Sitch Sensor Documentation

Release 4.1

Ash Wilson

Jul 22, 2020

Contents:

1 Sensor Environment Variables **3**

2 SITCH Sensor Alert Types **5**

3 Understanding the Log Data Collected by Sitch **7**
- 3.1 cells.log . . . 7
- 3.2 geoip.log . . . 9
- 3.3 gps.log . . . 10
- 3.4 gsm_modem_channel.log . . . 10
- 3.5 health_check.log . . . 12
- 3.6 heartbeat.log . . . 13
- 3.7 kal_channel.log . . . 13
- 3.8 scanner.log . . . 14
- 3.9 sitch_alert.log . . . 14
- 3.10 sitch_init.log . . . 14

4 Event Lifecycle **17**
- 4.1 Ingestion . . . 17
- 4.2 Decomposition . . . 17
- 4.3 Correlation . . . 17
- 4.4 Transmission . . . 18
- 4.5 Reception . . . 18

5 Sensor Troubleshooting **19**
- 5.1 GSM modem device detection . . . 19
- 5.2 Found but undetected TTY . . . 20
- 5.3 GPS device not detected (U-Blox7) . . . 21
- 5.4 No events in Kibana . . . 21

6 Reference Images **23**
- 6.1 Connecting the USB TTY cable to the SIM 900 GSM modem . . . 23

7 SITCH Sensor Internal API **25**
- 7.1 AlertManager . . . 26
- 7.2 ArfcnCorrelator . . . 26
- 7.3 CgiCorrelator . . . 26
- 7.4 ConfigHelper . . . 26

7.5 Decomposer . . . 26
7.6 DeviceDetector . . . 26
7.7 FeedManager . . . 26
7.8 GeoCorrelator . . . 26
7.9 GeoIp . . . 26
7.10 GeoipDecomposer . . . 26
7.11 GpsDecomposer . . . 26
7.12 GpsListener . . . 26
7.13 GsmDecomposer . . . 26
7.14 GsmModem . . . 26
7.15 KalDecomposer . . . 26
7.16 LocationTool . . . 26
7.17 Logger . . . 26
7.18 Utility . . . 26

8 Indices and tables **27**

Version 4.1

CHAPTER 1

Sensor Environment Variables

The SITCH Sensor requires some environment variables to be set in order to operate.

Environment Variable	Purpose
CGI_WHITELIST	(Optional) List of trusted CGIs.
FEED_RADIO_TARGETS	(Optional) Radio types to target for feed ingestion. Defaults to `GSM`
FEED_URL_BASE	(Optional) Base URL for Sensor feed. Defaults to SITCH auto-built public feed
GSM_MODEM_BAND	Restrict GSM modem to this band. Options: (EGSM_MODE | PGSM_MODE | DCS_MODE | GSM850_MODE | PCS_MODE | EGSM_DCS_MODE | GSM850_PCS_MODE | EGSM_PCS_MODE | ALL_BAND) Defaults to `ALL_BAND`
GSM_MODEM_PORT	(Optional) Set the tty for the GSM modem. If unset, the Sensor will attempt to auto-configure
KAL_BAND	Band for Kalibrate to scan. (GSM850 | GSM-R | GSM900 | EGSM | DCS | PCS) Defaults to `GSM850`
KAL_GAIN	Gain value for Kalibrate. Defaults to `60`
KAL_THRESHOLD	Hard threshold for Kalibrate channel power level. Defaults to `1000000`
LOCATION_NAME	Name of the location for this sensor. No spaces.
LOG_HOST	Logstash endpoint. Formatted like this: `hostname:port`
MCC_LIST	(Optional) List of Mobile Country Codes to ingest from feed. List is comma-separated: `310,311,316` Defaults to `310,311,312,316`
MODE	Set to `clutch` to go into a wait loop on start. Useful for troubleshooting. Defaults to `full`
NO_FEED_UPDATE	(Optional) If set, do not attempt to update the feed on boot.
STATE_LIST	Comma-separated list of states for feed ingestion. California and Texas would be: `CA,TX`
VAULT_PATH	Path to Logstash/Filebeat credentials in Vault. Defaults to `secret/client`, which will work with the demo environment.
VAULT_TOKEN	Client token used to retrieve credentials from Vault.
VAULT_URL	URL for Vault instance containing Logstash/Filebeat credentials. Looks like: `https://server.com:8200`
NO_FEED_UPDATE	If set, do not attempt to update the feed on boot.
GSM_MODEM_PORT	(Optional) GSM modem USB-TTY port. This should be autodetected and not need to be set. Looks like: `/dev/ttyUSB0` See: “Found but undetected TTY ” in the docs
GPS_DEVICE_PORT	(Optional) GPS device USB-TTY port. This should be autodetected and not need to be set. Looks like: `/dev/ttyUSB0` See: “Found but undetected TTY ” in the docs

CHAPTER 2

SITCH Sensor Alert Types

SITCH has a well-defined set of alerts, which are meant to be easy to parse with a log management or SIEM system.

The alert log message format is defined here: http://sensor.readthedocs.io/en/test/data.html#sitch-alert-log

The supported message types are listed here (in the __init__ function): http://sensor.readthedocs.io/en/test/_modules/sitchlib/alert_manager.html#AlertManager

CHAPTER 3

Understanding the Log Data Collected by Sitch

The following sections describe the data for the files found in '/data/sitch/log/'.

3.1 cells.log


```
{"scan_results": [
  {"cgi_str": "310:260:275:20000",
   "site_name": "sitch-site-testing",
   "bsic": "16",
   "mcc": "310",
   "rla": 0,
   "lac": "275",
   "band": "ALL_BAND",
   "feed_info": {
     "mcc": "310",
     "lon": "-122.464146",
     "lac": "275",
     "range": 325,
     "lat": "37.776641",
     "mnc": "260",
     "cellid": "20082"},
   "scan_location": "sitch-site-testing",
   "mnc": "260",
   "txp": 03,
   "distance": 534.3820159387475,
   "scan_finish": "2017-05-06T06:25:49.837957",
   "rxl": 20.0,
   "cell": 0,
   "scanner_public_ip": "1.1.1.1",
   "rxq": 0.0,
   "ta": 255,
   "cellid": "20082",
   "cgi_int": 31026027520082,
```


(continues on next page)

(continued from previous page)


```
  "arfcn": 684}
  ...],
 "band": "ALL_BAND",
 "site_name": "sitch-site-testing",
 "platform": "Unspecified",
 "scan_start": "",
 "scan_location": "sitch-site-testing",
 "scanner_public_ip": "1.1.1.1",
 "scan_finish": "2017-05-06T06:25:49.837957",
 "scan_program": "gsm_modem"
 "event_timestamp": "2017-05-06T06:25:49.837957"}
```


3.1.1 <cell>

possible values	description
0	The serving cell
1-6	The index of the neighboring cell

3.1.2 <arfcn>

[Absolute radio frequency channel number](https://en.wikipedia.org/wiki/Absolute_radio-frequency_channel_number)

3.1.3 <rxl>

Receive level

The measured signal level shall be mapped to an RXLEV value between 0 and 63, as follows:

possible values	description
0	less than -110 dBm.
1	-110 dBm to -109 dBm.
2	-109 dBm to -108 dBm.
...	
...	
62	-49 dBm to -48 dBm.
63	greater than -48 dBm.

3.1.4 <rxq>

Receive quality

possible values	description
0...7	as [RXQUAL](https://en.wikipedia.org/wiki/Rxqual) values
99	not known or not detectable

3.1.5 <mcc>

[Mobile country code](https://en.wikipedia.org/wiki/Mobile_country_code)

3.1.6 <mnc>

[Mobile network code](https://en.wikipedia.org/wiki/Mobile_country_code)

3.1.7 <bsic>

[Base station identity code](https://en.wikipedia.org/wiki/Base_station_identity_code)

3.1.8 <cellid>

[Cell id](https://en.wikipedia.org/wiki/Cell_ID)

NOTE: In a 7-item line, cellid is not provided. We set it to 0 to prevent barfing elsewhere.

3.1.9 <lac>

[Location area code](http://www.telecomabc.com/l/lac.html)

3.1.10 <rla>

Receive level access minimum

GUESS: Minimum receiving level permitted to access the system Per: similar AT engineering mode (AT+QENG) command in [M95 AT commands manual](http://eddywireless.com/yahoo_site_admin/assets/docs/M95_AT_Commands_Manual_V12.196112248.pdf)

3.1.11 <txp>

Transmit power maximum CCCH

3.1.12 <TA>

[Timing Advance](https://en.wikipedia.org/wiki/Timing_advance)

3.2 geoip.log


```
{"geometry":
  {"type": "Point",
   "coordinates": [-122, 37]
   },
 "scan_program": "geo_ip",
 "type": "Feature",
 "event_timestamp": "2017-05-06T06:25:49.837957"}
```

This is in geojson structure, with the addition of an event_timestamp field.

3.3 gps.log


```
{"sat_time": "2017-05-02T06:26:08.000Z",
 "geometry": {
   "type": "Point",
   "coordinates":
     [-122, 37]
     },
 "time_drift": 0.006355733333333334,
 "sys_time": "2017-05-02T06:26:08.381344",
 "scan_program": "gpsd",
 "type": "Feature"
 "event_timestamp": "2017-05-06T06:25:49.837957"}
```


3.4 gsm_modem_channel.log


```
{"cgi_str": "310:260:275:20082",
"site_name": "sitch-site-testing",
"bsic": "16",
"mcc": "310",
"rla": 8,
"lac": "275",
"band": "ALL_BAND",
"feed_info": {
    "mcc": "310",
    "lon": "-122.123",
    "lac": "275",
    "range": 325,
    "lat": "37.123",
    "mnc": "260",
    "cellid": "20082"
    },
"scan_location": "sitch-site-testing",
"mnc": "260",
"txp": 3,
"distance": 568.12345,
"scan_finish": "2017-05-16T02:21:23.901298",
"event_timestamp": "2017-05-16T02:21:23.901298",
"rxl": 24.0,
"cell": 0,
"scanner_public_ip": "1.1.1.1",
"rxq": 0.0,
"ta": 255,
"cellid": "20082",
"cgi_int": 31026027520082,
"arfcn": 684}
```

3.4.1 <cell>

possible values	description
0	The serving cell
1-6	The index of the neighboring cell

3.4.2 <arfcn>

[Absolute radio frequency channel number](https://en.wikipedia.org/wiki/Absolute_radio-frequency_channel_number)

3.4.3 <rxl>

Receive level

The measured signal level shall be mapped to an RXLEV value between 0 and 63, as follows:

possible values	description
0	less than -110 dBm.
1	-110 dBm to -109 dBm.
2	-109 dBm to -108 dBm.
. . .	
. . .	
62	-49 dBm to -48 dBm.
63	greater than -48 dBm.

3.4.4 <rxq>

Receive quality

possible values	description
0. . . 7	as [RXQUAL](https://en.wikipedia.org/wiki/Rxqual) values
99	not known or not detectable

3.4.5 <mcc>

[Mobile country code](https://en.wikipedia.org/wiki/Mobile_country_code)

3.4.6 <mnc>

[Mobile network code](https://en.wikipedia.org/wiki/Mobile_country_code)

3.4.7 <bsic>

[Base station identity code](https://en.wikipedia.org/wiki/Base_station_identity_code)

3.4.8 <cellid>

[Cell id](https://en.wikipedia.org/wiki/Cell_ID)

NOTE: In a 7-item line, cellid is not provided. We set it to 0 to prevent barfing elsewhere.

3.4.9 <lac>

[Location area code](http://www.telecomabc.com/l/lac.html)

3.4.10 <rla>

Receive level access minimum

GUESS: Minimum receiving level permitted to access the system Per: similar AT engineering mode (AT+QENG) command in [M95 AT commands manual](http://eddywireless.com/yahoo_site_admin/assets/docs/M95_AT_Commands_Manual_V12.196112248.pdf)

3.4.11 <txp>

Transmit power maximum CCCH

3.4.12 <TA>

[Timing Advance](https://en.wikipedia.org/wiki/Timing_advance)

3.5 health_check.log


```
{"cpu_times":
  {"iowait": 4694.23,
   "idle": 3089452.32,
   "user": 1786751.62,
   "system": 125489.34},
 "data_vol": 5.5,
 "root_vol": 5.5,
 "cpu_percent": [42.0, 53.0, 35.9, 38.0],
 "mem":
   {"swap_percent_used": 0.0,
    "free": 464707584},
 "queue_sizes": {
   "arfcn_correlator": 0,
   "geo_correlator": 0,
   "scan_results": 0,
   "cgi_correlator": 0},
 "application_uptime_seconds": 32461,
 "event_timestamp": "2017-05-07T06:32:09.816725",
 "scan_program": "health_check"}
```


The frequency with which these events are generated is determined by the `HEALTH_CHECK_INTERVAL` environment variable.

How is this information useful?

If you notice a trend where a metric under "queue_sizes" is always-increasing, you may have a failed processing thread. Correlate this with the events coming from heartbeat.log. Look for the absence of a heartbeat event for the corresponding thread). If you've confirmed that a thread has failed, the fastest fix is to just restart the sensor. If you can get a traceback for the thread failure, please submit it as an issue at https://github.com/sitch-io/sensor/issues/new.

3.6 heartbeat.log


```
{"heartbeat_service_name": "MainThread", "event_timestamp": "2017-05-07T06:32:09.
↪815061", "scan_program": "heartbeat"}
{"heartbeat_service_name": "kalibrate_consumer", "event_timestamp": "2017-05-
↪07T06:32:09.815243", "scan_program": "heartbeat"}
{"heartbeat_service_name": "arfcn_correlator", "event_timestamp": "2017-05-
↪07T06:32:09.815323", "scan_program": "heartbeat"}
{"heartbeat_service_name": "decomposer", "event_timestamp": "2017-05-07T06:32:09.
↪815391", "scan_program": "heartbeat"}
{"heartbeat_service_name": "gsm_modem_consumer", "event_timestamp": "2017-05-
↪07T06:32:09.815456", "scan_program": "heartbeat"}
{"heartbeat_service_name": "geoip_consumer", "event_timestamp": "2017-05-07T06:32:09.
↪815520", "scan_program": "heartbeat"}
{"heartbeat_service_name": "writer", "event_timestamp": "2017-05-07T06:32:09.815584",
↪"scan_program": "heartbeat"}
{"heartbeat_service_name": "geo_correlator", "event_timestamp": "2017-05-07T06:32:09.
↪815648", "scan_program": "heartbeat"}
{"heartbeat_service_name": "gps_consumer", "event_timestamp": "2017-05-07T06:32:09.
↪815711", "scan_program": "heartbeat"}
{"heartbeat_service_name": "cgi_correlator", "event_timestamp": "2017-05-07T06:32:09.
↪815780", "scan_program": "heartbeat"}
```


These events are most useful when chasing down thread failure. It doesn't happen often, but when it does, you can look at these events as a time-series and see where one ceases to appear. This is most useful when correlated with queue sizes as reflected in health_check.log.

3.7 kal_channel.log


```
{"site_name": "sitch-site-testing",
 "power": 854930.16,
 "final_freq": "874979084",
 "band": "GSM-850",
 "scan_finish": "2017-05-07T06:28:38.545421",
 "event_timestamp": "2017-05-07T06:28:38.545421",
 "sample_rate": "270833.002142",
 "gain": "80.0",
 "scanner_public_ip": "1.1.1.1",
 "scan_start": "2017-05-07T06:23:39.482440",
 "scan_program": "kalibrate",
 "arfcn_int": 157,
 "channel": "157"}
```

3.8 scanner.log


```
{"site_name": "sitch-site-testing",
 "scan_results": [
 {"channel_detect_threshold": "105949.217083",
  "power": "854930.16",
  "final_freq": "874979084",
  "mod_freq": 20916.0,
  "band": "GSM-850",
  "sample_rate": "270833.002142",
  "gain": "80.0",
  "base_freq": 875000000.0,
  "device": "0: Generic RTL2832U OEM",
  "modifier": "-",
  "channel": "157"}
  ],
 "platform": "Unspecified",
 "scan_start": "2017-05-07T06:23:39.482440",
 "scan_location": "sitch-site-testing",
 "scanner_public_ip": "1.1.1.1",
 "scan_finish": "2017-05-07T06:28:38.545421",
 "event_timestamp": "2017-05-07T06:28:38.545421",
 "scan_program": "kalibrate",
 "scanner_name": "sitch-site-testing"}
```


The list of items under scan_results is used by the Decomposer to produce messages that end up in the kal_channel log file.

3.9 sitch_alert.log


```
{"details": "Primary BTS was 310:260:275:20082 now 310:260:275:42302. Site: sitch-
↪site-testing",
 "type": "Primary BTS metadata change.",
 "id": 110,
 "device_id": "sitch-site-testing"
 "event_timestamp": "2017-05-07T06:28:38.545421"}
```


- `details` is a human-readable representation of the event, with details.
- `type` is a human-readable description of the alert type. For a list of supported event types, look in the `__init__` section of http://sensor.readthedocs.io/en/test/_modules/sitchlib/alert_manager.html#AlertManager
- `id` is an ID that maps to a specific event type. This is meant to simplify integration with SIEM and log management systems.
- `device_id` is the device ID (see device configuration environment vars)
- `event_timestamp` is generated when the alert is detected.

3.10 sitch_init.log


```
{"evt_data": "T-Mobile",
 "evt_type": "registration",
```


(continues on next page)

(continued from previous page)


```
 "evt_cls": "gsm_consumer",
 "event_timestamp": "2017-05-06T06:25:49.837957"}

{"evt_data": "\r\n | OK\r\n | ATV1Q0&V \r\r\n | DEFAULT PROFILE\r\n | S0: 0\r\n | S3:␣
↪13\r\n | S4: 10\r\n | S5: 8\r\n | S6: 2\r\n | S7: 60\r\n | S8: 2\r\n | S10: 15\r\n␣
↪| +CRLP: 61,61,48,6\r\n | V: 1\r\n | E: 1\r\n | Q: 0\r\n | X: 4\r\n | &C: 1\r\n | &
↪D: 1\r\n | +CLTS: 0\r\n| +CREG: 0\r\n | +CGREG: 0\r\n | +CMEE: 0\r\n | +CIURC:␣
↪1\r\n | +CFGRI: 2\r\n | +CMTE: 0\r\n | +CANT: 0,0,10\r\n | +STKPCIS: 0\r\n | +CMGF:␣
↪0\r\n | +CNMI: 2,1,0,0,0\r\n | +CSCS: \"IRA\"\r\n | +VTD: 1\r\n | +CALS: 1\r\n |␣
↪+CHF: 0\r\n | +CAAS: 1\r\n | +CBUZZERRING: 0\r\n | +DDET: 0\r\n | +MORING: 0\r\n |␣
↪+SVR: 16\r\n | +CCPD: 1\r\n | +CSNS: 0\r\n | +CSGS: 1\r\n | +CNETLIGHT: 1\r\n |␣
↪+SLEDS: 64,64,64,800,3000,300\r\n | +CSDT: 0\r\n | +CSMINS: 0\r\n | +EXUNSOL: 0\r\n␣
↪| +FSHEX: 0\r\n | +FSEXT: 0\r\n | +IPR: 0\r\n | +IFC: 0,0\r\n | +CSCLK: 0\r\n |␣
↪\r\n | USER PROFILE\r\n | S0: 0\r\n | S3: 13\r\n | S4: 10\r\n | S5: 8\r\n | S6:␣
↪2\r\n | S7: 60\r\n | S8: 2\r\n | S10: 15\r\n | +CRLP: 61,61,48,6\r\n | V: 1\r\n |␣
↪E: 1\r\n | Q: 0\r\n | X: 4\r\n | &C: 1\r\n | &D: 1\r\n | +CLTS: 0\r\n | +CREG:␣
↪0\r\n | +CGREG: 0\r\n | +CMEE: 0\r\n |+CIURC: 1\r\n | +CFGRI: 2\r\n | +CMTE: 0\r\n␣
↪| +CANT: 0,0,10\r\n | +STKPCIS: 0\r\n | +CMGF: 0\r\n | +CNMI: 2,1,0,0,0\r\n |␣
↪+CSCS: \"IRA\"\r\n | +VTD: 1\r\n | +CALS: 1\r\n | +CHF: 0\r\n | +CAAS: 1\r\n |␣
↪+CBUZZERRING: 0\r\n | +DDET: 0\r\n | +MORING: 0\r\n | +SVR: 16\r\n | +CCPD: 1\r\n |␣
↪+CSNS: 0\r\n | +CSGS: 1\r\n | +CNETLIGHT: 1\r\n | +SLEDS: 64,64,64,800,3000,300\r\n␣
↪| +CSDT: 0\r\n | +CSMINS: 0\r\n | +EXUNSOL:0\r\n | +FSHEX: 0\r\n | +FSEXT: 0\r\n |␣
↪+IPR: 0\r\n | +IFC: 0,0\r\n | +CSCLK: 0\r\n | \r\n | ACTIVE PROFILE\r\n | S0: 0\r\n␣
↪| S3: 13\r\n | S4: 10\r\n | S5: 8\r\n | S6: 2\r\n | S7: 60\r\n | S8: 2\r\n | S10:␣
↪15\r\n | +CRLP: 61,61,48,6\r\n | V: 1\r\n | E: 1\r\n | Q: 0\r\n | X: 4\r\n | &C:␣
↪1\r\n | &D: 1\r\n | +CLTS: 0\r\n | +CREG: 0\r\n | +CGREG: 0\r\n | +CMEE: 0\r\n |␣
↪+CIURC: 1\r\n | +CFGRI: 2\r\n | +CMTE: 0\r\n | +CANT: 0,0,10\r\n | +STKPCIS: 0\r\n␣
↪| +CMGF: 0\r\n | +CNMI: 2,1,0,0,0\r\n | +CSCS: \"IRA\"\r\n | +VTD: 1\r\n | +CALS:␣
↪1\r\n | +CHF: 0\r\n | +CAAS: 1\r\n | +CBUZZERRING: 0\r\n | +DDET: 0\r\n | +MORING:␣
↪0\r\n | +SVR: 16\r\n | +CCPD: 1\r\n | +CSNS: 0\r\n | +CSGS: 1\r\n | +CNETLIGHT:␣
↪1\r\n | +SLEDS: 64,64,64,800,3000,300\r\n | +CSDT: 0\r\n | +CSMINS: 0\r\n |␣
↪+EXUNSOL: 0\r\n | +FSHEX: 0\r\n | +FSEXT: 0\r\n | +IPR:0\r\n | +IFC: 0,0\r\n |␣
↪+CSCLK: 0\r\n | \r\n | OK\r\n",
"evt_type": "device_config",
"evt_cls": "gsm_consumer",
"event_timestamp": "2017-05-06T06:25:49.837957"}

{"evt_data": "IMSI_GOES_HERE",
"evt_type": "sim_imsi",
"evt_cls": "gsm_consumer",
"event_timestamp": "2017-05-06T06:25:49.837957"}
```


These messages are only generated when the application starts.

- `registration` records the cell service provider, according to the GSM modem.
- `device_config` dumps the profiles in use from the GSM modem.
- `sim_imsi` records the IMSI from your cell modem's SIM card.

CHAPTER 4

Event Lifecycle

The lifecycle of an event in SITCH begins in the Sensor, and ends with the user's (or alert management system's) consumption. We'll follow the most frequent event, the GSM modem scan event.

4.1 Ingestion

The Sensor runs the gsm_modem_consumer() function as a thread in runner.py. This thread produces events from the output of the GSM modem being in engineering mode. gsm_modem_consumer() wraps the GsmModem class (found in gsm_modem.py), takes the output from the __iter__() in GsmModem, and places it into the `scan_results_queue` FIFO buffer.

4.2 Decomposition

The decomposer() function in runner.py is also run in a thread, and as scan results are placed into the `scan_results_queue` FIFO, it pulls them out and decomposes them into individual events, one for each cell. Copies of these decomposed events (labeled `gsm_modem_channel`) are placed into the `cgi_correlator_queue`, `arfcn_correlator_queue`, and `message_write_queue` FIFO buffers.

4.3 Correlation

The cgi_correlator() and arfcn_correlator() functions are run in threads and consume events from the `cgi_correlator_queue` and `arfcn_correlator_queue` FIFO buffers, respectively. The cgi_correlator() correlates the information contained in the event with the feed information based on the OpenCellID database, taking the geolocation of the sensor into account. If any alarms are produced, they are placed in the `message_write_queue`. The arfcn_correlator() function compares the ARFCN in the event metadata with information contained in the feed based on the FCC license database, taking into account the geolocation of the sensor.

4.4 Transmission

The output() function is run in a thread and listens for events being placed into the `message_write_queue` FIFO. It takes the events it finds there and writes them to disk, appending them to files by event type.

At this point, you have the original scan event, each decomposed channel event, and any alerts produced, logged on disk in specific files, based on event type.

These events are picked up from disk by filebeat, and transmitted to Logstash, which runs in the service side of SITCH.

4.5 Reception

Logstash splits the information between two data stores. The events themselves get sent to Elasticsearch, and you can query them with Kibana. Some of the measurement metadata is sent to influxDB, and can be viewed with Chronograf.

Events with type `sitch_alert` are sent to Slack by Logstash.

CHAPTER 5

Sensor Troubleshooting

5.1 GSM modem device detection

If you're using a GSM modem that's not recognized by the device detector, please add the output from running the `ATI` command against your GSM modem in the variable named `positive_match` in the `is_a_gsm_modem()` ` method, in the `sensor/sitch/sitchlib/device_detector.py` file. Then send a pull request so that everyone can get the benefit of your discovery.

You can do this using the resin.io terminal on the device by doing the following steps.

1. Set the environment variable `GSM_MODEM_BAND` to `nope` to disable the scanner.
2. Identify which TTY port your device is running on. You can find this in the startup logs under the string `DeviceDetector: Detected USB devices`.
3. Run python from the sensors virtual environment

```
/app/sitch/venv/bin/python
```

4. Create a serial connection to the GSM modem.

```
> import serial
> port = '/dev/[THE_MODEMS_TTY_SYS_NAME]'
> serconn = serial.Serial(port, 4800, timeout=1)
```

5. Run the following snippet to get the string you need.

```
> test_command = "ATI \r\n"
> serconn.flush()
> for i in xrange(10):
> line = None
> line = serconn.readline()
> if line is None:
>     time.sleep(1)
>     pass
```

(continues on next page)

(continued from previous page)

```
> else:
>     print("Use this GSM Modem String in your pull request: {0}".format(line))
> serconn.flush()
> serconn.close()
```

5.2 Found but undetected TTY

The DeviceDetector shows it found my GSM Modem or GPS Device by the Configurator does not detect it

5.2.1 How to identify if this is your issue

You will be able to recognize this issue if three conditions are met.

1. You are receiving an error that the device is not configured or cannot bind to its socket.
2. Your Configurator returns an empty array instead of a USB-TTY device name when it attempts to detect a device.
3. Your device detector is detecting these devices

If the device detector cannot find the devices, as the following log message shows, then *this is not your issue.*

5.2.2 How to fix this issue

To fix this issue you can set the hard-coded environment variable for the device that is not detected.

In the following example the GSM modem is not detected.

```
> 22.04.17 08:53:27 (-0400) Configurator: Detected GSM modems:
> 22.04.17 08:53:27 (-0400) []
> 22.04.17 08:53:27 (-0400) Configurator: Detected GPS devices:
> 22.04.17 08:53:27 (-0400) [u'/dev/ttyUSB0']
```

This shows me that the GSM modem was not detected and that my GPS device can be found at '/dev/ttyUSB0'.

By looking at my DeviceDetector I can see that I have two USB devices connected. It also gives me the 'sys_name' of each device.

Since I know that my GPS device has a sys_name of `ttyUSB0` I know that the sys_name GSM device is `ttyUSB1`.

I can now set the `GSM_MODEM_PORT` environment variable to point to /dev/ttyUSB1 in the resin.io `Environment Variables` interface.

(NOTE: for those unfamiliar with python strings it should be noted that the `u` in front of each quoted value in these example logs is specifying that the string is a Unicode string. You do not want to enter the `u` in front of /dev/ttyUSB1 when setting your environment variables.)

If you have successfully set the environment variable you will no longer receive an error message.

In the case of the GSM modem you will also see that the following message has replaced the original error.

5.3 GPS device not detected (U-Blox7)

The U-Blox7 USB GPS device registers as a ttyACM device. If everything else (with respect to the sensor hardware stack) is built to spec, the U-Blox7 GPS will land at `/dev/ttyACM0`. Set the `GSM_MODEM_PORT` Sensor environment variable in resin.io to `/dev/ttyACM0`. The application on the sensor will then restart. Open the terminal in resin.io and `tail -f /data/sitch/log/gps.log` to confirm that the GPS is correctly configured and able to get a location fix. You may have to wait for a few minutes. If this does not work, you can also use the terminal to run `gpsmon` to see if gpsd is able to communicate with the device.

5.4 No events in Kibana

The SITCH sensor relies on Filebeat to read events from log files and transmit them to the Logstash instance running in the SITCH service. There are a few indicators when the transmission process is broken:

1. Confirm that log files are being written:

- Confirm that log files are being written at `/data/sitch/log/`. If your sensor isn't populating log files, the most likely reason is that the sensor is in a reboot loop due to mis-configuration.
- Check the Device Summary page in Resin, for the affected sensor. If the reason that the sensor isn't coming online cleanly isn't celarly explained in the log messages, please reach out in the gitter channel (https://gitter.im/sitch-io/sensor) or open an issue in the sensor project on Github: https://github.com/sitch-io/sensor/issues

2. Make sure that the filebeat process is running on the sensor:

- Check using `ps ef` from the terminal: you should see a line containing: `/usr/local/bin/filebeat-linux-arm -c /etc/filebeat.yml`. If you don't, you can try to start the process manually and look for errors printed to stdout.
- Your crypto certs and keys are retieved in the sensor initialization process and dropped at `/host/run/dbus/crypto/`. You should see three files there: `ca.crt`, `logstash.crt`, and `logstash.key`. If you don't have those files on your system, there's a really good chance that your sensor environment variables aren't set correctly.
- You should confirm that the `VAULT_PATH`, `VAULT_TOKEN`, and `VAULT_URL` environment variables are correct, and that the network path is open between your sensor and Vault.
- You can confirm the network path is open by running this command: `openssl s_client -connect VAULT_HOSTNAME:8200`. Replace `VAULT_HOSTNAME` with the DNS name from the output of `echo $VAULT_URL`, when run in the terminal on the sensor. So if your $VAULT_URL is `https://myvault.mydomain.com:8200`, the command you should run in the terminal on the sensor is: `openssl s_client -connect myvault.mydomain.com:8200`.
- An error message containing `gethostbyname failure` indicates a failure in DNS resolution.
- A message containing `connect: Connection refused` indicates that the OpenSSL client is unable to connect to the port that Vault is running on, and you need to check your iptables and security groups settings, and confirm that Vault is actually listening on that port.
- You should also confirm that Vault is actually running.
- If the Vault container is stopped and restarted, you will need to unseal the Vault again. See the docs for the demo environment (https://github.com/sitch-io/demo) for details on how to unseal the vault.

3. Confirm that Filebeat is processing the log files:

- There's a registry file managed by Filebeat, located at `/data/sitch/log/fb_registry`. This file is what Filebeat uses to maintain its place in your log files, in the event it gets restarted. If this file is empty, confirm that the network path to Logstash is open by running this command: `openssl s_client -connect`

`${LOG_HOST}` If the connection times out, you should take a hard look at your network ACLs and iptables rules.

4. If Filebeat appears to be transmitting events to Logstash and you still don't see events in Kibana, you can run the logstash container in debug mode by changing the docker-compose.yml file's setting for `services.logstash.image` from `docker.io/sitch/logstash` to `docker.io/sitch/logstash:debug`. Then, use docker-compose to take your stack down and back up again. This will be very verbose, and can cause a substantial amount of disk space consumption. Don't leave it like that forever.
5. If there is no indication that Logstash is having trouble getting events into Elasticsearch, check that you have an index for logstash built in Kibana.

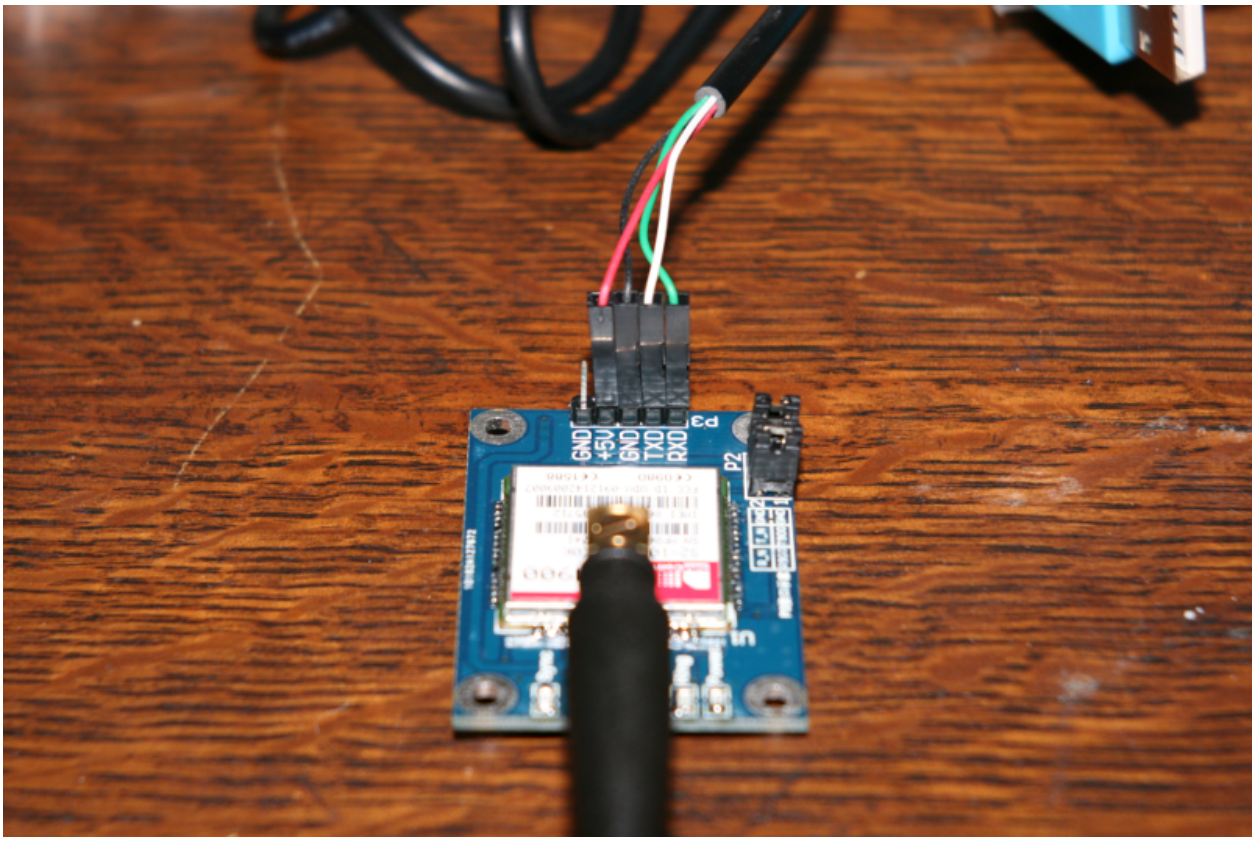
- Navigate to this URL: https://MY_SITCH_SERVICE_HOSTNAME:8443/app/kibana#/management/kibana/indices , replacing MY_SITCH_SERVICE_HOSTNAME with the hostname of your SITCH service-side environment. If you have an index, you should have events.
- Try adjusting your time window, and confirm that the system clocks in your SITCH service side components are correct.
- Time drift can not only cause the query in Kibana to look weird, but it can cause an SSL connection negotiation failure between the sensor and service.

If none of the above lead you to success, please don't hesitate to file an issue in the sensor's Github repository: https://github.com/sitch-io/sensor/issues and/or reach out in the Gitter channel: https://gitter.im/sitch-io/sensor.

CHAPTER 6

Reference Images

6.1 Connecting the USB TTY cable to the SIM 900 GSM modem



CHAPTER 7

SITCH Sensor Internal API

7.1 AlertManager

7.2 ArfcnCorrelator

7.3 CgiCorrelator

7.4 ConfigHelper

7.5 Decomposer

7.6 DeviceDetector

7.7 FeedManager

7.8 GeoCorrelator

7.9 GeoIp

7.10 GeoipDecomposer

7.11 GpsDecomposer

7.12 GpsListener

7.13 GsmDecomposer

7.14 GsmModem

CHAPTER 8

Indices and tables

- genindex
- modindex
- search