BAARS error handling

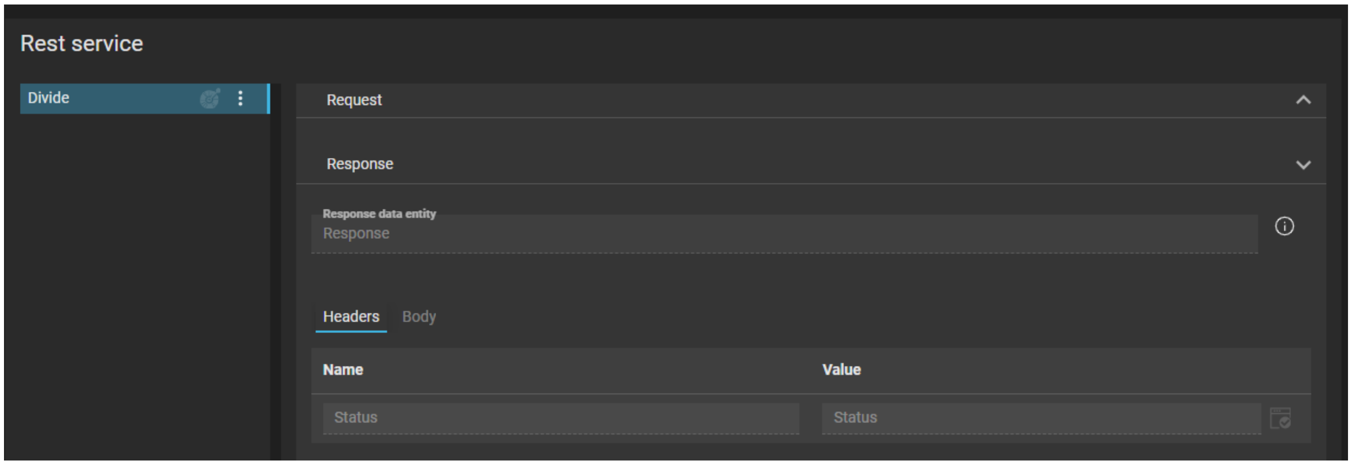
Tags	Webservice, REST
Type of accelerator	Example project to demonstrate Blueriq functionality and accelerate your learning journey.
How to get	BaarsErrorHandling.package.zip
Compatibility	Blueriq 16.0 and higher

Description

This model shows how error handling can be modeled in Blueriq as a Rest service (BAARS).

HTTP status code

The HTTP status code can be used in the model to handle errors. The HTTP status code is a header in the response and can be mapped to an attribute in the model (example below). Blueriq only supports single-valued numeric status codes between 2xx and 5xx. If something else is mapped, the BAARS will throw a 500 message. Mind the capital letter S in Status since the header name is case sensitive.



Response data entity: Response

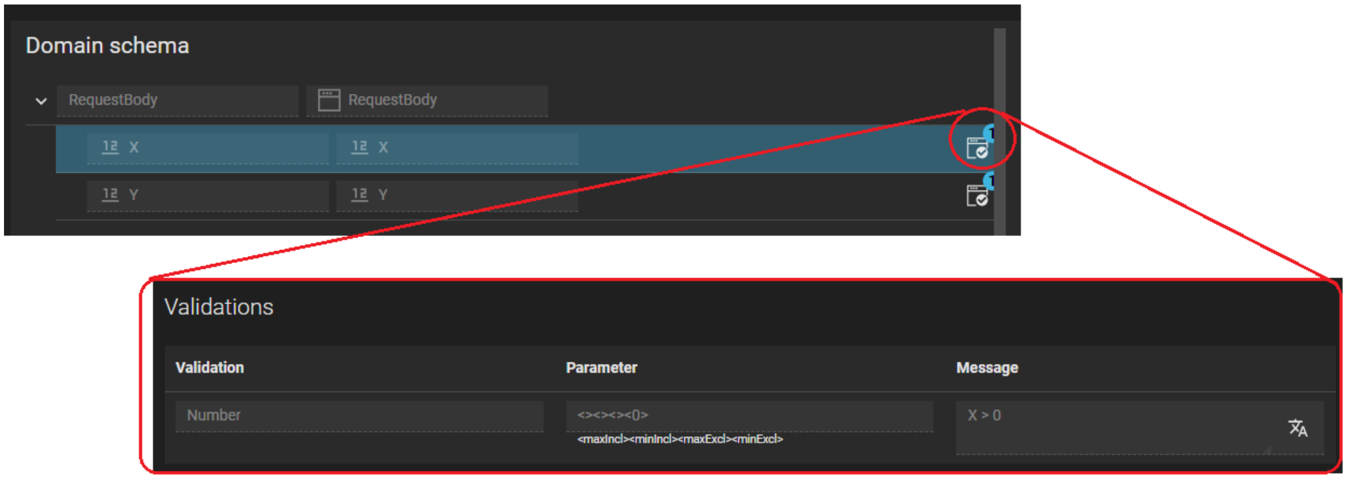
Name	Value
Status	Status

Types of validations / errors

Input validation

Input validation is done on the received request. Input validations can be modeled on the domain schema of the request (example below).

For the AQ_RestServiceClient, no validation is performed on the response.



Validation	Parameter	Message
Number	<><><><0> <maxIncl><minIncl><maxExcl><minExcl>	X > 0

Parsing and validating the request

For REST services based on Domain Schemas, the request is parsed, mapped in the profile and the elements (attributes and relations) are validated. This means that when the validation starts, the profile is complete, so elements from **fragments, arguments, headers and request body** that depend on each other can also be validated.

For REST services based on XSD schemas, validations are performed on the fly, as the message is being parsed. This means that when the validation of an element starts, the validation rules on that element might depend on other attributes and relations which have not been parsed yet, and whose values at that point are unknown.

Basic input validation

By default, validation messages are returned in a single string error message, concatenated with new lines characters. This validation mode is available for both XSD based and Domain Schema based REST services.

Example request

```
POST ../TestApp/0.0-Trunk/Insurance/fragmentTooLong?argumentName=valueTooLong

Request Headers
...
headerName: valueTooLong
...

Request Payload
{
        "insured_person": {
                "first_name": "John",
                "last_name": "",
                "birth_date": "2005-05-10"
        },
        "coverage":[{
                "type": "life",
                "life": {
                        "claim_limit": 100000,
                        "last_health_checkup": "1902-01-01"
                }
        }, {
                "type": "car",
                "car": {
                        "claim_limit": 5000,
                        "fabrication_date": "1982-01-01"
                }
        }]
}
```

Example of response when extended input validation is not enabled

```
{
  "errorMessage": "Unable to handle request, validation messages: First name is required\nPerson must be at
least 18 years old to obtain an insurance.\nYou must be over the age of 25 to insure a car older than 10
years.\nYou must be at least 21 years old to be eligible for car insurance.\nName may only be 5 characters
long.\nName may only be 5 characters long.\nName may only be 5 characters long."
}
```

Extended input validation

For REST services based on Domain Schemas, input validation can be set to return an extended response in the form of a validation tree of the whole request domain. The validation tree contains four sections: `arguments`, `body`, `fragments` and `headers`. In this way, it is easier to identify the instance on which a validation error occurred. By default, this behavior is disabled. For more details on how to enable this functionality, please check the documentation here.

Example request

```
POST ../TestApp/0.0-Trunk/Insurance/fragmentTooLong?argumentName=valueTooLong

Request Headers
...
headerName: valueTooLong
...

Request Payload
{
        "insured_person": {
                "first_name": "John",
                "last_name": "",
                "birth_date": "2005-05-10"
        },
        "coverage":[{
                "type": "life",
                "life": {
                        "claim_limit": 100000,
                        "last_health_checkup": "1902-01-01"
                }
        }, {
                "type": "car",
                "car": {
                        "claim_limit": 5000,
                        "fabrication_date": "1982-01-01"
                }
        }]
}
```

Example of a response with extended input validation enabled

```
{
  "errorMessage": "There are invalid fields.",
  "validationTree": {
    "arguments" : [{
          "name": "argumentName",
          "value": ["valueTooLong"],
      "validations" : [{
        "message": "Name may only be 5 characters long."
          }]
    }],
    "body": {
      "values": [],
      "objects": [
        {
          "name": "insured_person",
          "values": [
            {
              "key": "first_name",
              "value": "John",
              "validations": []
            },
            {
              "key": "last_name",
              "value": "",
              "validations": [
                {
                  "message": "Last name is required!"
                }
              ]
            },
            {
              "key": "birth_date",
              "value": "2005-05-10",
              "validations": [
                {
                  "message": "Person must be at least 18 years old to obtain an insurance."
                }
              ]
            }
          ],
          "objects": [],
          "validations": []
        },
        {
          "name": "coverage",
          "objects": [
            {
              "values": [
                {
                  "key": "type",
                  "value": "life",
                  "validations": []
                }
              ],
              "objects": [
                {
                  "name": "life",
                  "values": [
                    {
                      "key": "claim_limit",
                      "value": "100000.0",
                      "validations": []
                    },
                    {
                      "key": "last_health_checkup",
                      "value": "1902-01-01",
                      "validations": []
                    }
                  ],
                  "objects": [],
                  "validations": []
                }
              ],
              "validations": []
            },
            {
              "values": [
                {
                  "key": "type",
                  "value": "car",
                  "validations": []
                }
              ],
              "objects": [
                {
                  "name": "car",
                  "values": [
                    {
                      "key": "claim_limit",
                      "value": "5000.0",
                      "validations": []
                    },
                    {
                      "key": "fabrication_date",
                      "value": "1982-01-01",
                      "validations": [
                        {
                          "message": "You must be over the age of 25 to insure a car older than 10 years."
                        },
                        {
                          "message": "You must be at least 21 years old to be eligible for car insurance."
                        }
                      ]
                    }
                  ],
                  "objects": [],
                  "validations": []
                }
              ],
              "validations": []
            }
          ],
          "validations": []
        }
      ],
      "validations": []
    },
    "fragments": [
          "name": "fragmentName",
          "value": "fragmentTooLong",
          "validations": [{
                "message": "Name may only be 5 characters long."
          }]
    ],
    "headers": [
          "name": "headerName",
          "value": "valueTooLong",
      "validations": [{
            "message": "Name may only be 5 characters long."
          }]
    ]
  }
}
```

Constructing the structured validation response

Values returned in the validation messages are from the profile and not directly from the initial request. By being returned from the profile, it means that value formatting applies to the values that are present in the validation message, like in the example below. Please note the number and percentage values from the request and from the response with validation message.

Example request with invalid input

```
{
    "insured_person": {
        "first_name": "John",
        "last_name": "Doe",
        "birth_date": "2018-02-02",
        "number": "2000",
        "percentage": "20"
    }
}
```

Example response with validation messages that contain formatted values

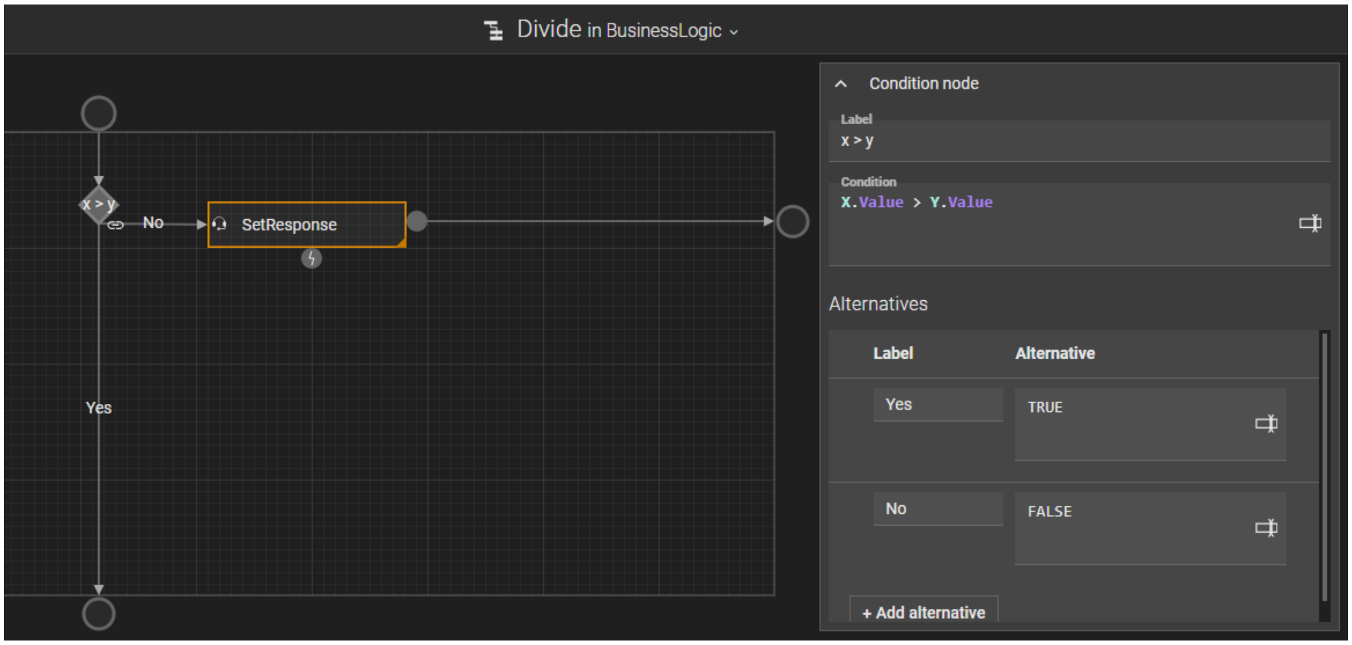
```
{
    "errorMessage": "There are invalid fields.",
    "validationTree": {
        "arguments": [],
        "body": {
            "values": [],
            "objects": [
                {
                    "name": "insured_person",
                    "values": [
                        ...,
                        {
                            "key": "birth_date",
                            "value": "2018-02-02",
                            "validations": [
                                {
                                    "message": "Person must be at least 18 years old to obtain an insurance."
                                }
                            ]
                        },
                        {
                            "key": "number",
                            "value": "2000.0",
                            "validations": []
                        },
                        {
                            "key": "percentage",
                            "value": "20.0",
                            "validations": []
                        }
                    ],
                    "objects": [],
                    "validations": []
                }
            ],
            "validations": []
        },
        "fragments": [],
        "headers": []
    }
}
```

Other considerations on input validation

- Having null values in an array: see point 1 Know Issues Section
- Validating duplicate values: see point 2 in Know Issues Section
- When calling a BAARS from an AQ_RestServiceClient, please note the fact that the client does not validate responses.

Model validation

More complex validations can be handled in the model, using the HTTP status code combined with self created error codes and error messages (example below).


Divide in BusinessLogic

Condition node

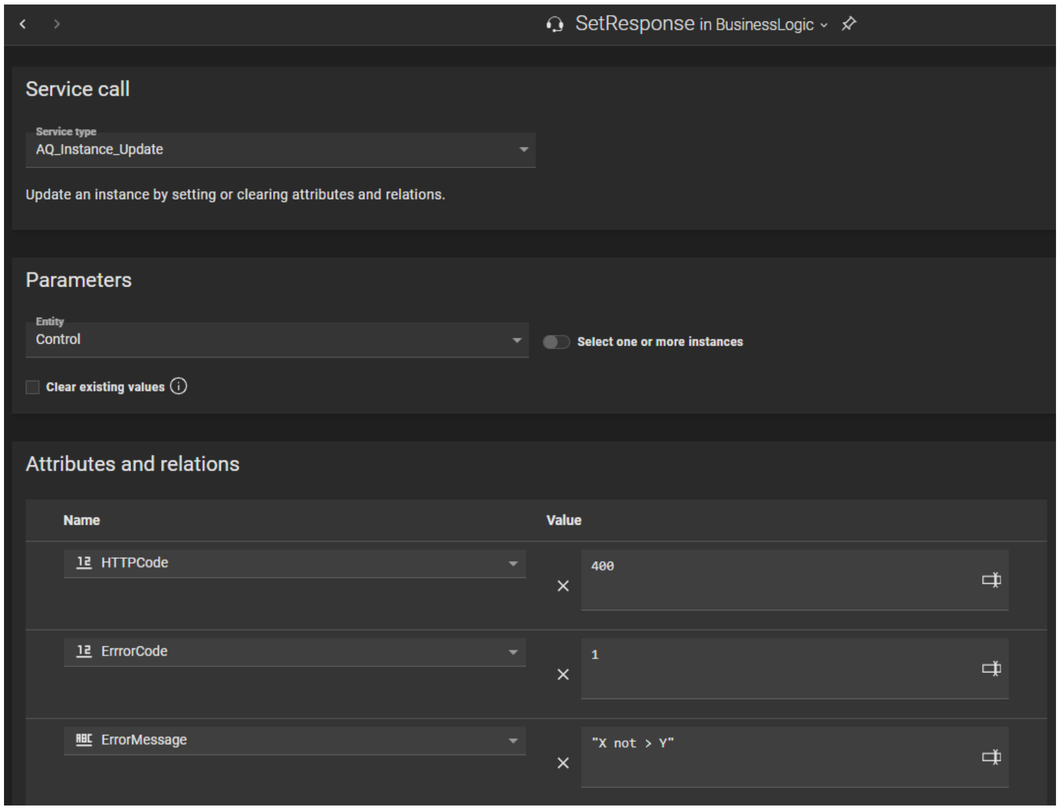
Label: x > y

Condition: `X.Value > Y.Value`

Alternatives

Label	Alternative
Yes	TRUE
No	FALSE

+ Add alternative


SetResponse in BusinessLogic

Service call

Service type: AQ_Instance_Update

Update an instance by setting or clearing attributes and relations.

Parameters

Entity: Control

Select one or more instances

Clear existing values

Attributes and relations

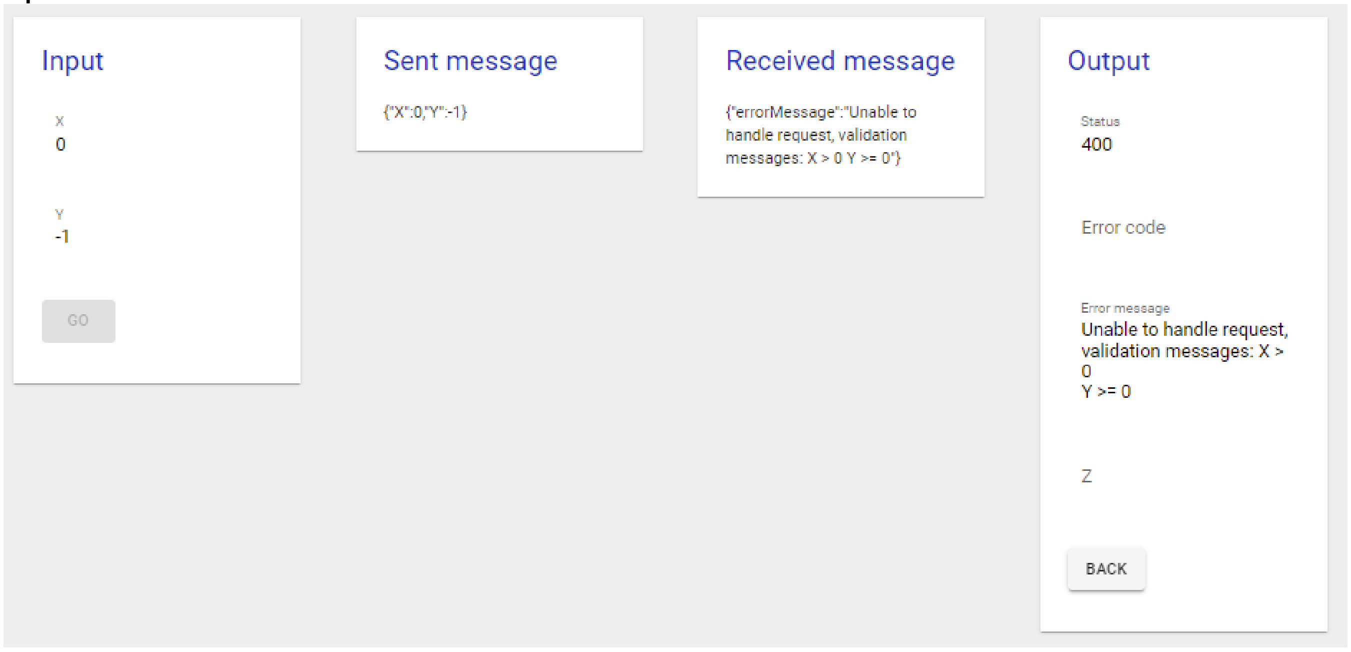
Name	Value
HTTPCode	400
ErrorCode	1
ErrorMessage	"X not > Y"

Errors

The BAARS will throw an 500 message in case of an unexpected error. In the case of the example this is when one or both input parameters are unknown. This should be handled by another model validation so this can not happen in a production environment.

Example project

Input validation:

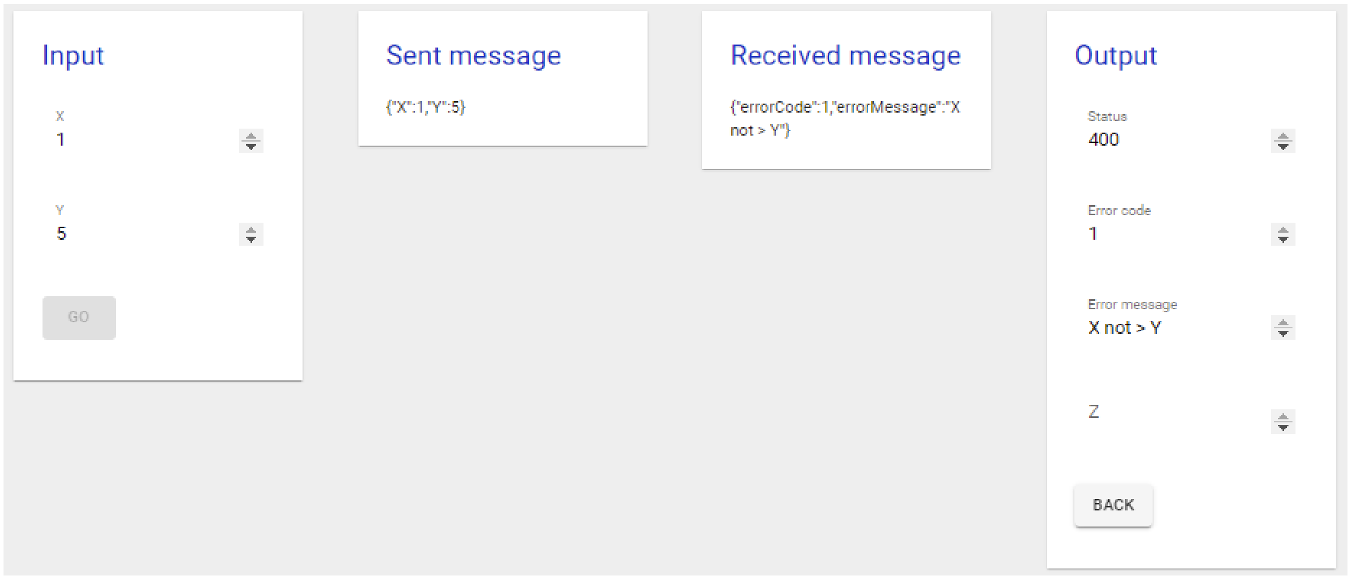


{"X":0,"Y":-1}

{"errorMessage":"Unable to handle request, validation messages: X > 0 Y >= 0"}

Unable to handle request, validation messages: X > 0
Y >= 0

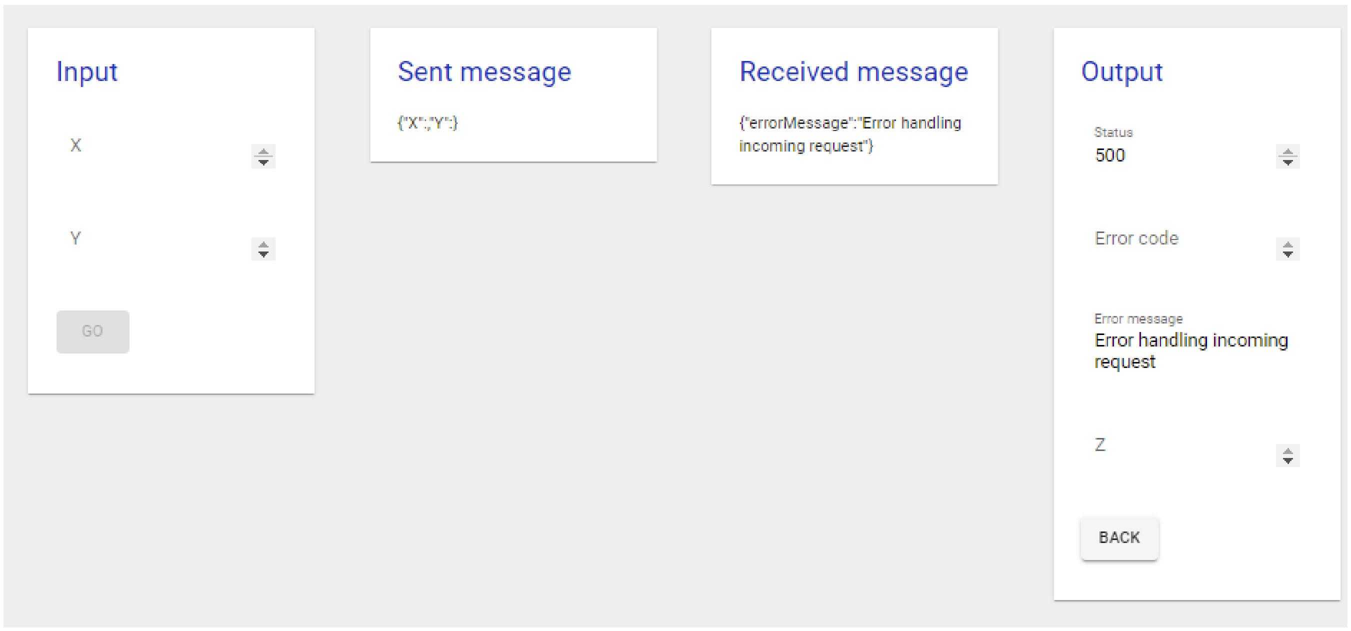
Model validation:



{"X":1,"Y":5}

{"errorCode":1,"errorMessage":"X not > Y"}

Errors:



{"X":,"Y":}

{"errorMessage":"Error handling incoming request"}

Status
500

Error code

Error message
Error handling incoming request

Known issues in input validation

1. When sending a JSON array which contains a null value, the Runtime fails with a 500 Internal Server Error, instead of a 400 Bad Request and some information about what was wrong. This problem is reproducible when using a Rest Service with a domain schema with hidden root.
2. When structured schema validation is used, there are validation errors on the input and the request contained an array with duplicate values, then the part of the structured validation response corresponding to the array with duplicate values will not show the duplicates. This is because values from the validation message are being returned from the profile, and not from the initial request and in Blueriq, multi-valued attributes cannot have duplicates.