# Vilant Node Module Documentation

App version 1.1.0.f740a699

# Table of Contents

# Introduction

# Module

The logic for the Vilant Node application is implemented using connected modules. Individual modules can perform actions, such as generate new data, filter or buffer flowing data, control I/O devices connected to the Vilant Node application, or send data to external services. Modules can also control processing triggers for other modules.

Each of the modules have four common properties:

1) **Name** is unique identifier for a module. Name size is limited to 20 characters.

2) **Type** defines the module type. Each module type has its own characteristics and set of parameters.

3) **Log level** is useful in problematic scenarios which require more information how the module works. Increasing log level will increase the verbosity of the module in the internal logging. See log page for log events from Vilant Node application.

4) **Engineering notes** are notes that engineers can add in the configuration to help documenting installation. These will have no functional effect on Node process, so the purpose is documentation only.

# Classification of Modules

1) **Source modules** bring new data or trigger events to module system. Examples of such modules are INPUT and RFID_READER modules.

2) **Output modules** take data out of system and perform action external service, such as I/O or backend systems. Examples of such modules include EVENTSENDER, STORE, OUTPUT and MQTT modules.

3) **Trigger modules** are controlled only by trigger actions. No data is being passed. Examples of such modules include INPUT, OUTPUT, DELAY, TIMEOUT and logic modules AND, and OR.

4) **Filter modules** can filter data stream to pick only relevant data for further processing in next modules. If the data fails the condition, the data will be sent to failing chain. Examples of such modules include EPCHEX, DUPLICATE, ANTENNA, RSSI and MIN_READCOUNT modules.

5) **Data transformation modules** can be used to change data during processing. In some cases, there can be a need to normalize RFID tag EPCs for further processing. For instance, if a handling unit contains multiple different tags representing same asset, these can be normalized into a single entity in the further processing. Examples of such modules include EPCMANGLE module.

6) **Computational modules** will perform calculations on the data it receives. These modules implement own logics that can generate new data and trigger events or filter data in addition to the stream they are receiving. Examples of such modules include DIRECTION and SEEN modules.

# Connections

The modules themselves are not enough to create a program. In order to create a program, modules need to be connected using connections. Connections will create logic paths, or a graph, that will control how the data flows in the program.

There are two types of connections:

1) Data Connections

2) Trigger Connections

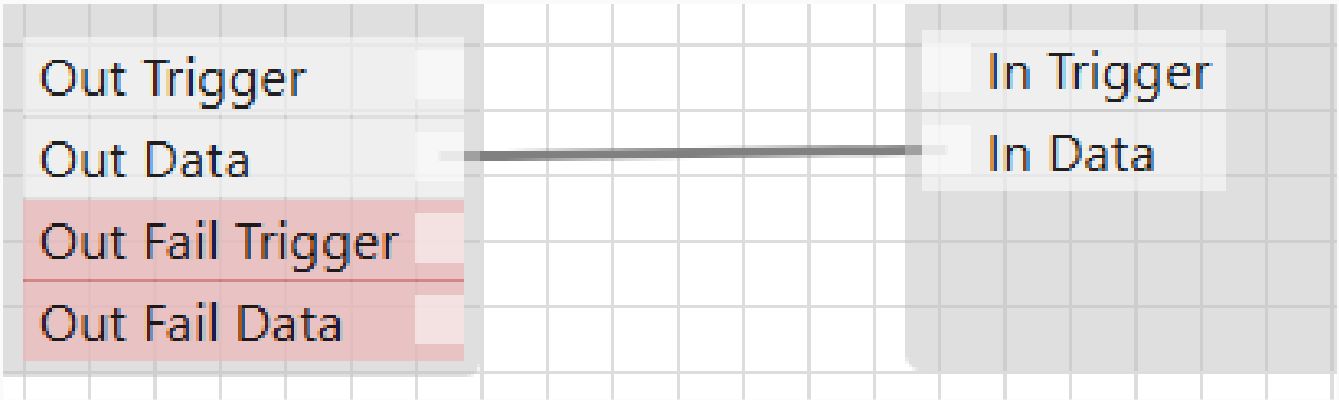
# DATA Connection

Data connections are used to create data flows. Data flow will forward the data from source to target module. Data can be flowing either from success or failure path, depending on the data processing result.

Things to consider with DATA connections:

1) Data processing in modules require trigger events via TRIGGER connections.

2) Not all modules will use DATA connections. In Vilant Node version 1, the usage is limited to i) Source, ii) Filtering, iii) Data Transformation, and iv) Computational modules.



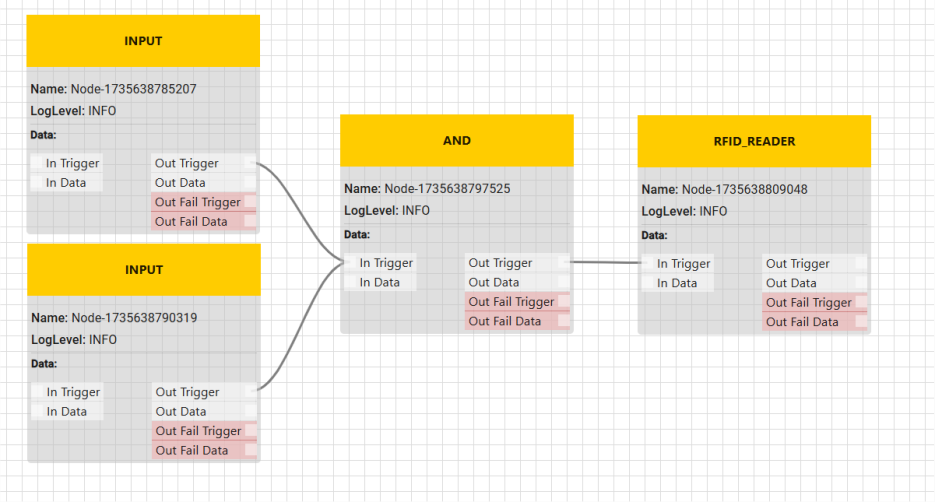
# TRIGGER Connection

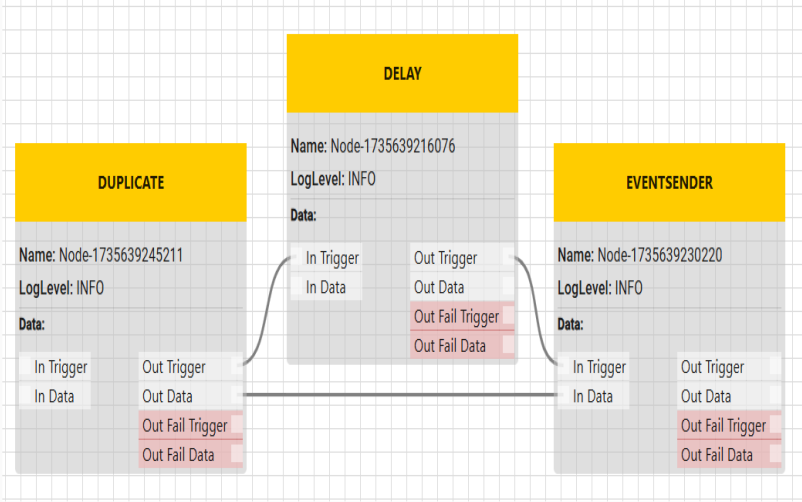
Trigger connections are used for triggering the processing in the modules. Trigger connections will emit trigger events requesting for processing of the data buffers on the triggered modules. By controlling triggering of modules, it is possible to create large variety of different logics.



# TRIGGER Control

In addition creating connections, trigger events can be controlled with modules. Such modules include logic modules (AND, OR) and timeout-based modules (DELAY, TIMEOUT). These modules allow changing the behaviour of the trigger which results in flexibility in usage of the trigger events.

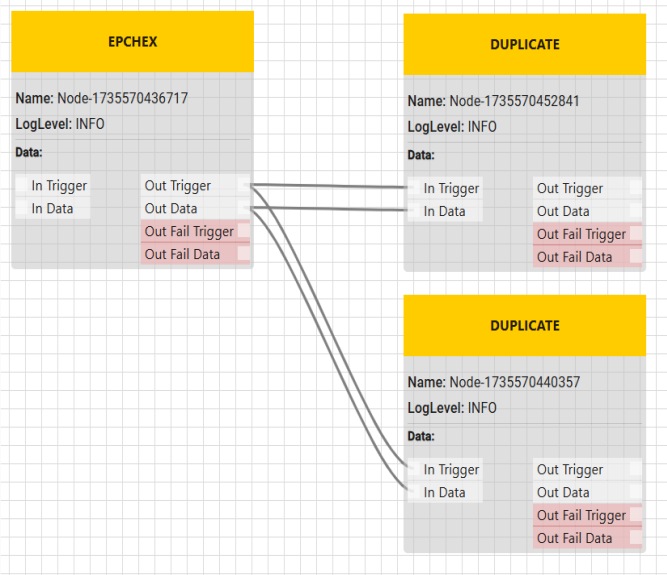


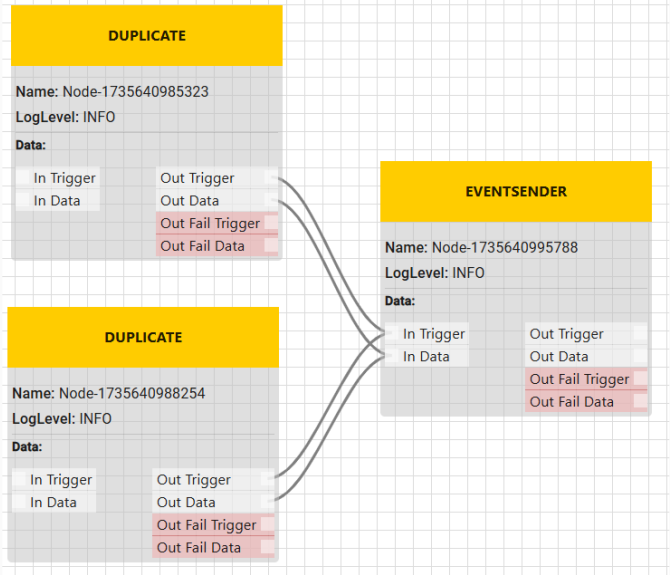



# Branching & Joining

One connection can lead into many other modules which will result data or trigger events being forwarded from one source to multiple targets. This can be effective in cases, such as, where different type of data requires different filtering configuration or data needs to be outputted in a different way. It is also possible to output same data to multiple systems.

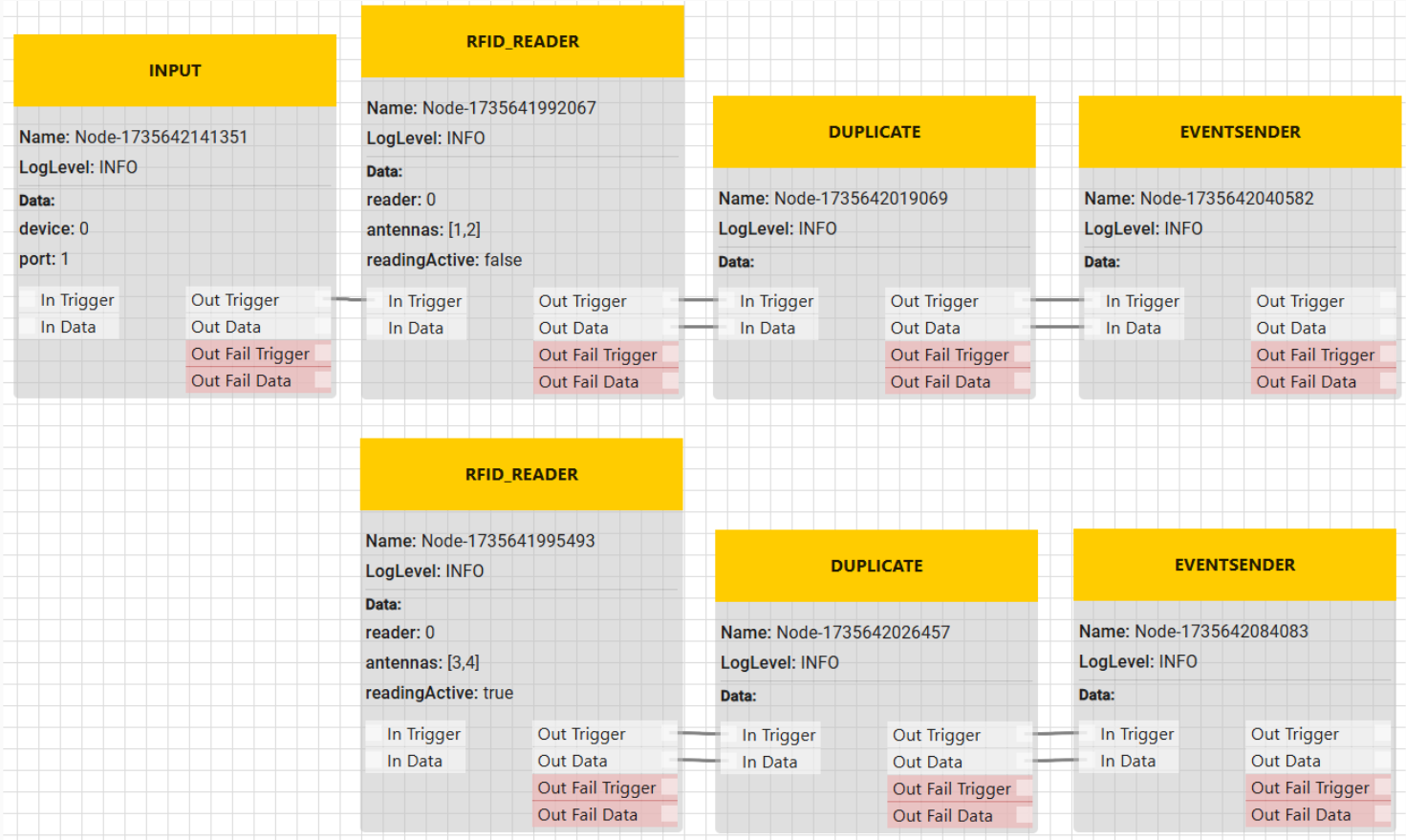
Connections can also join data and trigger events from many source to one target. For instance, it is possible to combine data from multiple filtering modules into a single output module.



# Parallel Logics

Vilant Node application allows to create multiple logic graphs working simultaneously. There are no limitations for the number of parallel logic graphs. For instance, one graph can perform operations on data from RFID antennas 1 and 2, while the other graph will perform operations on antennas 3 and 4.



# Module Descriptions

# RFID_READER module

The RFID_READER module can be used as a source for RFID tag read events. RFID tag read events are received from the selected antennas that are configured in the RFID_READER module configuration. Module can listen to one or more antennas from a single RFID reader.

RFID_READER module is triggering next modules whenever it receives new events from the RFID reader. The data is being forwarded only from success path.

Triggering RFID_READER module will control RFID reading state for the antennas used by the module. Trigger with ON (1) event will start RFID reading and OFF (0) event will stop RFID reading. For instance, INPUT module can be used to trigger RFID reading. If using triggered reading, set *readingActive* parameter to false to disable RFID reading from starting.

RFID_READER module can be configured to start RFID reading after the initialization of the modules. This is useful in use cases which require RFID reading to be always active.

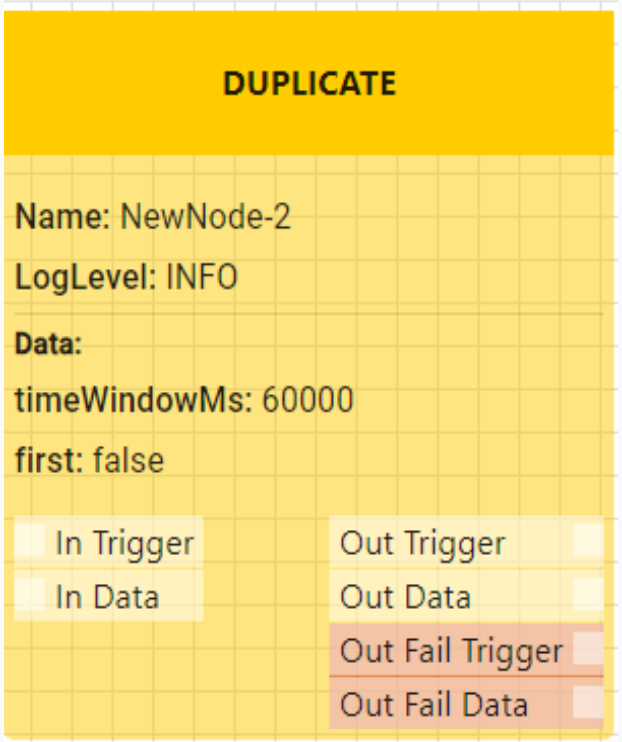
| Name | Type | Description | Required |
|---|---|---|---|
| reader | integer (Only ID 0 allowed for now) | Unique id for the reader | X |
| antennas | array of integers | Antennas used by the module. Module will accept events only from the specified antennas. Accepted values for antennas are 1 - 32. | X |
| readingActive | integer (Only ID 0 allowed for now) | Starting state for antennas. Default is true (= constant reading) | |

# DUPLICATE Module

Duplicate filters tags based on 1) time window or 2) count of unique tags.

In **time window** mode, filtering will pass RFID tag for the first time it sees the tag and keep filtering out the RFID tag until the tag has disappeared for more than *timeWindowMs* time. The extension, **first**, will let tag pass first time and again when *timeWindowMs* has passed since last time tag passed filtering.

In **count** mode, filtering will hold memory of *count* number of RFID tags and will not pass RFID tags it holds in memory. When the *count* is exceeded, the oldest RFID tags are erased from memory and become readable again.




| Name | Type | Description | Required |
|---|---|---|---|
| count | integer (0 - 2147483647) | Duplicate buffer. If the maximum number of tags is exceeded, oldest tags are removed from the first. | Either "count" or "timeWindowMs" is required. |
| timeWindowMs | integer (0 - 2147483647) | Same tag is filtered until "timeWindowMs" amount of time. Every new read will restart timewindow. Value should be given in milliseconds. | Either "count" or "timewindow" is required. |
| first | boolean | Extra mode for "timeWindowMs" mode. If enabled, timestamp is taken from first time tag is seen and not updated with new events until "timeWindowMs" amount of time has passed. Disabled by default. | |

# EVENTSENDER (Pro) Module 1/2

The EVENTSENDER module sends RFID tag events to a specified endpoint using HTTP/REST interface. The module will add new messages to the sender queue which will handle sending of messages. Sender can be configured with resending mechanism in case of errors while sending. Messages in the queue are visible in *send queue* page.

The message is sent to the endpoint with HTTP POST method. The message structure follows the TVS REST/JSON asset interface:


```
{
"assetOperations": [
{
"id": {
"primary": "30180000000000000000001"
},
"operation": "MOVE",
"currentLocationSite": "site1",
"currentLocation": "Loc1",
"deviceId": "Testdev-1735656736.6111813",
"timestamp": "2024-12-31T14:53:34.067Z"
}
],
"deviceId": "Testdev-1735656736.6111813",
"applicationVersion": "TEST-1.0.0.1cf5fe0f-TEST",
"deviceModelId": "VilantNode/Turck"
}
```

# EVENTSENDER (Pro) Module 2/2

| Name | Type | Description | Required |
|---|---|---|---|
| endpoint | string | Endpoint address to HTTP REST Server including protocol, credentials, hostname and path. | X |
| operation | string | Operation used for asset event. Can be any but if sending to TVS servers, then need to follow interface specification. If undefined or empty, then defaults to MOVE. | |
| site | string | Site parameter for the TVS asset event. Describes higher level detail of location. | |
| username | string | HTTP basic authentication username. | |
| password | string | HTTP basic authentication password. | |
| locations | object | Define location to antennas mapping. Keys object properties is the location name and value is the antenna. | X |
| locations.{key} | string | Key value of the property defines location name. | X |
| locations.{value} | array of integers | Value of the property defines antenna numbers for the location defined in key field. Allowed antenna values are 1 - 32. | X |

# STORE Module 1/2

Stores tag events that can be collected with the HTTP REST interface. The module collects tag events, meaning that the same tag can be stored multiple times. Supports either time window based or count based mode.

| Name | Type | Description | Required |
|---|---|---|---|
| timeWindowMs | integer (1 - 2147483647) | "timeWindowMs" field defines how long tag event is accessible by STORE module. Tag event will be dropped after expiration. | X |
| keep | boolean | "keep" field defines whether a call to HTTP interface would remove tag event or should module keep it in memory until expiration. Defaults to false. | |
| count | integer (1 - 2147483647) | "count" field defines how many tag events can be stored by module. | |

# STORE Module 2/2

Supported HTTP/REST GET endpoints:

1) All STORE modules: http://<host>:<port>/rfid/store

2) Specific STORE module: http://<host>:<port>/rfid/store/<module_name>

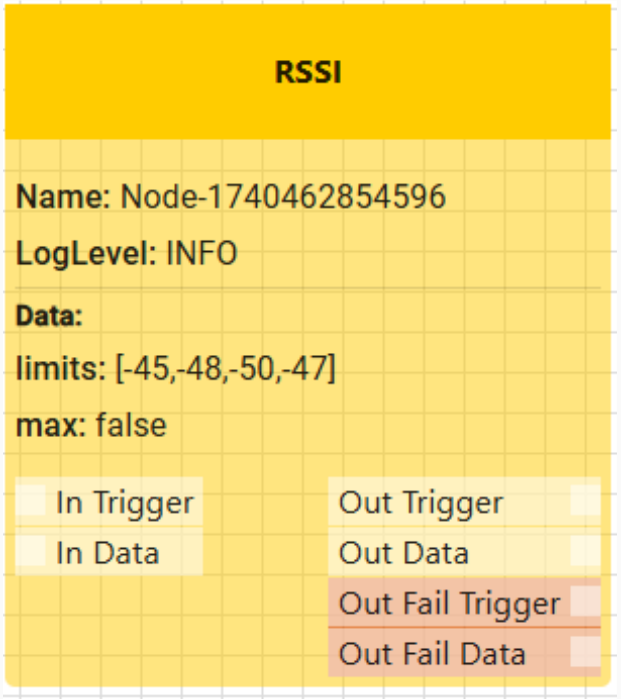
Example of response to HTTP/REST GET requests:


```
{
"tags": {
"<firstmodule_name>": [
{
"EPC": "300C69F6BC7115D9DEBD01C7",
"URI": "urn:epc:id:sgtin:111111111.1111.111111111111",
"readerid": 0,
"readcount": 1,
"rssi": -39,
"antenna": 1,
"phase": 3732,
"timestamp": "2023-08-23T07:40:05.520Z"
}
],
"<secondmodule_name>": []
}
}
```

# RSSI Module

The RSSI module is used to filter RFID tag events based on the RSSI (Received Signal Strength Indicator) value. The module will pass the RFID tag event only if the RSSI value from a specific antenna is within the specified range.

The RSSI value is a negative integer value.



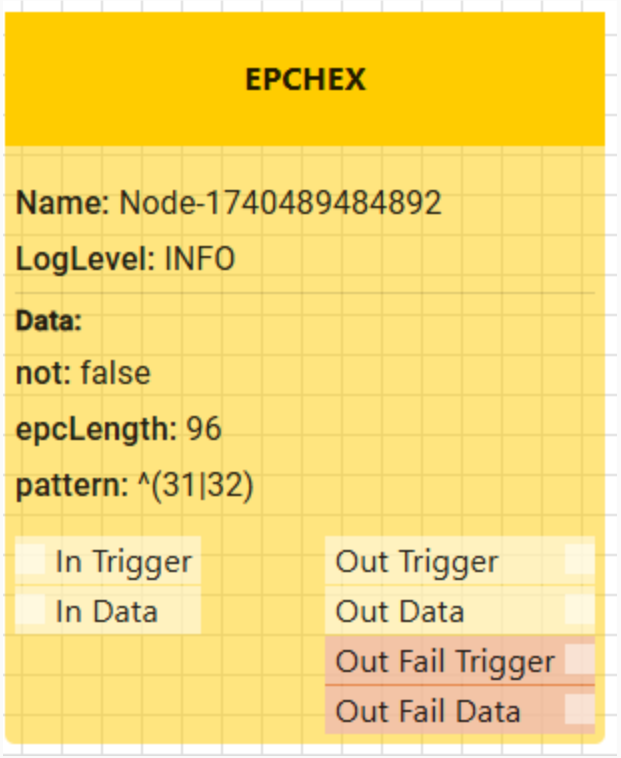

| Name | Type | Description | Required |
|---|---|---|---|
| limits | integer (-128 - 127) | Per antenna rssi limit. First value is required, if value is missing, first value is used | Yes |
| max | boolean | Common parameter for all antennas. If true, then tag is passed if antenna rssi is below limit. If false, then tag is passed if antenna rssi is above limit. Default is false. | Yes |

# EPCHEX Module

The EPCHEX module filters tag reads based on 1) EPC length or 2) EPC regular expression pattern.

Pattern should match the hex representation of tag EPC, or not match if *not* is set

Length parameter specifies expected length of tag EPC in bits




| Name | Type | Description | Required |
|---|---|---|---|
| pattern | String (regular expression) | Regular expression pattern to match tag EPC. If tag EPC matches the pattern, the tag is passed to the next module, unless 'not' is true. | Yes, either pattern or length is required. |
| not | boolean | Inverts the pattern match. If true, then tag is passed if EPC does not match the pattern. | Yes |
| length | integer | Checks EPC length in bits, if tag EPC length matches given length in bits, tag is passed to the next module. | Yes, either pattern or length is required. |

# DIRECTION (Pro) Module 1/2

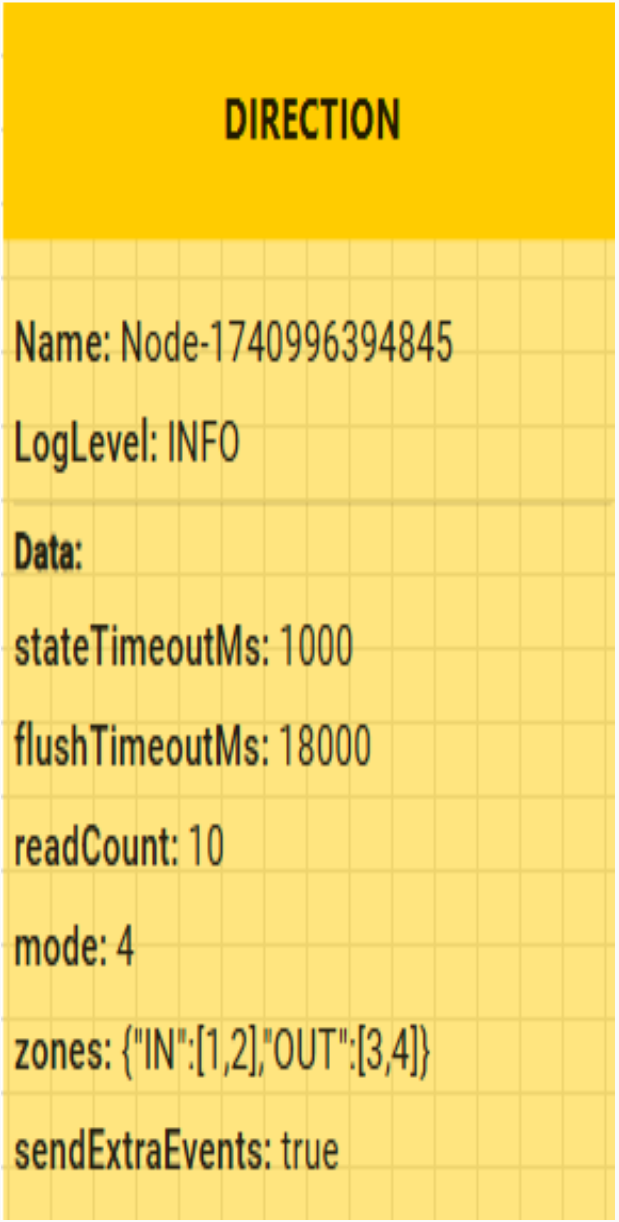
Module implementing direction detection. There are four modes:

- **MODE1 (1)** when the tag is not seen for 'stateTimeoutMs' msec, it is passed to the next module with the direction of the last zone.

- **MODE2 (2)** similar to MODE1. In addition, requires that first and last read events come from different zones. Or else UNKNOWN direction will be reported.

- **MODE3 (3)** Once tag has entered the second zone, a timeout of 'stateTimeoutMs' starts. At the end of the timeout, if the tag is still in the second zone, it is passed to the next module. If during the timeout, the tag comes back to the first zone, the timeout is stopped and nothing happens until the tag comes back to the second zone. After event has been sent the tag is still in waiting state and new event can be sent if tag changes zone. If noextra is specified then this will only sent single event per tag.

- **MODE3D (4)** Requires 'readCount' reads within 'stateTimeoutMs' from single zone without crossreads. Event is sent after two distinct zones are recognized. If noextra is specified then only single event will be sent.



# DIRECTION (Pro) Module 2/2

| Name | Type | Description | Required |
|---|---|---|---|
| mode | integer (1-4) | Module operating mode (see above) | Yes |
| stateTimeoutMs | integer (1-2147483647) | State timeout in milliseconds, depends on mode | Yes |
| flushTimeoutMs | integer (1-2147483647) | Flush timeout in milliseconds | Yes |
| zones | Object | Defines antenna & zone mapping. Key value defines the zone name and value defines a list of antennas representing the zone. Antennas can be assigned to only one zone each. Allowed antenna value range is 1 - 32. | Yes |
| readCount | integer (1-2147483647) | Required if using MODE3D. | Yes |
| sendExtraEvents | boolean | Used in modes 3 and 3D. Defaults to true. Controls whether to to send zone changes after first passing event. | No |
| useUserMemory | boolean | Tag is dropped if user memory is missing. | No |
| debug | boolean | Enable websocket debugging of direction detection. TODO how to access debug information? | No |

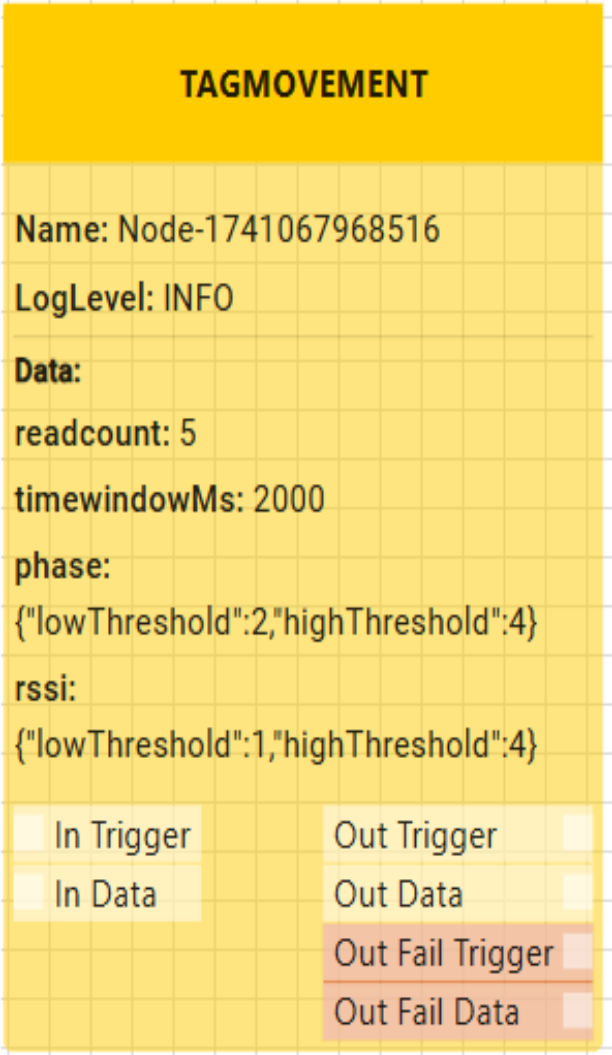
# TAGMOVEMENT (Pro) Module 1/2

Module calculating Mean Absolute Deviation (MAD) of rssi and/or phase values of tag reads and passing tags whose group charaster-istics are within thresholds to next module.

Tag reads are grouped by EPC and antenna. When new tag read enters a group, the group content is validated: Are there enough entries in the group considering the 'readcount' and 'timewindowMs' parameters? timewindowMs actually being the maximum age of group entries.

If the set of tagreads is valid, then the MAD of rssi and/or phase of the group is calculated. If the MAD is within the set thresholds, then the tag that triggered the MAD calculation is passed to the next module.

The user of this module must keep in mind that the decision of forwarding a tag read to next module is not based on its individual properties, but what the rssi/phase angle MAD properties of its group are.



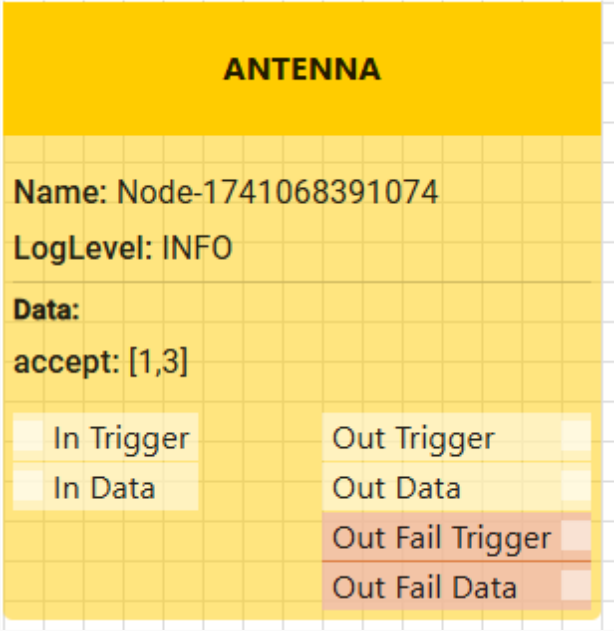
# TAGMOVEMENT (Pro) Module 2/2

## Tagmovement parameters:

| Name | Type | Description | Required |
|---|---|---|---|
| readcount | integer (1-1023) | Minimum size of reads in tag/antenna group, in order for MAD to be calculated. | Yes |
| timewindowMs | integer (1-2147483647) | Maximum age of read entry in group, from which rssi/phase MAD is calculated. | Yes |
| phase | object | Object containing phase angle MAD lower ad upper threshold values. If not set, phase MAD is not calculated. | Either phase or rssi object is required and both can be present |
| phase.lowThreshold | integer (0-127) | Low threshold for phase angle MAD. | Yes |
| phase.highThreshold | integer (0-127) | High threshold for phase angle MAD. | Yes |
| rssi | object | Object containing rssi MAD lower ad upper threshold values. If not set, phase MAD is not calculated. | Either phase or rssi object is required and both can be present |
| rssi.lowThreshold | integer (0-127) | Low threshold for rssi MAD. | Yes |
| rssi.highThreshold | integer (0-127) | High threshold for rssi MAD. | Yes |

# ANTENNA Module

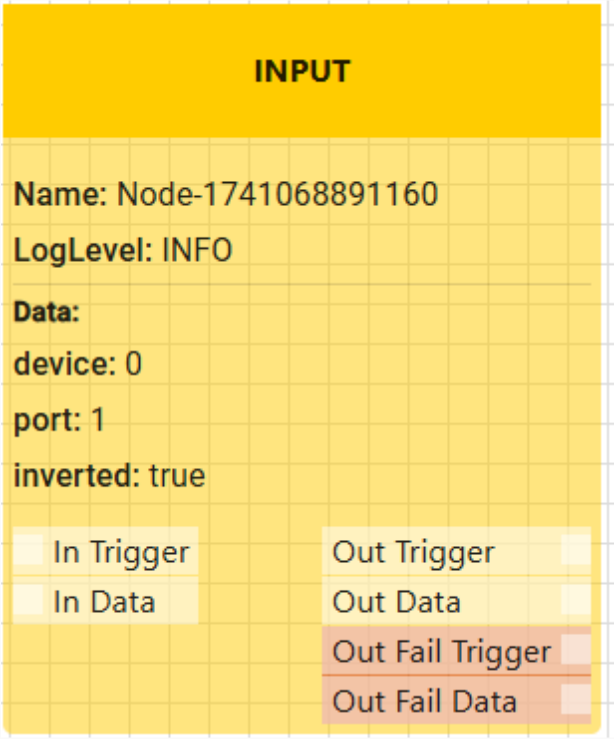
ANTENNA module forwards tags reads based on accepted antennas




| Name | Type | Description | Required |
|---|---|---|---|
| accept | array of integers | Antennas used by the module. Module accepts events only from the specified antennas. Accepted values for antennas are 1 - 32. | Yes |

# INPUT Module

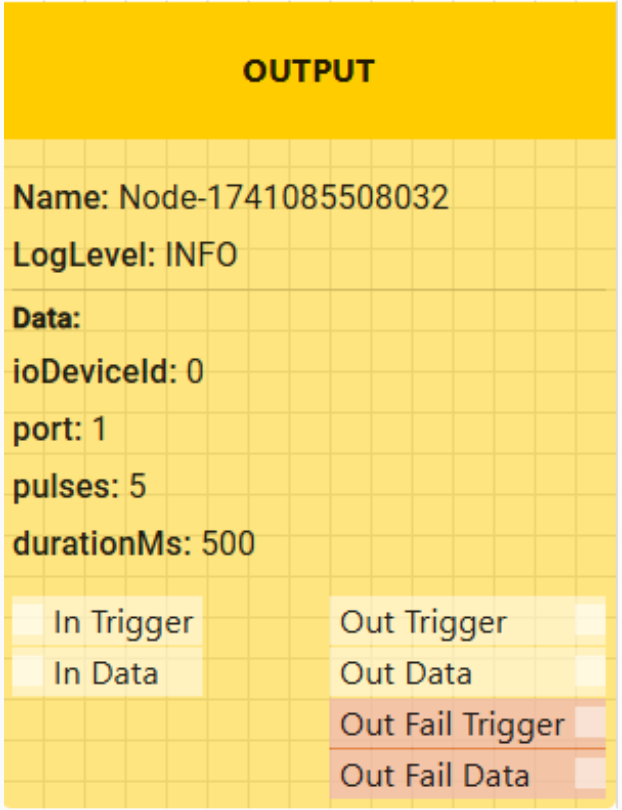
The INPUT module can be used to forward input events from a specific IO-device. Note! IO devices need to be configured separately.




| Name | Type | Description | Required |
|---|---|---|---|
| device | integer (0-8) | The target IO device ID | Yes |
| port | integer (0-31) | The target IO port | Yes |
| inverted | boolean | Inverts the input signal | No |

# OUTPUT Module

The OUTPUT module can be used for pulsing output events to a specific IO-device. Duration will configure the period for ON and OFF signals in pulse. Note! IO devices need to be configured separately.



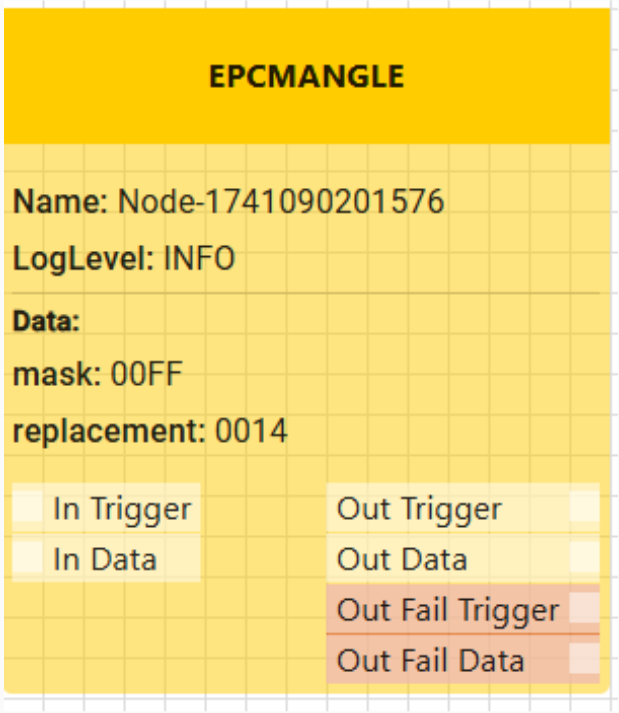

| Name | Type | Description | Required |
|---|---|---|---|
| ioDeviceId | integer (0-8) | The target IO device ID, defaults to internal device (0). | No |
| port | integer (0-31) | The target IO port. | Yes |
| pulses | integer (0-2147483647) | Defines the number of pulses to output port. One pulse includes 1 ON and 1 OFF signal, each lasting durationMs (total time 2 * durationMs). | Yes |
| durationMs | integer (1-2147483647) | Duration of the pulse in milliseconds, defines the duration of ON and OFF signals. Defaults to 500 ms. | No |

# EPCMANGLE Module

The EPCMANGLE module can be used to sanitize some parts of the EPC for the next module's use. The module is not modifying the EPC on the tag itself.

Note! Indexing starts from the beginning, so remember to pad both mask and replacement values with zeroes until the desired position.

Mask defines what bits are subject to mangling. 0 (binary 0000) matches nothing. F (binary 1111) matches all bits. Replacement defines new values for the selected bits.

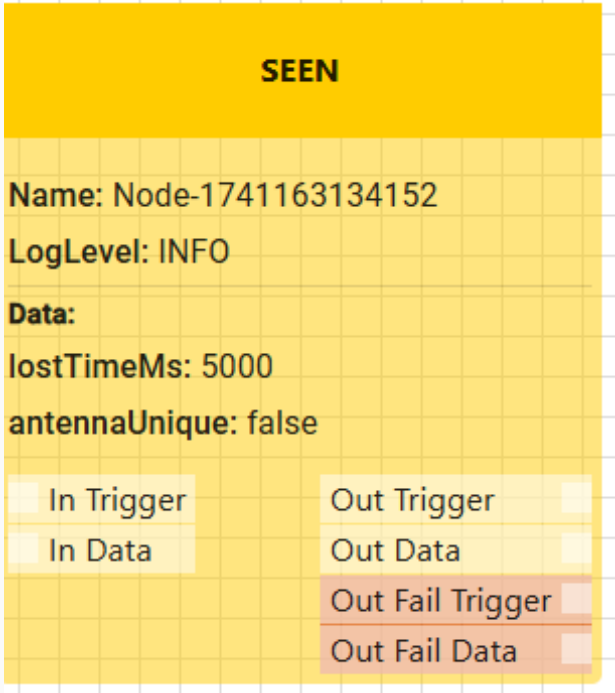



| Name | Type | Description | Required |
|---|---|---|---|
| mask | string, representing hexadecimal value (0-9 A-F) | Hex string defining what bits are selected for modification. | Yes |
| replacement | string, representing hexadecimal value (0-9 A-F) | Hex string defining new values for selected bits. | Yes |

# SEEN Module

The SEEN module provides a way to get events when a tag appears and disappears.

Module keeps an internal buffer of tag reads. When an EPC arrives, that is not in the buffer, a seen event is passed to next module and the tag is stored to a buffer, along with a timestamp. Each time the tag arrives, its timestamp will be updated. If the age of the tag in the buffer exceeds 'lostTimeMs', the tag is removed from the buffer and a lost event is passed to the next module. Lost event being identified by the original event's antenna id + 100.

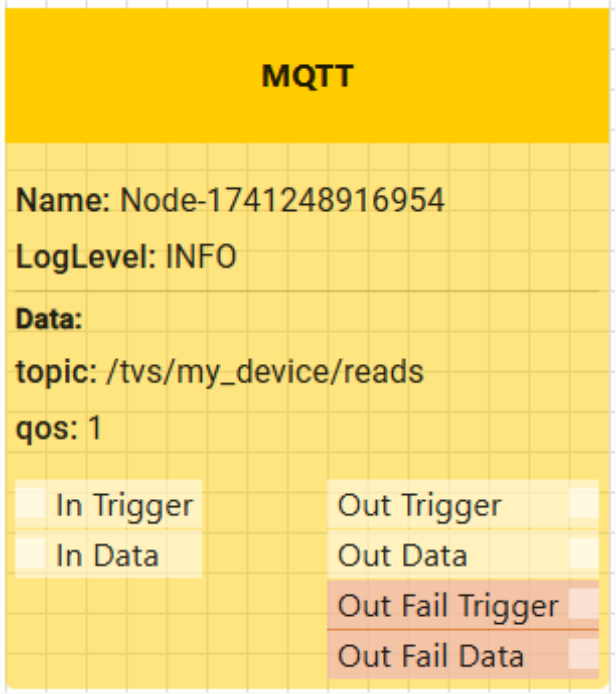



| Name | Type | Description | Required |
|---|---|---|---|
| lostTimeMs | integer (0-2147483647) | Buffer age of tag in milliseconds. If tag has not been seen in this time, it is considered lost. | Yes |
| antennaUnique | boolean | Defaults to false. If true, then EPCs from different antennas are considered unique. | No |

# MQTT Sender (Pro) Module

The MQTT module can be used to send tag events to a specified MQTT broker.

The message is sent to the endpoint with MQTT protocol. The message structure follows the common Tag Data json format.

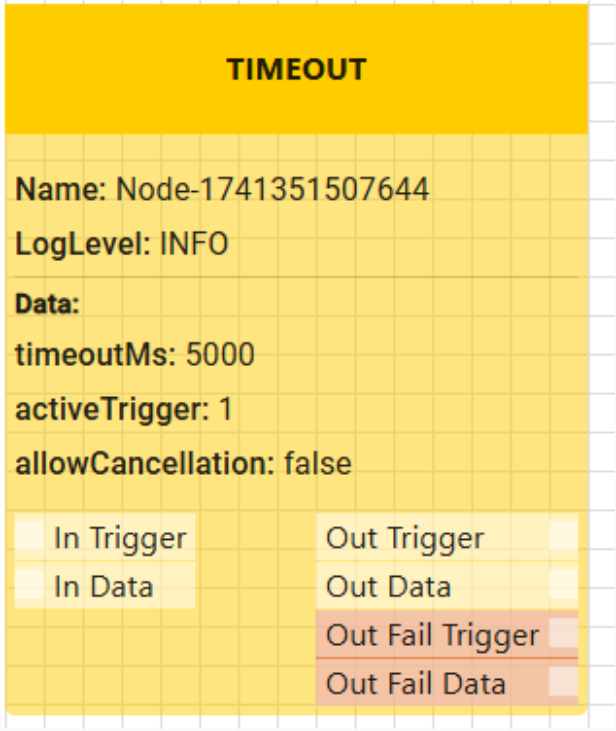



| Name | Type | Description | Required |
|---|---|---|---|
| topic | string | The topic to which tag event is published. If used, module will replace <ID> with the device unique id, found on status page | Yes |
| qos | string | Quality of Service for MQTT message. 0) At most once delivery 1) At least once delivery 2) Exactly once delivery. Defaults to 0. | Yes |

# TIMEOUT Module

The TIMEOUT module allows you to create timeout-based activations of triggers. After receiving a trigger with the activeTrigger event, the module will send that trigger and wait until timeout. After timeout, the module will send a non-active trigger.

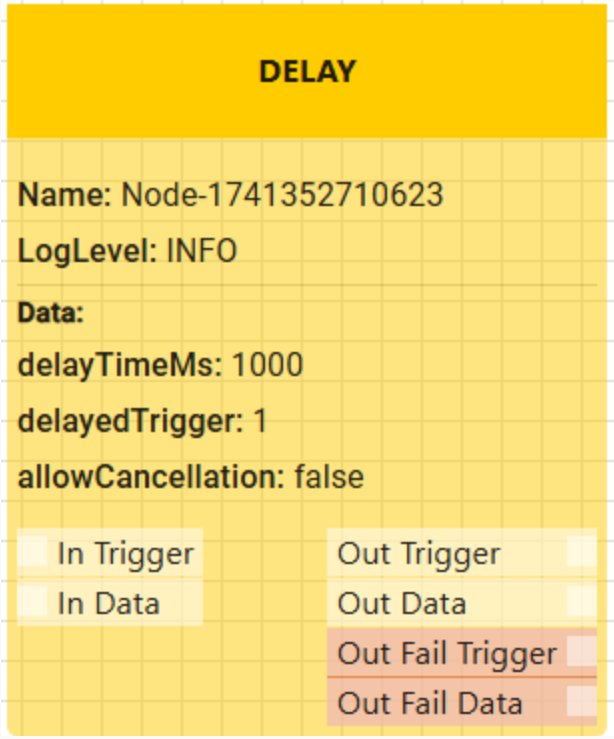
One useful scenario for this module is to create a trigger for the RFID_READER module that is activating RFID reading only for the specified timeout period. If new active trigger is received when timeout period is still active, then the timeout period is restarted.




| Name | Type | Description | Required |
|---|---|---|---|
| timeoutMs | integer (10-2147483647) | How long to keep trigger active until timeout | Yes |
| activeTrigger | integer (0-1) | Defines whether active trigger with timeout is ON (1) or OFF (0). Defaults to ON (1). | No |
| allowCancellation | boolean | Defines whether timeout period can be cancelled before timeout with non-active trigger events. Defaults to disabled. | No |

# DELAY Module

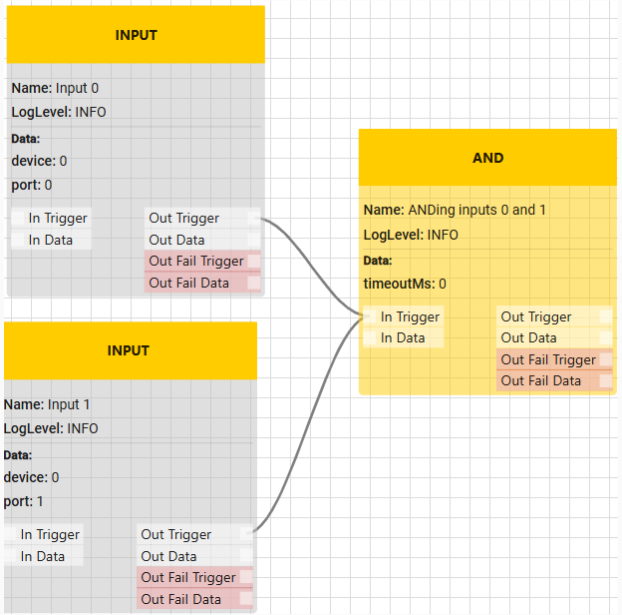
The DELAY module can be used to delay the passing of the event to the next module. The module will wait for the specified time before passing the event.




| Name | Type | Description | Required |
|---|---|---|---|
| delayTimeMs | integer (10-2147483647) | How long to delay sending of the trigger event | Yes |
| delayedTrigger | integer (0-1) | Defines whether delayed trigger event is ON (1) or OFF (0). Defaults to ON (1). | No |
| allowCancellation | boolean | Defines whether the delayed trigger will be cancelled if receiving non-delayed trigger event before delay timeout. Defaults to disabled. | No |

# AND Logic Module

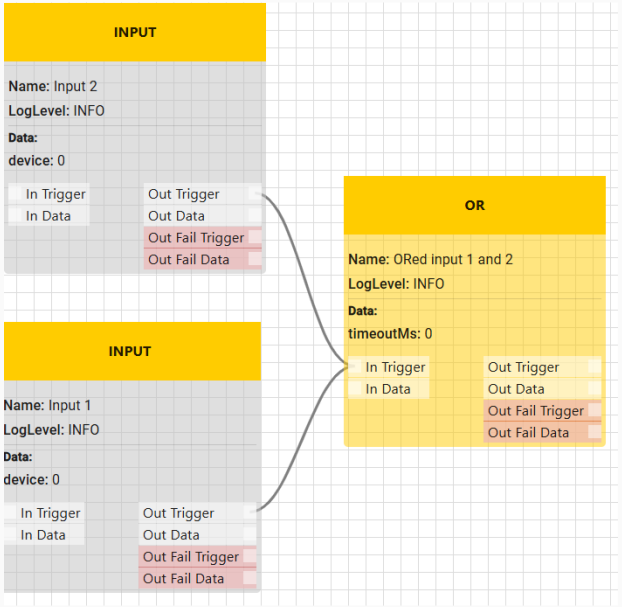
The AND module can be used to combine multiple events into a single event. This module requires all input module triggers to be set to send an active event




| Name | Type | Description | Required |
|---|---|---|---|
| timeoutMs | integer (0-2147483647) | How long an input trigger is considered set. When the timeout is reached, the trigger is reset. Defaults to 0 | No |

# OR Logic Module

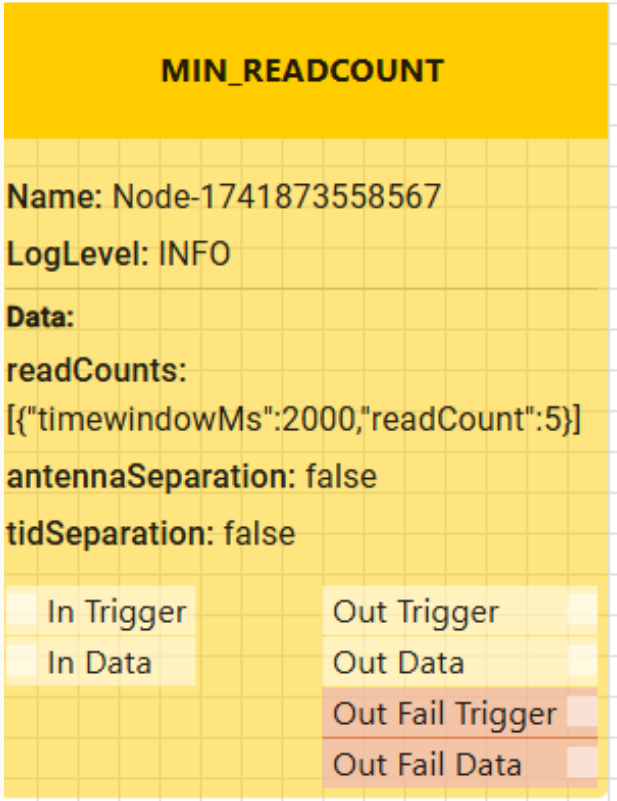
The OR module can be used to combine multiple events into a single event. This module requires any input module triggers to be set to send an active event




| Name | Type | Description | Required |
|---|---|---|---|
| timeoutMs | integer (0-2147483647) | How long an input trigger is considered set. When the timeout is reached, the trigger is reset. Defaults to 0 | No |

# MIN_READCOUNT (Pro) Module 1/2

MIN_READCOUNT module filters RFID tag data based on buffer(s) collected during sliding time window. No tag read forwarding happens until the buffer size reaches the minimum read count.




| Name | Type | Description | Required |
|---|---|---|---|
| readCounts | array of object Readcount | Array of entries defining read count and time limitations for antennas. If `antennaSeparation` is disabled, then first object's properties are used for all tag reads. | Yes |
| antennaSeparation | boolean | Antenna separation of RFID reads. If enabled, then reads are grouped based on epc or tid and antenna. If disabled, then reads are grouped based on epc or tid only and only first readCounts object is used. | No |
| tidSeparation | boolean | TID separation of RFID reads. Singulates tag reads based on tid instead of epc. Tag read is dropped if tid is missing. | No |

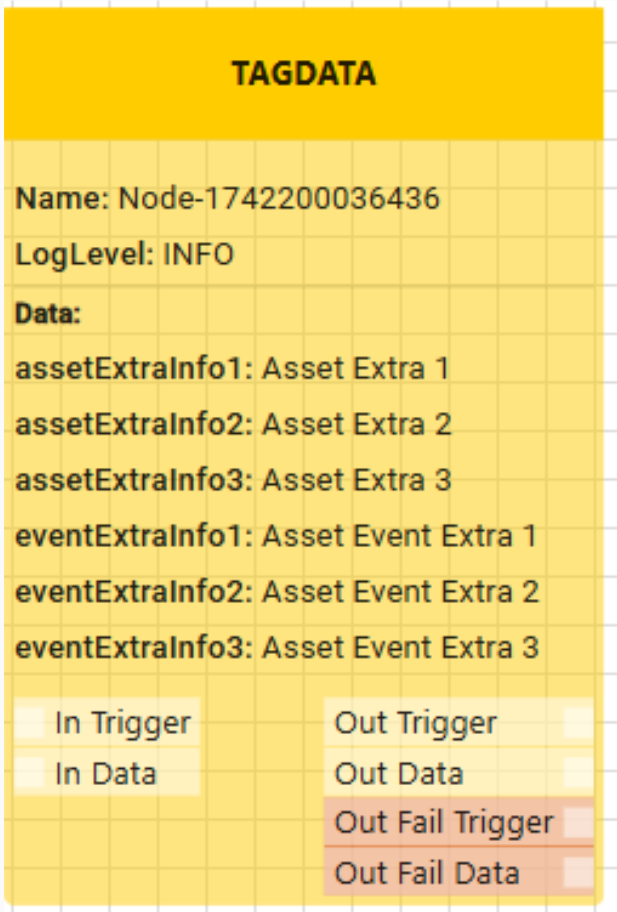
# MIN_READCOUNT (Pro) Module 2/2

ReadCount object properties:

| Name | Type | Description | Required |
|---|---|---|---|
| timeWindowMs | integer (1-2147483647) | Sliding time window in milliseconds. Default is 1000 ms. | Yes |
| readCount | integer (1-2147483647) | Minimum number of reads in buffer before tag is forwarded to next module. | Yes |

# TAGDATA (Pro) Module

The TAGDATA module can be used to add extra information to tag read object passing this module




| Name | Type | Description | Required |
|---|---|---|---|
| assetExtraInfo1 | string, max 20 characters | Data to be put into assetExtraInfo1 field of tag read object | Yes |
| assetExtraInfo2 | string, max 20 characters | Data to be put into assetExtraInfo2 field of tag read object | Yes |
| assetExtraInfo3 | string, max 20 characters | Data to be put into assetExtraInfo3 field of tag read object | Yes |
| eventExtraInfo1 | string, max 20 characters | Data to be put into eventExtraInfo1 field of tag read object | Yes |

| | | | |
|---|---|---|---|
| eventExtraInfo2 | string, max 20 characters | Data to be put into eventExtraInfo2 field of tag read object | Yes |
| eventExtraInfo3 | string, max 20 characters | Data to be put into eventExtraInfo3 field of tag read object | Yes |

# Modules Configuration

# CONNECTING MODULES

Module connections description here

# WEB UI MODIFICATION