SESSION

04

Edge Analytics Fundamentals

WORKING WITH ARRAYS

Master the different Array modules

Session 4

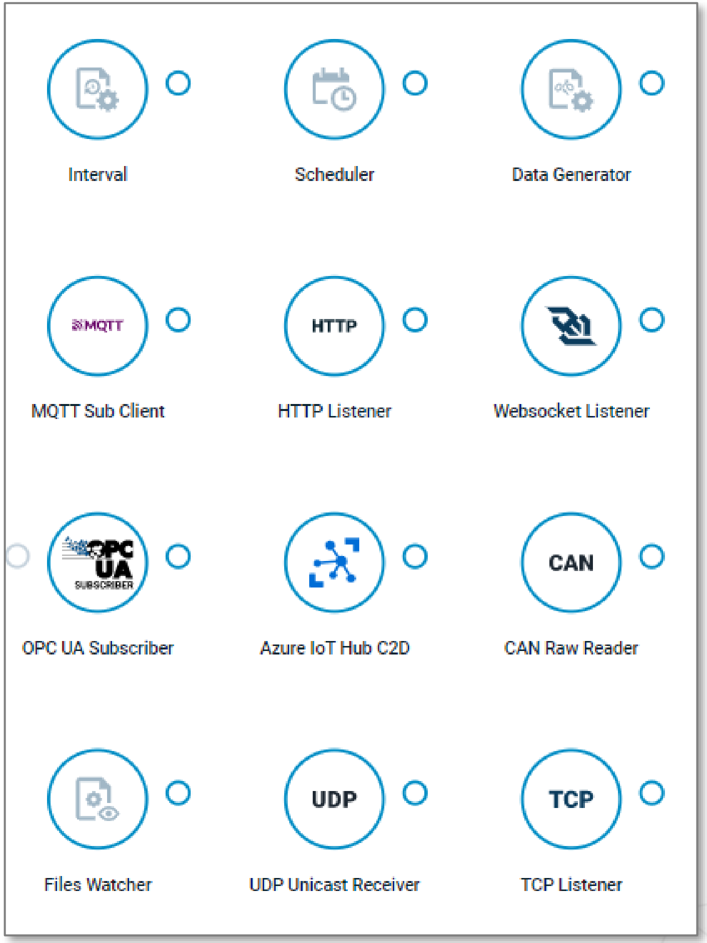
Agenda

- Trigger modules
- Array modules
- Exercise 3: Processing array data

Trigger Modules

Starting a flow

- Modules will only do something when they receive message
- Processing always starts with a *Trigger* module
  - Internal triggers: Interval, Scheduler, Data Generator
  - External triggers: MQTT Sub, OPC UA Subscriber, HTTP Listener…
- Trigger modules have no input connector



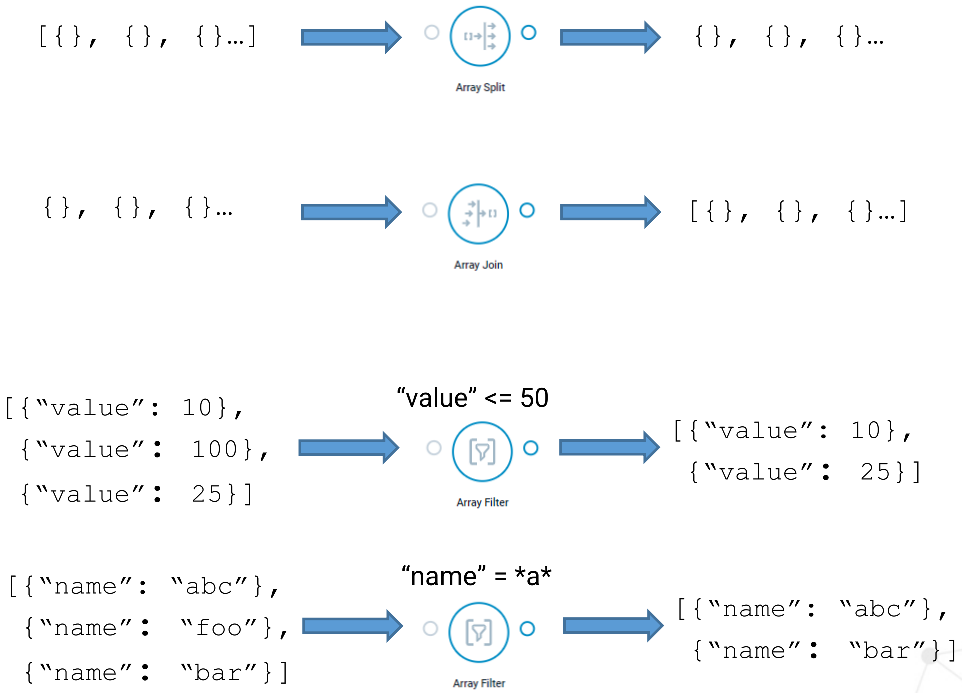
WORKING WITH ARRAYS

The *Array* modules

Arrays

- Input and Output modules sometimes use arrays of messages (objects)
- Analytics modules typically operates on one value per message
- ‘*Array **’ modules operates on arrays
- Create arrays (*Join*) by time or number of messages

Split, Join and Filter modules



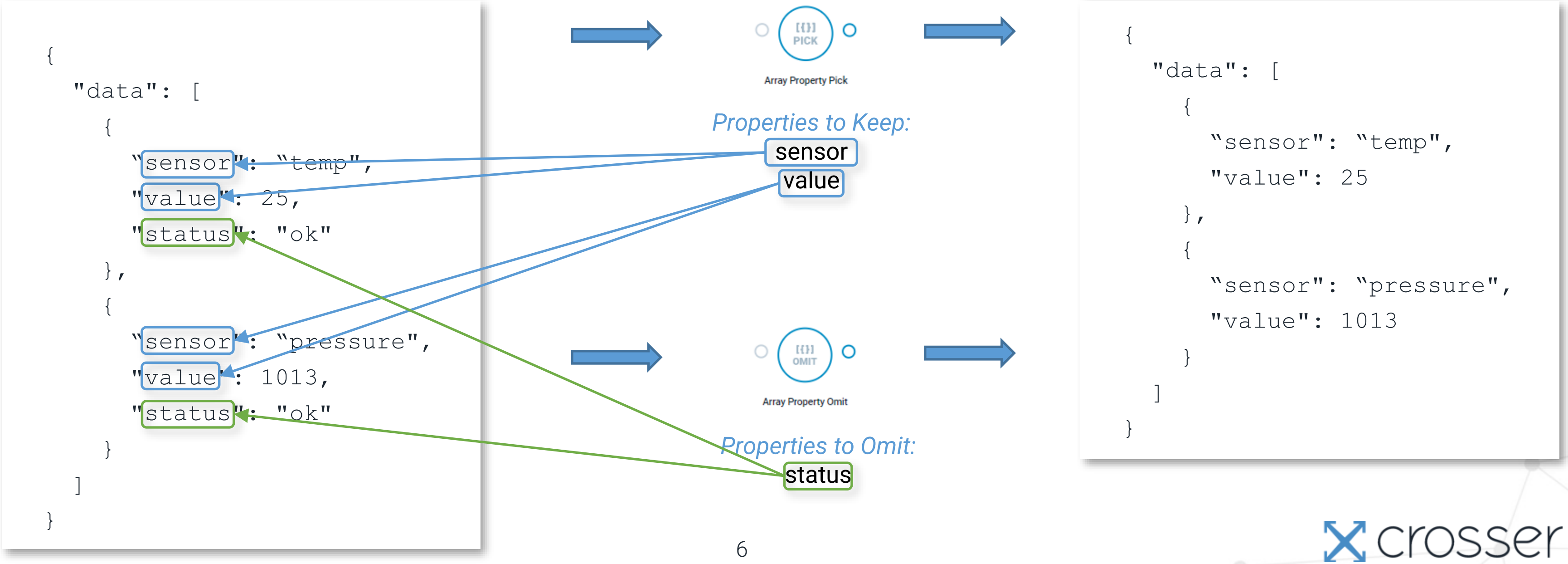
```
[{}, {}, {}…]  →  Array Split  →  {}, {}, {}…

{}, {}, {}…  →  Array Join  →  [{}, {}, {}…]

                  “value” <= 50
[{“value”: 10},
 {“value”: 100},  →  Array Filter  →  [{“value”: 10},
 {“value”: 25}]                        {“value”: 25}]

                  “name” = *a*
[{“name”: “abc”},
 {“name”: “foo”},  →  Array Filter  →  [{“name”: “abc”},
 {“name”: “bar”}]                      {“name”: “bar”}]
```

Module

Array Pick/Omit

Use Array Property Pick to select which properties to keep in each object in the array

Use Array Property Omit to select which properties to remove from each object in the array



```
{
  "data": [
    {
      "sensor": “temp",
      "value": 25,
      "status": "ok"
    },
    {
      "sensor": “pressure",
      "value": 1013,
      "status": "ok"
    }
  ]
}
```

Properties to Keep:

sensor

value

Properties to Omit:

status

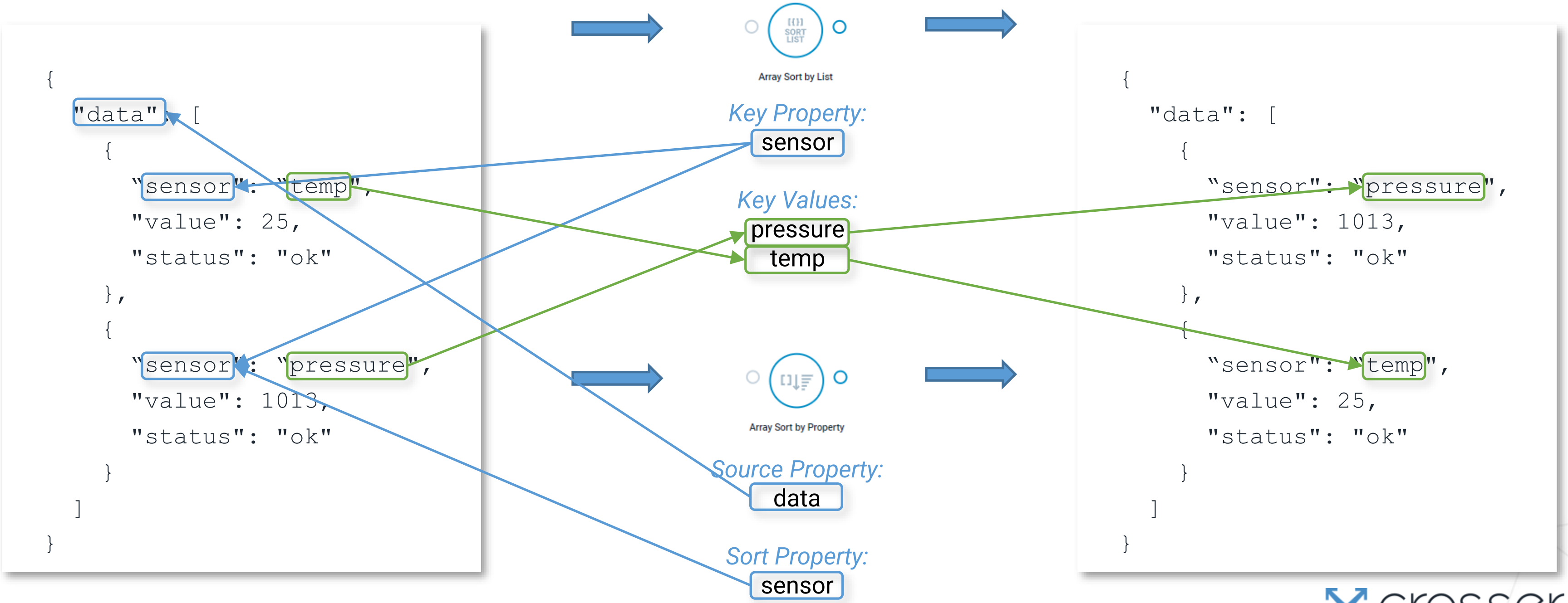
```
{
  "data": [
    {
      “sensor": “temp",
      "value": 25
    },
    {
      “sensor": “pressure",
      "value": 1013
    }
  ]
}
```

Module

Array Sort By List / Array Sort By Property

Use Array Sort By List to order the objects in the array based on a list of property values

Use Array Sort By Property to order objects in the array based on a value of a property (ascending or descending, strings or numbers)



```
{
  "data": [
    {
      "sensor": "temp",
      "value": 25,
      "status": "ok"
    },
    {
      "sensor": "pressure",
      "value": 1013,
      "status": "ok"
    }
  ]
}
```

Array Sort by List

Key Property: sensor

Key Values: pressure
temp

Array Sort by Property

Source Property: data

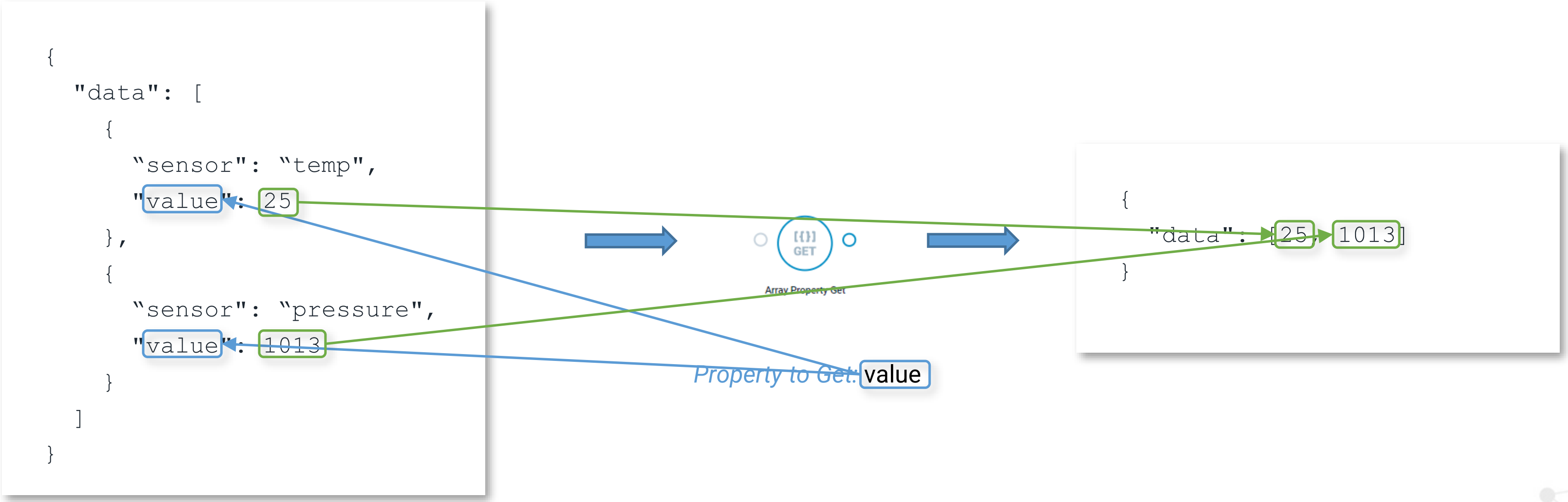
Sort Property: sensor

```
{
  "data": [
    {
      "sensor": "pressure",
      "value": 1013,
      "status": "ok"
    },
    {
      "sensor": "temp",
      "value": 25,
      "status": "ok"
    }
  ]
}
```

Module

Array Get

Use Array Get to convert an array of objects into an array of values



```
{
  "data": [
    {
      “sensor": “temp",
      "value": 25
    },
    {
      “sensor": “pressure",
      "value": 1013
    }
  ]
}
```

```
{
  "data": [25, 1013]
}
```

Arrays and Objects

Array to Object *and* Object to Array *modules*

Array of Objects

```
{
  "data": [
    {
      “sensor": “temp",
      “reading": 25
    },
    {
      “sensor": “pressure",
      “reading": 1013
    }
  ]
}
```

Array To Object

Object

```
{
  "data": {
      "temp": 25,
      "pressure": 1013
  }
}
```

Object To Array

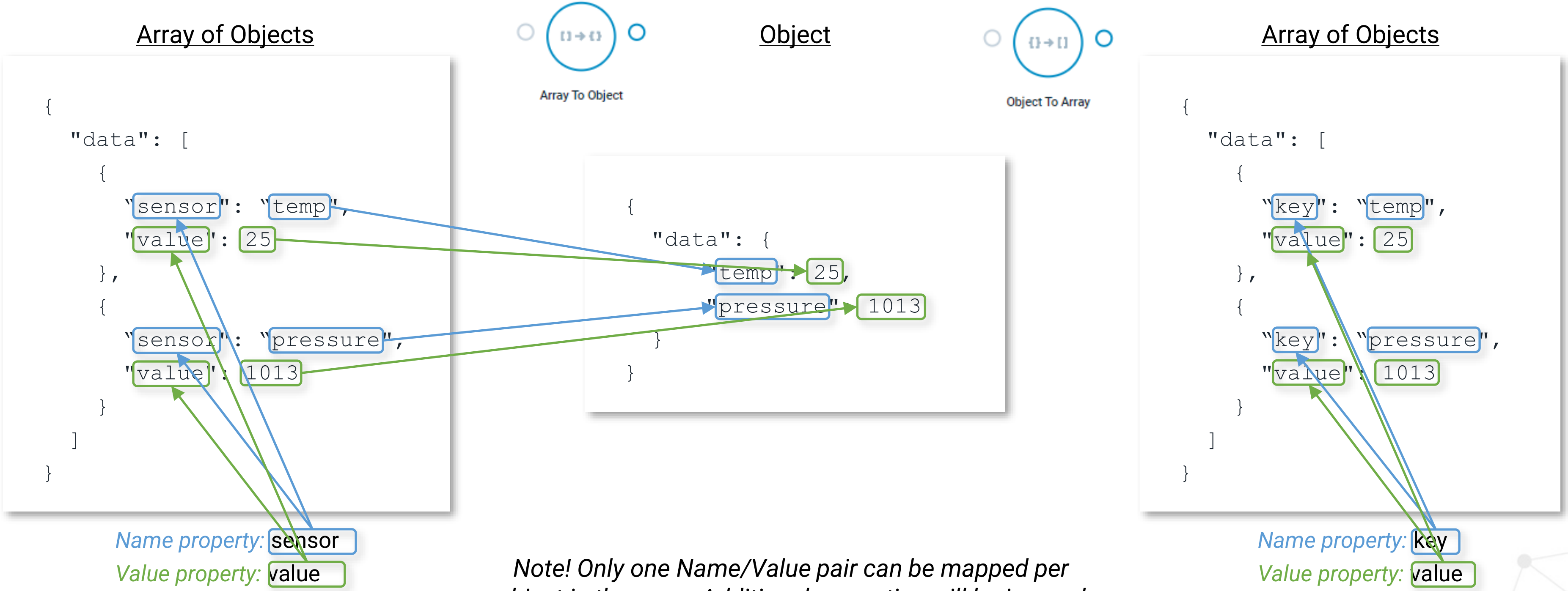
Array of Objects

```
{
  "data": [
    {
      “key": “temp",
      "value": 25
    },
    {
      “key": “pressure",
      "value": 1013
    }
  ]
}
```

Note! Only one Name/Value pair can be mapped per object in the array. Additional properties will be ignored

Arrays and Objects

Array to Object *and* Object to Array *modules*



Array of Objects

```
{
  "data": [
    {
      "sensor": "temp",
      "value": 25
    },
    {
      "sensor": "pressure",
      "value": 1013
    }
  ]
}
```

Name property: sensor

Value property: value

Object

```
{
  "data": {
    "temp": 25,
    "pressure": 1013
  }
}
```

Array of Objects

```
{
  "data": [
    {
      "key": "temp",
      "value": 25
    },
    {
      "key": "pressure",
      "value": 1013
    }
  ]
}
```

Name property: key

Value property: value

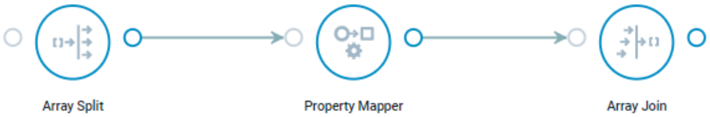
Note! Only one Name/Value pair can be mapped per object in the array. Additional properties will be ignored

Looping Over Array Elements

Array Split and *Array Join* modules working together

<u>Input array</u>

```
{
  "data": [
    {
      “sensor": “temp",
      “reading": 25
    },
    {
      “sensor": “pressure",
      “reading": 1013
    }
  ]
}
```



<u>Output array</u>

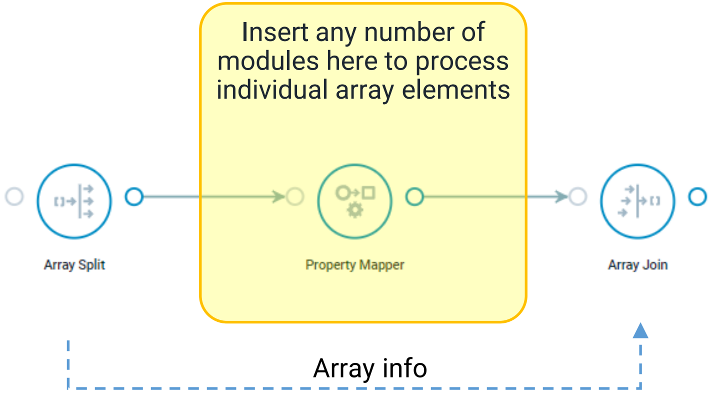
```
{
  "data": [
    {
      “key": “temp",
      "value": 25
    },
    {
      “key": “pressure",
      "value": 1013
    }
  ]
}
```

Looping Over Array Elements

Array Split and *Array Join* modules working together

Input array

```
{
  "data": [
    {
      “sensor": “temp",
      “reading": 25
    },
    {
      “sensor": “pressure",
      “reading": 1013
    }
  ]
}
```



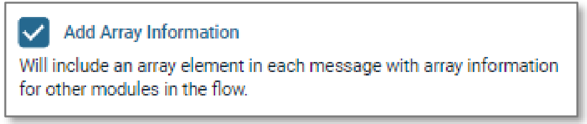
Insert any number of modules here to process individual array elements

Array info

Output array

```
{
  "data": [
    {
      “key": “temp",
      "value": 25
    },
    {
      “key": “pressure",
      "value": 1013
    }
  ]
}
```

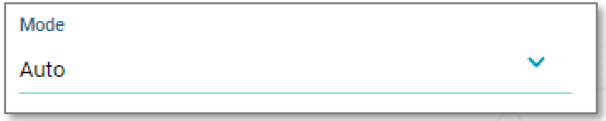
Enable this setting in Array Split



☑ Add Array Information

Will include an array element in each message with array information for other modules in the flow.

Set the ‘Mode’ in Array Join to ‘Auto’



Mode

Auto

EXERCISE 3

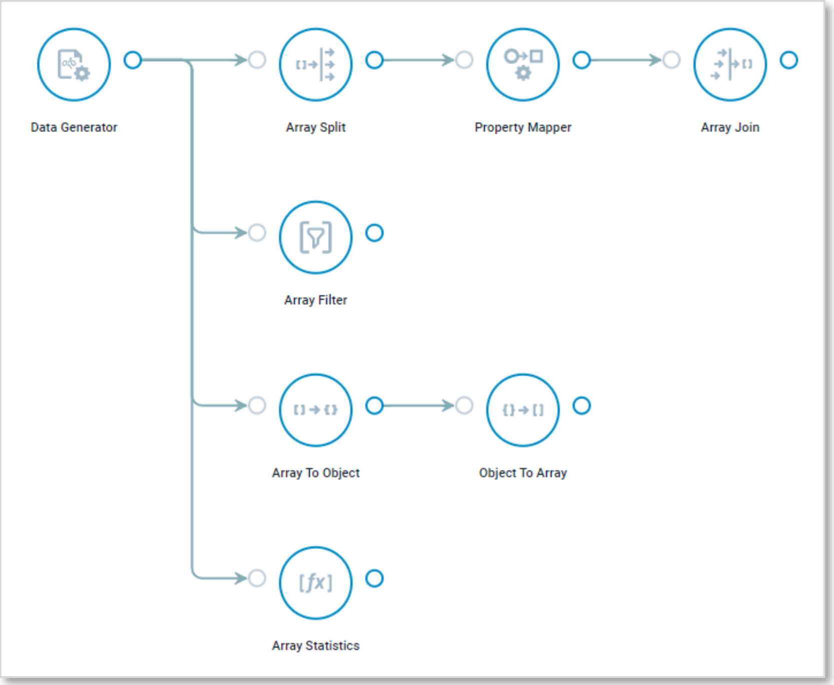
Working with arrays

Exercise 3

Overview

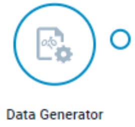
- How to work with array data:
  - Loop over array elements
  - Filter array elements
  - Convert an array to an object
  - Do calculations on array data

Note: You can use the same Data Generator module in all exercises



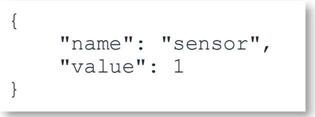
Exercise 3.1

Generate array data



Use a Data Generator module to create arrays that will be used as input data in all these exercises

1. Create a new flow called Exercise 3
2. Add a Data Generator module
   - Number of Samples: 5
   - Output Strategy: All Items as Array
   - JSON:

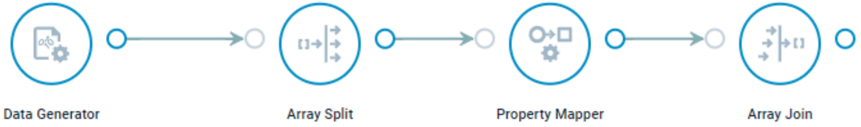


```
{
    "name": "sensor",
    "value": 1
}
```

   - Rules:
     - Data.name / Behavior: Identifier
     - Data.value / Max: 100
3. Run the Flow and check the output

Exercise 3.2

Loop over array elements

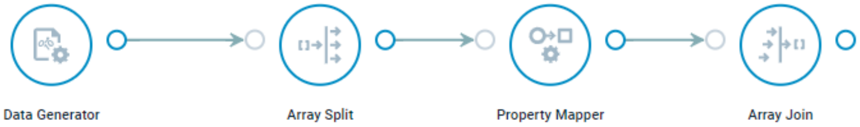


Run the Flow after each step and check the output

1. Break up the array into individual messages by adding an Array Split module
2. Do something with the data in each array element, e.g. by adding a Property Mapper module and rename ‘value’ to ‘measurement’
   Note: Any number of modules can be used between the Array Split and Join modules to perform operation on array elements
3. Re-combine the array by adding an Array Join module:
   - Mode: Count
   - Samples: 5
4. In the Data Generator, change “Number of Samples” to 10

Exercise 3.2

Loop over array elements



Run the Flow after each step and check the output

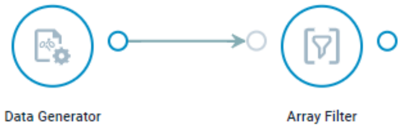
5. In the Array Split module, enable Add Array Information
6. In the Array Join module, change Mode to Auto

Note

There are two other modules that can create arrays out of sequences of messages: Window and Join

Exercise 3.3

Filter array elements

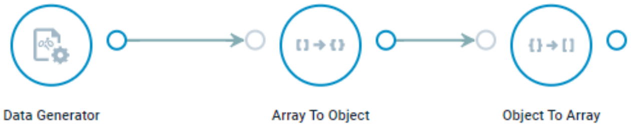


Run the Flow after each step and check the output

1. Add an Array Filter module:
    - Add a filter:
    - Property Name: value
    - Condition: Less Than
    - Property Value: 50

Exercise 3.4

Convert Arrays to Objects and back

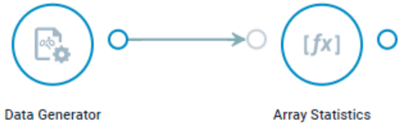


Run the Flow after each step and check the output

1. Add an Array To Object module:
   - Name Property: name
   - Value Property: value
2. Add an Object To Array module:
   - Name Property: name
   - Value Property: value

Exercise 3.5

Process array data

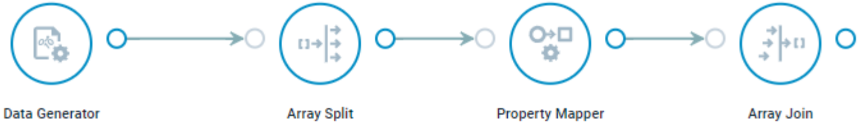


Run the Flow after each step and check the output

1. Add an Array Statistics module:
   - Target Property: stats
   - Value Property: value

Exercise 3

Wrap-up



Things to test/consider:

- Why did we have to enable “Add array information” on the Array Split module?
  - Take a look at the output from the module
- Why can’t you have a module like Aggregate in between Array Split and Join?
- Why is the number of elements in the final array different when you change mode to *auto* in the last step of exercise 3.2?
- The Array Statistics module produces an output similar to the Aggregate module we used before. Why couldn’t we use Aggregate here and how could you get the functionality of Aggregate by combining some of the modules used in this exercise?
- Extra exercises:
  - Use the Array Sort By Property module to sort the array elements based on the values
  - Use the Array Sort By List module to change the order of the elements based on the name

SESSION – 04 END

Working with Arrays