to support more accurate sampling through multiple chip samples before transmission of a result sample. Deviceworx recommends a Temp OSR of 1x (1) and Pressure OSR of 2x (2) and Pressure OSR of 4x(4) for best results.

xTAG Environmental Sample Read (0x4A)

Read the environmental sensor sample stored on the xGATEWAY. Samples from all sensors are read.

Command Format <Cmd=0x4A>,<Len Byte = 0x02>

Response Format<Cmd=0x4A>,<Len Byte = N>,<Error Byte>,<Rec 1 ID MS Byte>,<Rec 1 ID 2nd MS Byte>,<Rec 1 ID 3rd MS Byte>,<Rec 1 ID 4th MS Byte>,<Rec 1 ID 5th MS Byte>,<Rec 1 ID LS Byte>,<Rec 1 Unix MS Byte>,<Rec 1 Unix 2nd MS Byte>,<Rec 1 Unix 3rd MS Byte>,<Rec 1 Unix LS Byte>,<Rec 1 Temp MS Byte>,<Rec 1 Temp LS Byte>,<Rec 1 Press MS Byte>,<Rec 1 Press 2nd MS Byte>,<Rec 1 Press 3rd MS Byte>,<Rec 1 Press LS Byte>,<Rec 1 Humidity MS Byte>,<Rec 1 Humidity 2nd MS Byte>,<Rec 1 Humidity 3rd MS Byte>,<Rec 1 Humidity LS Byte> ... <Rec N ID MS Byte>....<Rec N Humidity LS Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. A success return (0) with total length (Len Byte) of 0 is possible when there are no data records to fetch.
3. There are 6 ID bytes, 4 Unix bytes, 2 Temperature bytes, 4 Pressure bytes and 4 Humidity bytes for each sample record. That is 20 bytes for each record. Therefore total length will always be 20xSamples plus 3 bytes for Cmd, Length and Error. For example 23 bytes for 1 sample, 43 bytes for 2 samples, etc.
4. Unix timestamps (number of seconds since 1970 when the sample was taken) are in the GMT time zone.
5. The temperature bytes denote a signed, 16-bit degree celsius value with 2 implied decimals (i.e. -32767 to 32767 denoting -327.67 to 327.67 deg C). Of course typical temperatures outdoors are around 20.00 deg C.
6. The pressure bytes denote an unsigned 32-bit pressure in kPa with 3 implied decimals (i.e. 0.000 - 4294967.295 kPa). Of course typical atmospheric pressure values are only 101.1 kPa.
7. The Humidity bytes denote an unsigned 32-bit relative humidity in % with 3 implied decimals ((i.e. 0.000 - 4294967.295 % RH). Of course typical atmospheric % RH values range from 15.000 to 100.000 %RH.
8. Note that a max of 400 samples can be read at a time. The first 400 samples (if more than 400 buffered in xGATEWAY) will be returned when there are more stored on the xGATEWAY.
9. After reading, samples read will be removed from xGATEWAY storage (leaving unread samples).

xTAG CO2 Config Read (0x50)

Read the CO2 sensor config on a connected or connectable xTAG.

Command Format <Cmd=0x50>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x50>,<Len Byte = 0x09>,<Error Byte>,<Alt MS Byte>,<Alt LS Byte>,<Atm Press MS Byte>,<Atm Press LS Byte>,<Temp Offset MS Byte>,<Temp Offset LS Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. See the [xTAG CO2 Config Write \(0x51\)](#) command for details on <Alt XX Byte>,<Atm Press XX Byte>,<Temp Offset XX Byte> details including their recommended values.

xTAG CO2 Config Write (0x51)

Read the environmental sensor config on a connected xTAG.

Command Format <Cmd=0x51>,<Len Byte = 0x12><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<Config UnixSec MS Byte>,<Config UnixSec 2nd MS Byte>,<Config UnixSec 3rd MS Byte>,<Config UnixSec LS Byte>,<Alt MS Byte>,<Alt LS Byte>,<Atm Press MS Byte>,<Atm Press LS Byte>,<Temp Offset MS Byte>,<Temp Offset LS Byte>

Response Format<Cmd=0x51>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte.
2. Config UnixSec Byte values denote the unix time when the accelerometer config data was set. This will be used on the xGATEWAY to determine when a config data update is required on target devices. Note that this value will also be set on xGATEWAYS when metadata is written (i.e. either a Metadata or accelerometer config write will set the value).
3. The Alt Bytes denote an unsigned 16-bit value that stipulates the expected altitude in meters above sea level where the sensor will be used. This altitude will affect the atmospheric pressure (when not provided).
4. The Atm Press Bytes denote an unsigned 16-bit value with 1 implied decimal that stipulates a recent current atmospheric pressure in kPa (e.g. 1024 denotes 102.4 kPa). With the current atmospheric pressure, the sensor is able to improve its accuracy. If this value is set, the Alt Bytes can be set to '0'. This supports the best sensor accuracy if it currently reflects the atmospheric pressure in which the sensor operates. Many sensors (in the same area) can be "fed" this value every hour or so to improve sensor performance. If only sensor altitude is known, set this value to '0' and set a reasonably accurate Alt value.
5. The Temp Offset Bytes denote a signed 16-bit value with 3 implied decimals that facilitates "trimming" the temperature value returned from the sensor. As this temperature value use is reasonably inaccurate (compared to the Deviceworx Environmental sensor), it is not expected to be used and this value can be set to 0. Importantly - this value will not affect the CO2 measurement.

xTAG CO2 Sample Read (0x5A)

Read the CO2 sensor samples stored on the xGATEWAY. Samples from all sensors are read.

Command Format <Cmd=0x5A>,<Len Byte = 0x02>

Response Format<Cmd=0x5A>,<Len Byte = N>,<Error Byte>,<Rec 1 ID MS Byte>,<Rec 1 ID 2nd MS Byte>,<Rec 1 ID 3rd MS Byte>,<Rec 1 ID 4th MS Byte>,<Rec 1 ID 5th MS Byte>,<Rec 1 ID LS Byte>,<Rec 1 Unix MS Byte>,<Rec 1 Unix 2nd MS Byte>,<Rec 1 Unix 3rd MS Byte>,<Rec 1 Unix LS Byte>,<Rec 1 CO2 MS Byte>,<Rec 1 CO2 LS Byte>,<Rec 1 Temp MS Byte>,<Rec 1 Temp 2nd MS Byte>,<Rec 1 Temp 3rd MS Byte>,<Rec 1 Temp LS Byte>,<Rec 1 Humidity MS Byte>,<Rec 1 Humidity 2nd MS Byte>,<Rec 1 Humidity 3rd MS Byte>,<Rec 1 Humidity LS Byte> ... <Rec N ID MS Byte>....<Rec N Humidity LS Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error. USB only.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. A success return (0) with total length (Len Byte) of 0 is possible when there are no data records to fetch.
3. There are 6 ID bytes, 4 Unix bytes, 2 Temperature bytes, 4 Pressure bytes and 4 Humidity bytes for each sample record. That is 20 bytes for each record. Therefore total length will always be 20xSamples plus 3 bytes for Cmd, Length and Error. For example 23 bytes for 1 sample, 43 bytes for 2 samples, etc.
4. Unix timestamps (number of seconds since 1970 when the sample was taken) are in the GMT time zone.
5. The CO2 bytes denote an unsigned 16-bit Parts Per Million or PPM value for measured CO2 concentration (0-32768 ppm). Sensor electronics are limited to 4000 ppm, so expect values between 0-4000.
6. The temperature bytes denote a signed, 32-bit degree celsius value with 3 implied decimals (i.e. -2147483647 to 2147483647 denoting -2147483.647 to 2147483.647 deg C). Of course typical temperatures outdoors are around 20.000 deg C.
7. The Humidity bytes denote an unsigned 32-bit relative humidity in % with 3 implied decimals ((i.e. 0.000 - 4294967.295 % RH). Of course typical atmospheric % RH values range from 15.000 to 100.000 %RH.
8. Note that a max of 400 samples can be read at a time. The first 400 samples (if more than 400 buffered in xGATEWAY) will be returned when there are more stored on the xGATEWAY.
9. After reading, samples read will be removed from xGATEWAY storage (leaving unread samples).