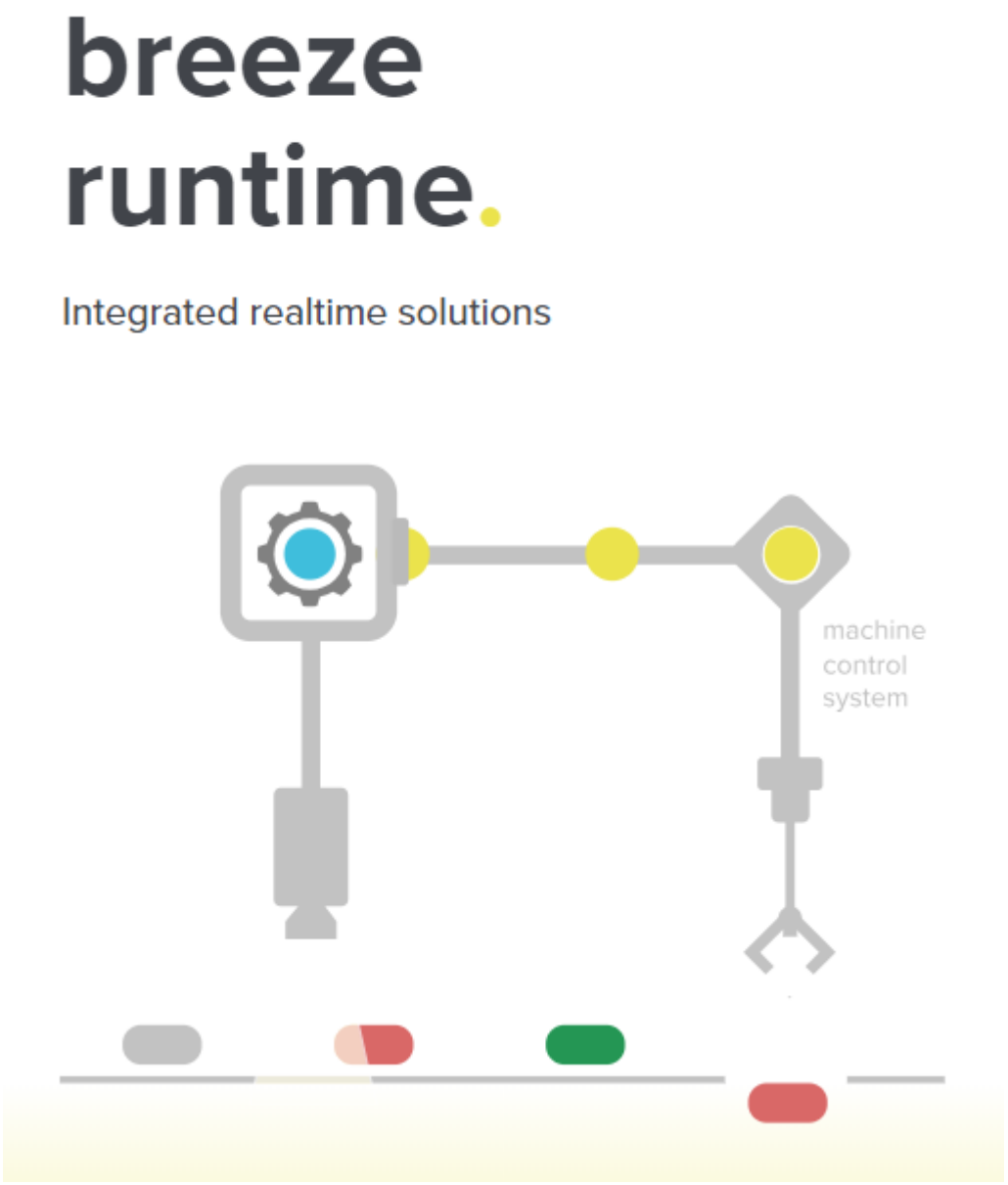
Developers Reference Guide



www.prediktera.com
info@prediktera.com

Table of content

Overview **4**

Commands **5**
- Camera commands 5
  - Initialize camera 5
    - Available cameras 6
  - Get camera property 9
  - Set camera property 10
- Raw-data capture commands 11
  - Start capture 11
  - Stop capture 11
- Prediction commands 11
  - Get workflows 11
  - Load workflow 17
  - Delete workflow 20
  - Open shutter 20
  - Close shutter 20
  - Take dark reference 21
    - Dark reference quality control 21
  - Take white reference 21
    - White reference quality control 22
  - Get workflow setup 22
  - Start predict 22
  - Stop predict 23
  - Get status 23
  - Get property 24
  - Set property 25
  - Re-Initialize prediction 26

Start capture on predict **26**

Error Handling and Events **28**
- Command Errors 28
  - Command Error Codes 28
- Event Stream 29
  - Regular Event Codes 29
    - Example 29
  - Error Event Codes 30
    - Example 30
  - Prediction Object 30

Data Stream **32**
- Header 32

Metadata body 33
Data body 34
Raw Pixel Line 34
Prediction Lines 34
Classification Vector 34
Quantification Vector 34
Confidence Values 34
Stream Start / End 35

Runtime Command Switches 36

Running Breeze Runtime with Breeze Client 37

Appendix A: Event Server 39
Time format 39
Event changes 40
Events specific to the event server 40
White reference intensity 40

Appendix B: Diagnostics 41
Command 41
The diagnostic information overview 41
Logging 42

Appendix C: Code example 43
Preconditions 43
Standard procedure for capturing data 43
Connect to Breeze Runtime for sending commands 43
Example commands and replies 43
Pseudo code 44
Listening to server events (optional) 45
Pseudo code 45
Listening to data stream (optional) 46
Pseudo code 46

Appendix D: Breeze properties 47

Appendix E: Generic camera 48
InitializeCamera command 48
Initialize command (Re-Initialize predicting) 48
Streaming Error 48

Appendix F: Detection Technology camera 49
Failed to obtain energy levels Error 49
Streaming Error 49
Heartbeat Error 49

Overview

This overview shows the parts of a complete system setup integrating Breeze Runtime with a sorting machine. The customer software can run on PC 1 or a separate PC 2.

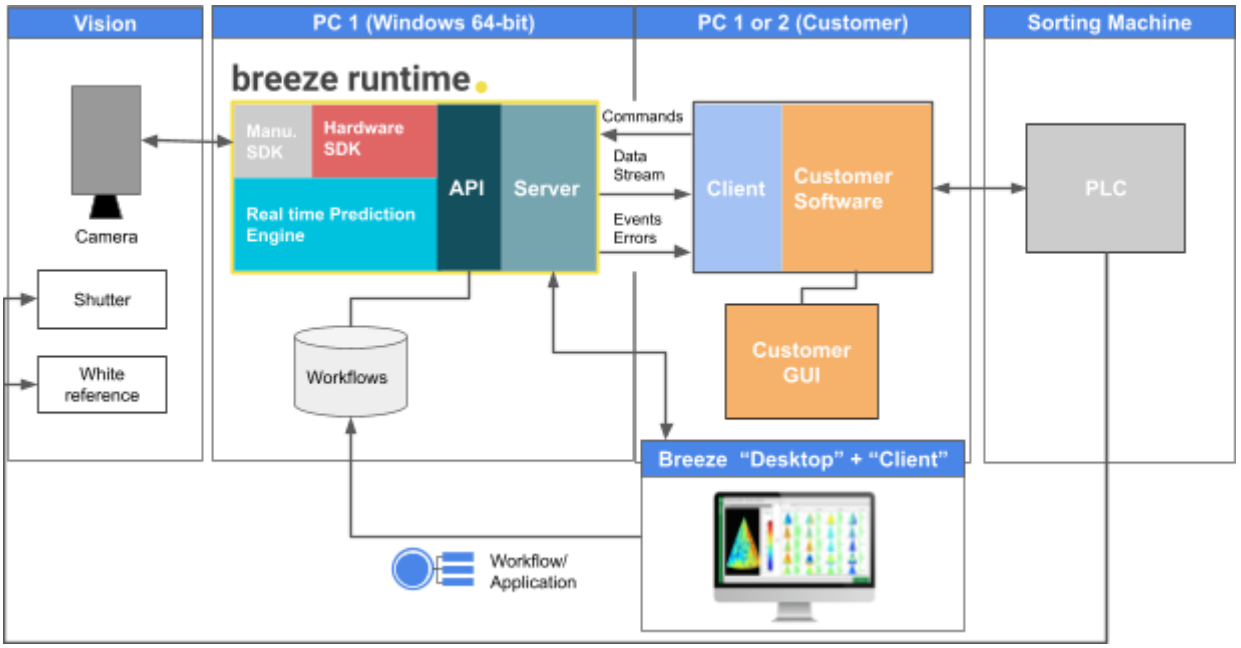


Fig. 1. The system setup example

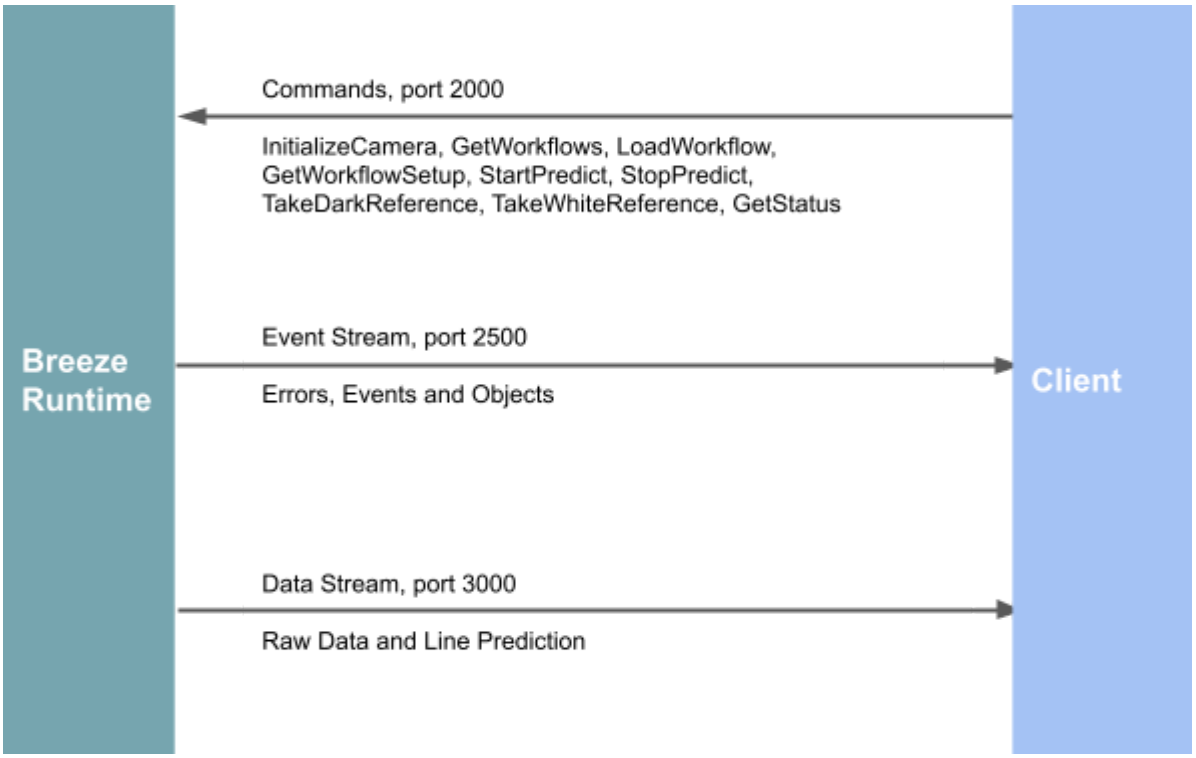


Fig. 2. Communication overview

Commands

The commands and the server's response are sent over TCP/IP on port 2000. The message format is unindented JSON ended with CR + LF (ASCII 13 + ASCII 10).

Note. You can use curl with telnet for communicating with Breeze Runtime and send commands. For example: curl -v telnet://<ip>:<port>

Camera commands

Initialize camera

Using Breeze default settings

```
Message:    {"Command"        :    "InitializeCamera",
             "Id"             :    "IdString"}
```

Optional. Camera specific settings

```
Message:    {"Command"        :    "InitializeCamera",
             "Id"             :    "IdString",
             "DeviceName"     :    "SimulatorCamera",
             "RequestedPort"  :    3000,
             "TimeOut"        :    5 (seconds)
             …
             …
```

```
... }
```

- The optional parameter `RequestedPort` can be used to set the data stream port, which defaults to 3000.
- The optional parameter TimeOut is in seconds. Used in for example when waiting for the camera to become stable.
- The command can have any number of camera specific arguments (see table below).

```
Reply:      {"Id"       :     "IdString",
             "Success"  :     true}
```

Available cameras

Manufacturer	Camera and parameters

Specim	LUMO SDK
	– Swir, FX-10, FX-17, Mwir, Etc.
	
	`DeviceName = "SpecimCamera"`
	`CameraType = E.g. "FX17 with NI" or "FileReader"`
	`ScpFilePath = "<path>\<file>.scp"`

Inno-Spec	Inno-Spec SDK
	
	`DeviceName = "InnoSpecCamera"`
	`CameraType = "RedEye"`
	`MissingPixelsFilePath = "<path>\<file>.xxx" (optional)`

HySpex	HySpex SDK
	- Swir, Vnir, Etc.
	
	`DeviceName = "HySpexCamera"`
	`CameraType = E.g. "VNIR_1800"`
	`FramesToAverage = 1, optional; default = 1`
	`SetFilePath = "<path>\<SetFileDirectory>"`

Basler	Pylon SDK
	
	`DeviceName = "BaslerCamera"`
	`CameraType = E.g. "Basler ac1920-155um"`
	`BinningHorizontal = 1, optional; default = 1`
	`BinningVertical = 1, optional; default = 1`

Detection Technology	Detection Technology - X-ray sensor with Ethernet connection
	
	`DeviceName = "DeeTeeCamera"`
	`CameraType = "X-Scan"`
	`HostIp = "127.0.0.1"`
	`SpatialBinning = 0, optional; default = 0`
	`BufferSize = 100, optional; default = 100`
	`TimeoutExceptionMs = 5000, optional; default = 5000`
	`TimeOut = 20, optional; default = 20`

```
Calibration data
Ob = 52000
DynCalEnable = false
DynCalLines = 100
DynCalThreshold = 0.01
DynCalNbBlockSaved = 10

Geometric calibration
GeoCorrEnable = true
SD = 1160
DLE = 36.411
DHE = 40.600
CenterPixel= 458.5
```

Default values on initialization

Op code	Operation name	Value	Explanation and default
0x22	Operation Mode	0x00	Continuous mode
0x30	Gain Correction	0x00	Disable gain correction
0x31	Offset Correcting	0x00	Disable offset correction
0x32	Baseline Correcting	0x00	Disable baseline correction
0x3B	PDC correction	0x00	Disable PDC correction
0x40	Pixelbinning	0xNN	0x00 or GUI input
0x41	Line averaging, the filter size is 2^n	0x00	n = 0
0x43	Output scale	0x00	Original
0x44	Offset averaging filter	0x00	1 line
0x51	External Line Trigger Status	0x00	Disable external line trigger
0x55	External Frame Trigger Status	0x00	Disable external frame trigger
0x5A	Trigger stamp parity check mode	0x00	Disable parity check
0x60	Period of heartbeat packets	0x3C	Every 60 seconds
0x61	X-GCU type	0x00	Ethernet
0x6A	Test pattern status	0x00	Disable

	<table><tr><td>0x6B</td><td>DM's test mode</td><td>0x00</td><td>Disable</td></tr><tr><td>0x75</td><td>LED lights</td><td>0x00</td><td>Disable</td></tr><tr><td>0x7A</td><td>Pixel order mode</td><td>0x00</td><td>Low from first, High from first</td></tr><tr><td>0x7B</td><td>Energy mode</td><td>0x01</td><td>DE stacked</td></tr><tr><td>0x90</td><td>Offset delay mode</td><td>0x00</td><td>Disable offset delay time setting</td></tr><tr><td>0x91</td><td>Camera link data format</td><td>0x00</td><td>Normal</td></tr></table> After calculating calibration and geometric correction values. <table><tr><td>0x27</td><td>Starts/stops scanning</td><td>0x01</td><td>Start scanning</td></tr></table> See Appendix F: Detection Technology camera

Prediktera	FileReader test camera.
	
	`DeviceName = "SimulatorCamera"`
	
	If only DeviceName is specified, a simulated camera with 9 frames with an image width at 10 pixels will be used. The frames contain one object. The frames 'wrap around'; the 10th frame will be the same as the first one, etc. The `CameraType` will be "SimulatorCamera" (see below).
	
	By specifying the data files used, any raw data stream can be used, e.g.:
	
	`RawDataFilePath = "Data\Nuts\measurement.raw"`
	`DarkReferenceFilePath = "Data\Nuts\darkref_measurement.raw"`
	`WhiteReferenceFilePath = "Data\Nuts\whiteref_measurement.raw"`
	`CameraType = "NutsSimulatorCamera"`
	
	• The file paths are relative to the directory where BreezeRuntime.exe is located.
	
	• The parameter `CameraType` can be any text, and the `CameraType` field in the reply from the `GetStatus` command will be set to this text.
	
	• See Running Breeze Runtime with Breeze Client for instructions on how to download and install the nuts test data, which should be used in conjunction with the workflow with ID "NutsWorkflow".

Generic camera	A camera implementing a JSON-RPC 2.0 standard, such as Specim FX-50.
	
	`DeviceName = "GenericeCamera"`
	`CameraType = "Cortex"`
	`HostIp = "127.0.0.1"`
	`TimeOut = 5, optional; default = 5`
	`TimeoutExceptionMs = 5000, optional; default = 5000`
	`InitializeTimeOut = 5, optional; default = TimeOut`
	`StableTimeOut = 5, optional; default = TimeOut`

```
PreprocessingTimeOut = 5, optional; default = TimeOut
BinningSpatial = 1, optional; default = 1
BinningSpectral = 1, optional; default = 1
BufferSize = 100, optional; default = 100
```

Default values on initialization

Op name	Default value
`preprocessing.enabled`	true if false else no action
`preprocessing.batch.frames`	1
`processing.batch.frames`	1
`camera.binning.spatial`	1 or GUI input
`camera.binning.spectral`	1 or GUI input

See Appendix E: Generic camera

Get camera property

```
Message:    {"Command"   :    "GetCameraProperty",
             "Id"        :    "IdString"}
             "Property"  :    "FrameRate"}

Reply:      {"Id"        :    "IdString",
             "Success"   :    true,
             "Message"   :    "100"
```

Below is a list of properties, which all logical cameras have. Some physical cameras cannot be queried for some properties though, and a default value will be returned by the logical camera. Some logical cameras have properties not listed here, but that can still be retrieved. E.g. The camera “SimulatorCamera” has the property `UniqueProperty` (Float, 32 bits) to demonstrate this feature. It also has the property “State”, which can be used for simulating dark and white references (see commands TakeDarkReference and TakeWhiteReference).

Property	Original Data Type	Comment
`IntegrationTime`	Float, 32 bits	µs
`FrameRate`	Float, 32 bits	Hz
`IsCapturing`	Boolean	`true` or `false`
`ImageWidth`	Integer, 32 bits	No of pixels
`ImageHeight`	Integer, 32 bits	No of pixels
`Wavelengths`	string	E.g. `"300;300.3;300.6;300.9;301.2…"`

`MaxSignal`	Integer, 32 bits	Maximum signal value
`Temperature`	Float, 32 bits	Sensor temperature (K). Celcius degrees = `Temperature` - 273.15. Fahrenheit degrees = `Temperature` x 1.8 - 459.67.
`DataSize`	Integer	1 = Byte, 2 = Short (Integer, 16 bits), 4 = Float (Float, 32 bits), 8 = Double (Float, 64 bits)
`Interleave`	Integer	1 = BIL (Band Interleaved by Line) 2 = BIP (Band Interleaved by Pixel)

Set camera property

```
Message:    {"Command"   :     "SetCameraProperty",
             "Id"        :     "IdString"}
             "FrameRate":      "200"}

Reply:      {"Id"        :     "IdString",
             "Success"   :     true,
             "Message"   :     "200"
```

Below is a list of properties, which all logical cameras have. Some physical cameras cannot have some properties set though, and the set operation will not have any effect. Some logical cameras have properties not listed here, but that can still be set. E.g. the camera “SimulatorCamera” has the property `UniqueProperty` (Float, 32 bits) to demonstrate this feature. It has also the property “State”, which can be used when simulating dark and white references (see commands TakeDarkReference and TakeWhiteReference).

Property	Original Data Type	Comment
`IntegrationTime`	Float, 32 bits	µs
`FrameRate`	Float, 32 bits	Hz

Raw-data capture commands

Start capture

```
Message:    {"Command"         :     "StartCapture",
             "Id"              :     "IdString",
             "NumberOfFrames"  :     10,
             "Folder"          :     "Path to folder”} // Optional
```

The parameter `NumberOfFrames` is optional; if omitted, the camera will capture until `StopCapture` is sent.
The parameter `Folder` is optional; if omitted, the camera will store the captured frames in a measurement.raw file in the chosen folder.

```
Reply:      {"Id"       :     "IdString",
             "Success"  :     true,
             "Message"  :     ""}
```

Stop capture

```
Message:    {"Command"  :     "StopCapture",
             "Id"       :     "IdString"}

Reply:      {"Id"       :     "IdString",
             "Success"  :     true,
             "Message"  :     ""}
```

Prediction commands

Get workflows

```
Message:    {"Command"              :     "GetWorkflows",
             "Id"                   :     "IdString"}
             "IncludeTestWorkflows" :     true}
```

The parameter `IncludeTestWorkflows` specifies whether the test workflows bundled with Breeze Runtime (see below) should be included; If the parameter is `false` or omitted, only the workflows created by the user with Breeze are included.

```
Reply:      {"Id"       :     "IdString",
             "Success"  :     true,
             "Message"  :     "
            [
              {
                "Name": "Test Workflow",
                "Id": "TestWorkflow",
                "Description": "Powder Quantification 10 pixels x 9 lines Test
      Sample",
                "CreatedTime": "20180325160219",
                "CreatedBy": "Administrator",
                "PredictionMode": "Normal",
                "ObjectFormat": {
                  "Id": "12345",
                  "Name": "spectral_sample",
                  "Descriptors": [
                    {
                      "Type": "Category",
                      "Name": "Type",
                      "Index": 0,
                      "Id": "ebf126aa",
                      "Classes": [
                        {
                          "Name": "-",
```

```
            "Color": "#ff0000",
            "Value": 0
          },
          {
            "Name": "V",
            "Color": "#3ad23a",
            "Value": 1
          },
          {
            "Name": "P",
            "Color": "#4664be",
            "Value": 2
          },
          {
            "Name": "B",
            "Color": "#f6f76d",
            "Value": 3
          }
        ]
      },
      {
        "Type": "Property",
        "Name": "B",
        "Index": 1,
        "Id": "28987009"
      },
      {
        "Type": "Property",
        "Name": "V",
        "Index": 2,
        "Id": "78f02a2b"
      },
      {
        "Type": "Property",
        "Name": "P",
        "Index": 3,
        "Id": "1c6d9580"
      }
    ]
  },
  "StreamFormat": {
    "TimeFormat": "Utc100NanoSeconds",
    "Lines": [
      {
        "Type": "Category",
        "Name": "SampleCategory",
        "Index": 0,
        "Id": "aa533a79",
        "Classes": [
          {
            "Name": "-",
```

```
      "Color": "#ff0000",
      "Value": 0
    },
    {
      "Name": "Sample",
      "Color": "#3ad23a",
      "Value": 1
    }
  ]
},
{
  "Type": "Category",
  "Name": "Type",
  "Index": 1,
  "Id": "ebf126aa",
  "Classes": [
    {
      "Name": "-",
      "Color": "#ff0000",
      "Value": 0
    },
    {
      "Name": "V",
      "Color": "#3ad23a",
      "Value": 1
    },
    {
      "Name": "P",
      "Color": "#4664be",
      "Value": 2
    },
    {
      "Name": "B",
      "Color": "#f6f76d",
      "Value": 3
    }
  ]
},
{
  "Type": "Property",
  "Name": "B",
  "Index": 2,
  "Id": "28987009"
},
{
  "Type": "Property",
  "Name": "V",
  "Index": 3,
  "Id": "78f02a2b"
},
{
```

```
          "Type": "Property",
          "Name": "P",
          "Index": 4,
          "Id": "1c6d9580"
        }
      ]
    }
  },
  {
    "Name": "Nuts Workflow",
    "Id": "NutsWorkflow",
    "Description": "This is a workflow with nuts and shells",
    "CreatedTime": "20180404145346",
    "CreatedBy": "Administrator",
    "PredictionMode": "Normal",
    "ObjectFormat": {
      "Id": "12345",
      "Name": "Sample - Nuts_Classification",
      "Descriptors": [
        {
          "Type": "Category",
          "Name": "Nut or shell",
          "Index": 0,
          "Id": "80c6940d",
          "Classes": [
            {
              "Name": "-",
              "Color": "#ff0000",
              "Value": 0
            },
            {
              "Name": "Nut",
              "Color": "#3ad23a",
              "Value": 1
            },
            {
              "Name": "Shell",
              "Color": "#4664be",
              "Value": 2
            }
          ]
        }
      ]
    },
    "StreamFormat": {
      "TimeFormat": "Utc100NanoSeconds",
      "Lines": [
        {
          "Type": "Category",
          "Name": "SampleCategory",
          "Index": 0,
```

```
          "Id": "5dae874a",
          "Classes": [
            {
              "Name": "-",
              "Color": "#ff0000",
              "Value": 0
            },
            {
              "Name": "Sample",
              "Color": "#3ad23a",
              "Value": 1
            }
          ]
        },
        {
          "Type": "Category",
          "Name": "Nut or shell",
          "Index": 1,
          "Id": "80c6940d",
          "Classes": [
            {
              "Name": "-",
              "Color": "#ff0000",
              "Value": 0
            },
            {
              "Name": "Nut",
              "Color": "#3ad23a",
              "Value": 1
            },
            {
              "Name": "Shell",
              "Color": "#4664be",
              "Value": 2
            }
          ]
        }
      ]
    }
  },
  {
    "Name": "Object To Pixels Pipeline Workflow",
    "Id": "ObjectToPixelsPipelineWorkflow",
    "Description": "Object To Pixels Pipeline Mode Workflow",
    "CreatedTime": "20180831131648",
    "CreatedBy": "Administrator",
    "PredictionMode": "ObjectToPixelsPipeline",
    "StreamFormat": {
      "TimeFormat": "Utc100NanoSeconds",
      "Lines": [
        {
```

```
            "Type": "Category",
            "Name": "Type",
            "Index": 0,
            "Id": "8a614f86",
            "Classes": [
              {
                "Name": "Background",
                "Color": "#4664be",
                "Value": 0
              },
              {
                "Name": "Sample",
                "Color": "#3ad23a",
                "Value": 1
              },
              {
                "Name": "Unknown",
                "Color": "#ff0000",
                "Value": 2
              },
              {
                "Name": "A",
                "Color": "#3ad23a",
                "Value": 3
              },
              {
                "Name": "B",
                "Color": "#4664be",
                "Value": 4
              },
              {
                "Name": "C",
                "Color": "#f6f76d",
                "Value": 5
              },
              {
                "Name": "D",
                "Color": "#73e1fa",
                "Value": 6
              },
              {
                "Name": "Unknown Sample",
                "Color": "#ff0000",
                "Value": 7
              }
            ]
          }
        ]
      }
    }
  ]"}
```

Load workflow

Loads a workflow to use for predictions

```
Message:    {"Command"        :     "LoadWorkflow",
             "Id"             :     "IdString",
             "WorkflowId"     :     "TestWorkflow",
             "Xml"            :     null, (can be omitted)
             "FieldOfView"    :     10.5}
```

- To load a **server** side workflow, the parameter `WorkflowId` is set to one of the IDs in the reply from `GetWorkflows`.
- To load a **client** side workflow, the parameter `Xml` is set to the contents of a client side workflow xml file.
- Either `WorkflowId` or `Xml` must be specified, but not both.
- `FieldOfView` is optional. if omitted, the value for the current camera will be used from settings in Breeze on the local computer.

```
Reply:{"Id"       :     "IdString",
             "Success"   :     true,
             "Message"   :     "
                  {
                    "Name": "Test Workflow",
                    "Id": "TestWorkflow",
                    "Description": "This is a test workflow",
                    "CreatedTime": "20180325160219",
                    "CreatedBy": "Administrator",
                    "PredictionMode": "Normal",
                    "ObjectFormat": {
                      "Id": "12345",
                      "Name": "spectral_sample",
                      "Descriptors": [
                        {
                          "Type": "Category",
                          "Name": "Type",
                          "Index": 0,
                          "Id": "ebf126aa",
                          "Classes": [
                            {
                              "Name": "-",
                              "Color": "#ff0000",
                              "Value": 0
                            },
                            {
                              "Name": "V",
                              "Color": "#3ad23a",
                              "Value": 1
                            },
                            {
```

```
            "Name": "P",
            "Color": "#4664be",
            "Value": 2
          },
          {
            "Name": "B",
            "Color": "#f6f76d",
            "Value": 3
          }
        ]
      },
      {
        "Type": "Property",
        "Name": "B",
        "Index": 1,
        "Id": "28987009"
      },
      {
        "Type": "Property",
        "Name": "V",
        "Index": 2,
        "Id": "78f02a2b"
      },
      {
        "Type": "Property",
        "Name": "P",
        "Index": 3,
        "Id": "1c6d9580"
      }
    ]
  },
  "StreamFormat": {
    "TimeFormat": "Utc100NanoSeconds",
    "LineWidth": 10,
    "Lines": [
      {
        "Type": "Category",
        "Name": "SampleCategory",
        "Index": 0,
        "Id": "aa533a79",
        "Classes": [
          {
            "Name": "-",
            "Color": "#ff0000",
            "Value": 0
          },
          {
            "Name": "Sample",
            "Color": "#3ad23a",
            "Value": 1
          }
```

```
        ]
      },
      {
        "Type": "Category",
        "Name": "Type",
        "Index": 1,
        "Id": "ebf126aa",
        "Classes": [
          {
            "Name": "-",
            "Color": "#ff0000",
            "Value": 0
          },
          {
            "Name": "V",
            "Color": "#3ad23a",
            "Value": 1
          },
          {
            "Name": "P",
            "Color": "#4664be",
            "Value": 2
          },
          {
            "Name": "B",
            "Color": "#f6f76d",
            "Value": 3
          }
        ]
      },
      {
        "Type": "Property",
        "Name": "B",
        "Index": 2,
        "Id": "28987009"
      },
      {
        "Type": "Property",
        "Name": "V",
        "Index": 3,
        "Id": "78f02a2b"
      },
      {
        "Type": "Property",
        "Name": "P",
        "Index": 4,
        "Id": "1c6d9580"
      }
    ],
  }
}"}
```

- `CreatedTime` has the format `yyyyMMddhhmmss`.

Delete workflow

```
Message:    {"Command"          :     "DeleteWorkflow",
             "WorkflowId"       :     "IdString"}
```

Open shutter

```
Message:    {"Command"          :     "OpenShutter",
             "Id"               :     "IdString"}
```

Opens the camera shutter.

```
Reply:{"Id"         :     "IdString",
             "Success"    :     true,
             "Message"    :     ""}
```

Close shutter

```
Message:    {"Command"          :     "CloseShutter",
             "Id"               :     "IdString"}
```

Closes the camera shutter.

```
Reply:{"Id"         :     "IdString",
             "Success"    :     true,
             "Message"    :     ""}
```

Take dark reference

```
Message:    {"Command"          :     "TakeDarkReference",
             "Id"               :     "IdString"}
```

If the camera “SimulatorCamera” is used (see command `Initialize camera`), the camera state must be set to `DarkReference` first; either use the command `CloseShutter`, or use the command `SetProperty` to set `State` to `DarkReference`. When the shutter is opened the state is set to `Normal`.

```
Reply:      {"Id"         :     "IdString",
             "Success"    :     true,
             "Message"    :     ""}
```

Dark reference quality control

Automatic dark reference quality control checks

- Standard deviation between lines is lower than 5%
- Average signal relative to max signal is lower than 50%

Invalid dark reference reply example:

```
Reply:      {"Id"        :     "IdString",
             "Success"   :     false,
             "Code"      :     1006
             "Error"     :     "InvalidDarkReference"
             "Message"   :     "Variation over lines is higher than 5%"}
```

Take white reference

```
Message:    {"Command"         :     "TakeWhiteReference",
             "Id"              :     "IdString"}
```

If the camera “SimulatorCamera” is used (see command `InitializeCamera`), the camera property `State` must be set to `WhiteReference` first, using the command `SetCameraProperty`. After the reference is taken the state can be set to `Normal`.

```
Reply:      {"Id"        :     "IdString",
             "Success"   :     true,
             "Message"   :     ""}
```

White reference quality control

Automatic white reference quality control checks

- Standard deviation between lines is lower than 5%
- Average signal relative to max signal is lower than 99%
- Average signal relative to max signal is higher than 50%
- Standard deviation between average pixels is lower than 5%
  Note. 10% of border pixels on each side is not included in the calculation

Invalid white reference reply example:

```
Reply:      {"Id"        :     "IdString",
             "Success"   :     false,
             "Code"      :     1007
             "Error"     :     "InvalidWhiteReference"
             "Message"   :     "White reference less than 50% of max signal"}
```

Get workflow setup

Should be called to retrieve prediction setup from the active workflow, which another client loaded.

```
Message:    {"Command"   :     "GetWorkflowSetup",
             "Id"        :     "IdString"}
```

`Reply:` See command `LoadWorkflow`.

Start predict

```
Message:    {"Command"              :     "StartPredict",
             "Id"                   :     "IdString",
             "IncludeObjectShape"   :     true,
             "FrameCount"           :     -1}

Reply:      {"Id"       :     "IdString",
             "Success"  :     true,
             "Message"  :     ""}
```

Parameters:

- IncludeObjectShape (optional, defaults to false): Whether start y offset, object center, and border coordinates should be included with object identified event.

The identified objects are sent over the event stream (please see section Event Stream). This is an example object:

```
{
 "Event"    : "PredictionObject",
 "Code"     : 4000,
 "Message"  : "
      %7B%22StartTime%22%3A636803791584234611%2C%22EndTime%22%3A63680379158431
      4525%2C%22Descriptors%22%3A%5B1.0%2C45.4867249%2C47.89075%2C6.40655136%5
      D%2C%22Shape%22%3A%7B%22Start%22%3A1%2C%22Center%22%3A%5B5%2C3%5D%2C%22B
      order%22%3A%5B%5B3%2C0%5D%2C%5B8%2C0%5D%2C%5B8%2C1%5D%2C%5B9%2C1%5D%2C%5
      B9%2C7%5D%2C%5B3%2C7%5D%2C%5B3%2C3%5D%2C%5B2%2C3%5D%2C%5B2%2C1%5D%2C%5B3
      %2C1%5D%2C%5B3%2C0%5D%5D%7D%7D"
}
```

Message field unescaped and indented:

```
{
  "StartTime": 636803787957020338,
  "EndTime": 636803787957110238,
  "Descriptors": [
    1.0,
    45.4867249,
    47.89075,
    6.40655136
  ],
  "Shape": {
    "Start": 1,
    "Center": [5, 3],
    "Border": [
      [3, 0],
      [8, 0],
      [8, 1],
      [9, 1],
```

```
      [9, 7],
      [3, 7],
      [3, 3],
      [2, 3],
      [2, 1],
      [3, 1],
      [3, 0]
    ]
  }
}
```

Stop predict

```
Message:    {"Command"   :     "StopPredict",
             "Id"        :     "IdString"}

Reply:      {"Id"        :     "IdString",
             "Success"   :     true,
             "Message"   :     ""}
```

Get status

```
Message:    {"Command"   :     "GetStatus",
             "Id"        :     "IdString"}

Reply:      {"Id"        :                    "IdString",
             "Success"   :                    true,
             "Message"   :
               "{"State":                     "Predicting",
                 "WorkflowId":                "abcde",
                 "CameraType":                "MWIR",
                 "FrameRate":                 200.0,          (Hz)
                 "IntegrationTime":           1500.0,         (µs)
                 "Temperature":               293.15,         (K)
                 "DarkReferenceValidTime":    4482.39258,     (s)
                 "WhiteReferenceValidTime":   9483.392,       (s)
                 "LicenseExpiryDate":         "2018-12-31",   (yyyy-MM-dd)
                 "SystemTime":                636761502932108549,
                 "SystemTimeFormat":          "Utc100NanoSeconds"}"
```

- The original data type for `SystemTime` is an unsigned 64-bit integer.

States:

- `Idle`
- `LoadingWorkflow`
- `Predicting`
- `StoppingPrediction`
- `CapturingRawPixelLines`
- `CapturingMultipleFrames`

Please note that the field `Temperature` represents the camera **sensor** temperature in **Kelvin** degrees. Celsius degrees = `Temperature` - 273.15. Fahrenheit degrees = `Temperature` x 1.8 - 459.67.

Get property

```
Message:    {"Command"   :     "GetProperty",
             "Id"        :     "IdString",
             "Property"  :     "PropertyString",

Optional:
            "NodeId"     :     "NodeIdString",
            "Name":     "FieldName"}

Reply:      {"Id"        :                    "IdString",
             "Success"   :                    true,
             "Message"   :                    "result string"}
```

The field `NodeId` can be used to get property information from a node in the analysis tree. If it is not specified then workflow global properties will be fetched.

Available Properties are:

- `Version`
- `State`
- `WorkspacePath`
- `WorkflowId`
- `DarkReferenceValidTime`
- `WhiteReferenceValidTime`
- `DarkReferenceFile`
- `WhiteReferenceFile`
- `LicenseExpiryDate`
- `SystemTime`
- `SystemTimeFormat`
- `PredictorThreads`
  - Return number of threads used in prediction.
    Default -1 = number of available virtual cpu cores
- `AvailableCameraProviders`
  - Returns list of semicolon separated available providers
- `Fields`
  - Returns list of semicolon separated workflow fields
- `FieldValue`
  - Requires “`Name`”
  - Workflow field value

Example 1

```
Message:    {"Command"   :     "GetProperty",
             "Id"        :     "1",
             "Property"  :     "Version"}
```

```
Reply:      {"Id"       :                   "1",
             "Success"  :                   true,
             "Message"  :                   "2021.1.0.999 Release"}
```

Example 2

```
Message:    {"Command"  :     "GetProperty",
             "Id"       :     "2",
             "Property" :     "FieldValue",
             "Name"     :     "Voltage"}

Reply:{"Id"       :     "2",
             "Success"  :     true,
             "Message"  :     "100"}
```

Set property

```
Message:    {"Command"  :     "SetProperty",
             "Id"       :     "IdString",
             "Property" :     "PropertyString",
             "Value"    :     "ValueString"

Reply:      {"Id"       :                   "IdString",
             "Success"  :                   true,
             "Message"  :                   "result string"}
```

Available Properties to set are:

- `PredictorThreads`
  - Number of threads used in prediction.
    Default -1 = number of available virtual cpu cores

Re-Initialize prediction

If there is a camera problem during prediction then Initialize command can be called to reconnect the camera and if successful start the prediction again. The tries parameter decides the number of tries that will be done before returning the error code CameraNotStable:1009 error code.

Re-Initialize command sequence order

- If predicting before Re-Initialize command was called
  - StopPredict
- For number of tires until camera is initialized
  - DisconnectCamera
  - InitializeCamera
- If predicting before Re-Initialize command was called
  - LoadWorkflow *(Same workflow as loaded before)*
  - StartPredict

```
Message:    {"Command"              :     "Initialize",
```

```
              "Id"                    :     "IdString"},
              "Tries"                 :     10,
              "TimeBetweenTrialSec    :     10}

Reply:       {"Id"        :     "IdString",
              "Success"   :     true,
              "Message"   :     ""}
```

Start capture on predict

Start capture measurements while predicting. The measurements are stored in the folder: {Breeze workspace folder}/Data/Runtime/Measurements/{date}
Capture folders are named by a date when starting the capture. The recorded measurements stored in that folder are divided by the max number count argument. The measurements are called “Measurement_1”, “Measurement_2”,...“Measurement_N”.

```
Message:     {"Command"         :     "StartCaptureOnPredict",
              "Id"              :     "IdString"},
              "Name"            :     "Name of measurements”,
              "MaxFrameCount    :     1000 // Max number of frames per
             measurement
              "Object"          :     true / false} // Save only object or not

Reply:       {"Id"        :     "IdString",
              "Success"   :     true,
              "Message"   :     ""}
```

Stop capture on predict

```
Message:     {"Command"         :     "StopCaptureOnPredict",
              "Id"              :     "IdString"}

Reply:       {"Id"              :     "IdString",
              "Success"         :     true,
              "Message"         :     ""}
```

Error Handling and Events

Errors can happen in two different ways: Either as a result of a command or as an event.

Command Errors

When a command fails, the `Success` field will be `false`, the `Error` field will hold a string error code and the `Code` field will hold an integer error code. The `Message` field will hold a further error description:

```
Reply:      {"Id"       :     "IdString",
            "Success"   :     false,
            "Message"   :     "Camera is not initialized",
            "Error"     :     "GeneralError",
            "Code":     3000}
```

When an error occurs on the server side during prediction or capture, the error information is sent via the event stream. Log files can be found in the directory ".Evince" in the home directory , e.g. "C:\Users\<user name>\.Evince.

Command Error Codes

These will be sent in response to a failed command.

Error Code	Constant	Description
1000	GeneralCommandError	General command error. Message field gives more details.

3001	UnknownError	Unexpected error in runtime command call. Stack trace gives more details.
		
		`{`
		`  "Id"  : "s94kl34",`
		`  "Success": false,`
		`  "Event"  : "Error",`
		`  "Error"  : "UnknownError",`
		`  "Code"  : 3001,`
		`  "Message": "ArrayIndexOutOfBoundsException",`
		`  "StackTrace": "At RtPredict.cs:281 ...\nAt ..."`
		`}`

1002	InvalidLicense	Sent on StartCapture if no valid license is found for Breeze. Sent on StartPredict if no valid license is found for BreezeAPI.

1003	MissingReferences	Sent on StartPredict if references are needed and both references are missing.
1004	MissingDarkReference	Sent on StartPredict if references are needed and dark reference is missing.
1005	MissingWhiteReference	Sent on StartPredict if references are needed and white reference is missing.
1006	InvalidDarkReference	Sent on TakeDarkReference if the captured dark reference is not good enough. Message will include detailed explanation.
1007	InvalidWhiteReference	Sent on TakeWhiteReference if the captured white reference is not good enough. Message will include a detailed explanation.
1008	MissingDarkReferenceFile	Sent on TakeWhiteReference if white reference **intensity** is used and there is no dark reference file to read.
1009	CameraNotStable	Sent on Initialize when cannot reinitialize prediction after given number tries

Event Stream

Events and errors that do not belong to a command will be sent over TCP/IP on port 2500. The message format is unindented JSON ended with CR + LF (ASCII 13 + ASCII 10). All errors, except those in command responses, will also be sent over this channel.

Regular Event Codes

These events can be sent during startup, prediction and capture.

Example

```
Message:    {"Event"    :    "DarkReferenceOutdated",
             "Code"     :    2001}
```

Event Code	Constant	Description
2000	*reserved for event server*	
2001	DarkReferenceOutdated	Dark reference is outdated
2002	WhiteReferenceOutdated	White reference is outdated
2003	WorkflowListChanged	Workflow in Runtime folder has been changed, removed or added

2004	WorkflowLoaded	Sent when a workflow is loaded

Error Event Codes

These events can be sent during startup, prediction and capture. An error event will have the field Event = “Error” and the field “Error” specifies the name of the error.

Example

```
Message:    {"Event"     :     "Error",
             "Error"     :     "UnknownError"
             "Message"   :     "<message>"
             "Code"      :     3001,
             "StackTrace":     "At RtPredict.cs:281..."}
```

Error Code	Constant	Description
3000	GeneralError	General error. Message field gives more details.

3001	UnknownError	Unexpected error during prediction or capture. Message will include error details
		
		`{`
		`  "Event"  : "UnknownError",`
		`  "Code"   : 3001,`
		`  "Message": "ArrayIndexOutOfBoundsException",`
		`  "StackTrace": "At RtPredict.cs:281 ...\nAt ..."`
		`}`

3002	CameraErrorCode	Forwards error code generated by camera supplier. A new field CameraErrorCode will be present in event. Example:
		
		`{`
		`"Event"    : "CameraErrorCode",`
		`"Code"     : 3002,`
		`"Message"  : "Camera not streaming"`
		`}`

3003	FrameQueueOverflow	Prediction frame queue overflow. Please use a lower camera frame rate.

Prediction Object

Event Code	Constant	Description
4000	PredictionObject	The Runtime has identified an object. Its properties can be found in the message field as escaped JSON. Please see Start predict for details.

		{
		"Event" : "PredictionObject",
		"Code" : 4000,
		"Message" : "<Escaped JSON>"
		}

Data Stream

The data stream from the server is sent over TCP/IP, and the port defaults to 3000, but can be set via the `InitializeCamera` command. Please note that all integers and floats are little-endian encoded in the stream.

Header

[Stream Type][Frame Number][Timestamp][Metadata size][Data Body Size]

1. Stream Type: 1 = Raw Pixel Line
   2 = Prediction Lines
   3 = Rgb Pixel Line
   4 = StreamStarted/EndOfStream

   Data Type: Byte

2. Frame Number: Frame number from camera
   Data Type: Integer, 64 bits

3. Timestamp: Time in 100-nanoseconds when line is read from camera
   (Number of 100-nanosecond intervals that have elapsed since 12:00:00 midnight on January 1, 0001, UTC)

   Data Type: Integer, 64 bits

4. Metadata Size: Number of bytes in metadata body

   Data Type: Integer, 32 bits

5. Data Body Size: Number of bytes in data body

   Data Type: Integer, 32 bits

Metadata Body: See below.

Data Body: See below.

Metadata body

1. CameraProcessingTime: Camera Processing Time, in 100-nanoseconds

   Data Type: Integer, 32 bits

2. CameraDeltaTimeToPreviousFrame:

   Time difference between frames on camera, in 100-nanoseconds

   Data Type: Integer, 32 bits

3. BreezeProcessingTime: Breeze Processing Time, in 100-nanoseconds

   Data Type: Integer, 32 bits

4. BreezeDeltaTimeToPreviousFrame:

   Time difference between frames in Breeze, in 100-nanoseconds

   Data Type: Integer, 32 bits

Data body

Raw Pixel Line

Stream Type = 1
Time in header represents the time when the raw data was received from the camera.
Data Body: Raw bytes

Prediction Lines

Stream Type = 2
Time in header represents the time when the frame raw data was received from the camera.

Data Body:
- Line1:
  - Array: The number of elements in the array equals the `LineWidth` field in the reply from the `LoadWorkflow` command.
  - Data Type: See below.
- Line 2
- …
- Line N

Classification Vector

Range: 0 (No class) - Number of classes (255)
Example 1: [0,0,0,0,1,1,1,1,0,0,0,2,2,2,2,0,0,0,0,0,3,3,3,0,0,...]
Example 2: [0,0,1,1,1,0,1,1,0,0,0,1,1,1,1,0,0,0,0,0,1,1,1,0,0,...] (sample vector)
Data Type: Byte

Quantification Vector

Range: -Max float - Max float
Example: [0,0,0,0,4.5,5.2,3.8,6.5,0,0,0,...]
Data Type: Float, 32 bits

Confidence Values

Range: 5-1 (High - Low)
Example: [0,0,0,0,3,3,3,3,0,0,0,5,5,5,5,0,0,0,0,0,1,1,1,0,0,...]
Data Type: Byte

Rgb Pixel Line

Range: R = 0 - 255, G = 0 - 255, B = 0 - 255
Example: [50,100,150],[50,45.55],[50.20,35],[85,65.255],[0,0,0],[...]
Data Type: 3 x Byte

Stream Start / End

Stream Type = 4
Time in header represents the time when the data was sent.
Data Body are these ASCII encoded strings:

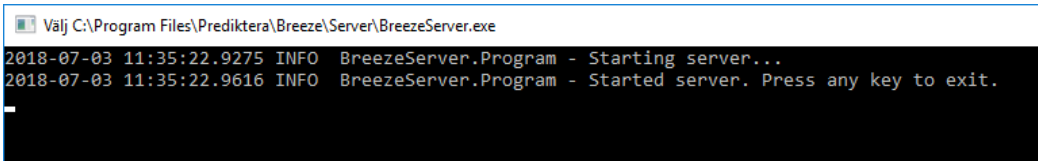
- `StreamStarted`
- `EndOfStream`

Runtime Command Switches

Switch	Description
/p:<n>, e.g. “/p:16”	Set no of calculation threads
/w:<Current Workspace Path>, e.g. /w:”C:\temp\My Folder”	Override current workspace path

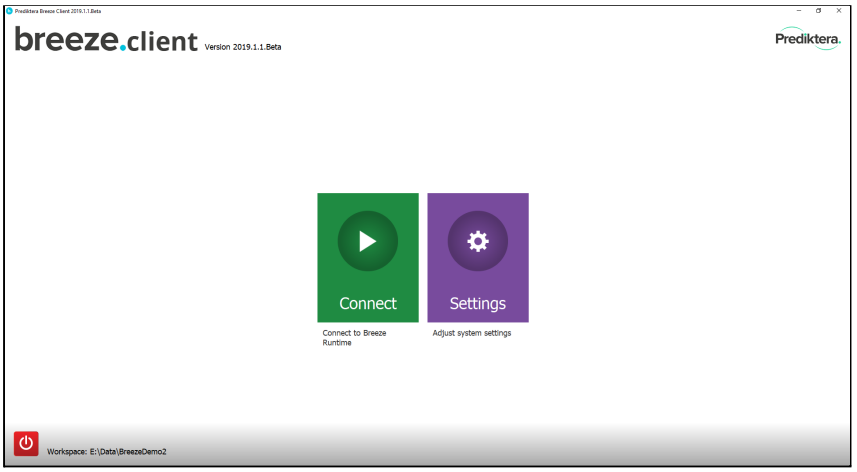
Running Breeze Runtime with Breeze Client

1. Start “BreezeRuntime.exe”. *Note: Located under “C:\Program Files\Prediktera\Breeze\Runtime”*. If the optional event server is used (see Event Stream), start “BreezeRuntime.exe ***/e***” instead.

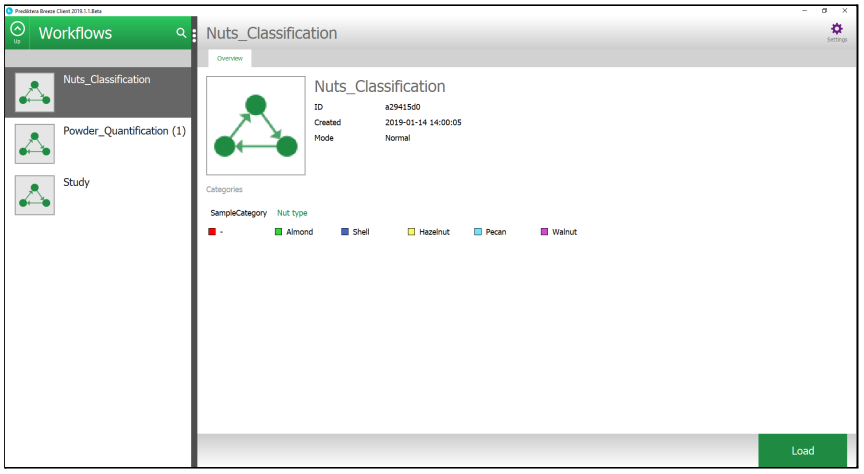


2. Start “BreezeClient.exe” from Start menu

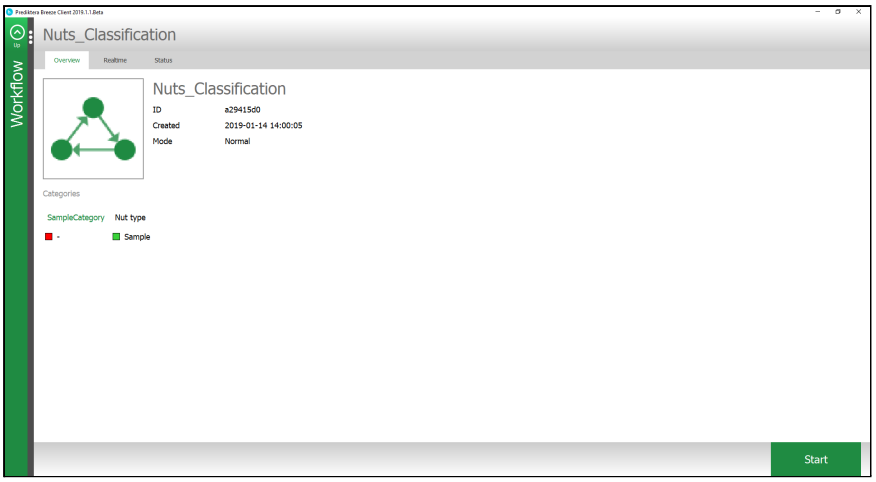
3. Press the “Connect” button.



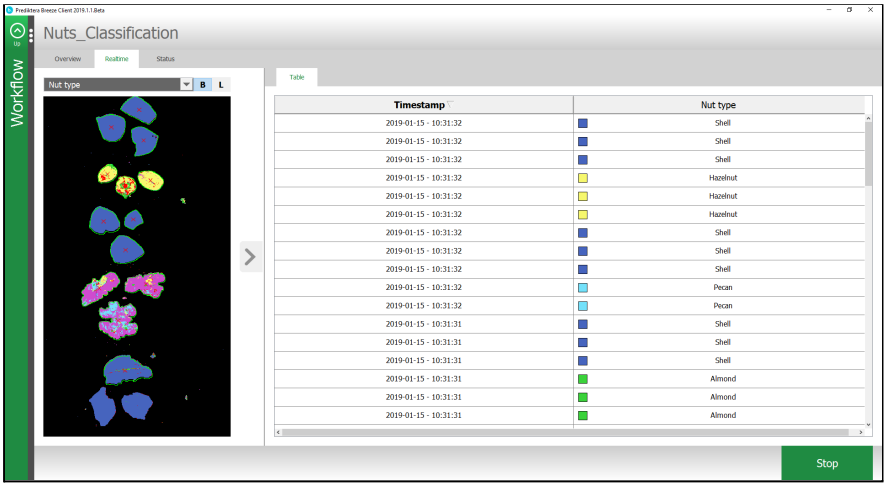
4. Select workflow exported from Breeze and press “Load”



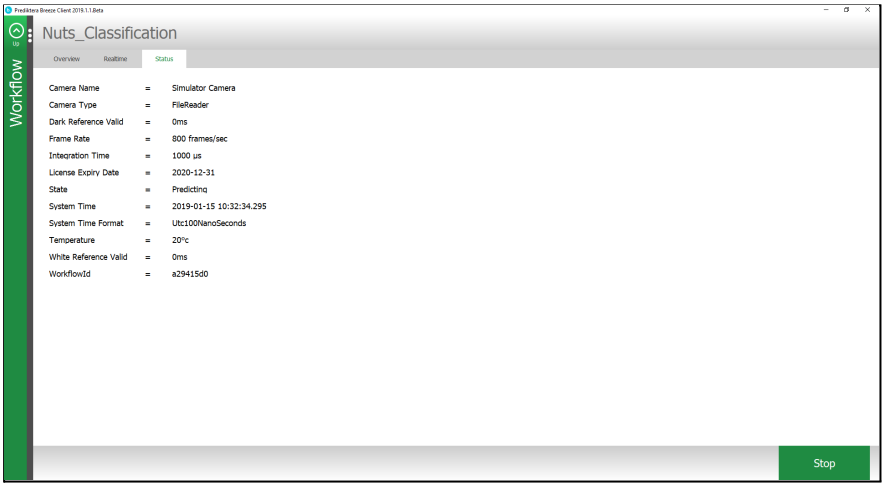
5. Start predicting on the loaded workflow by pressing “Start”



6. View real time predictions under the “Realtime” tab

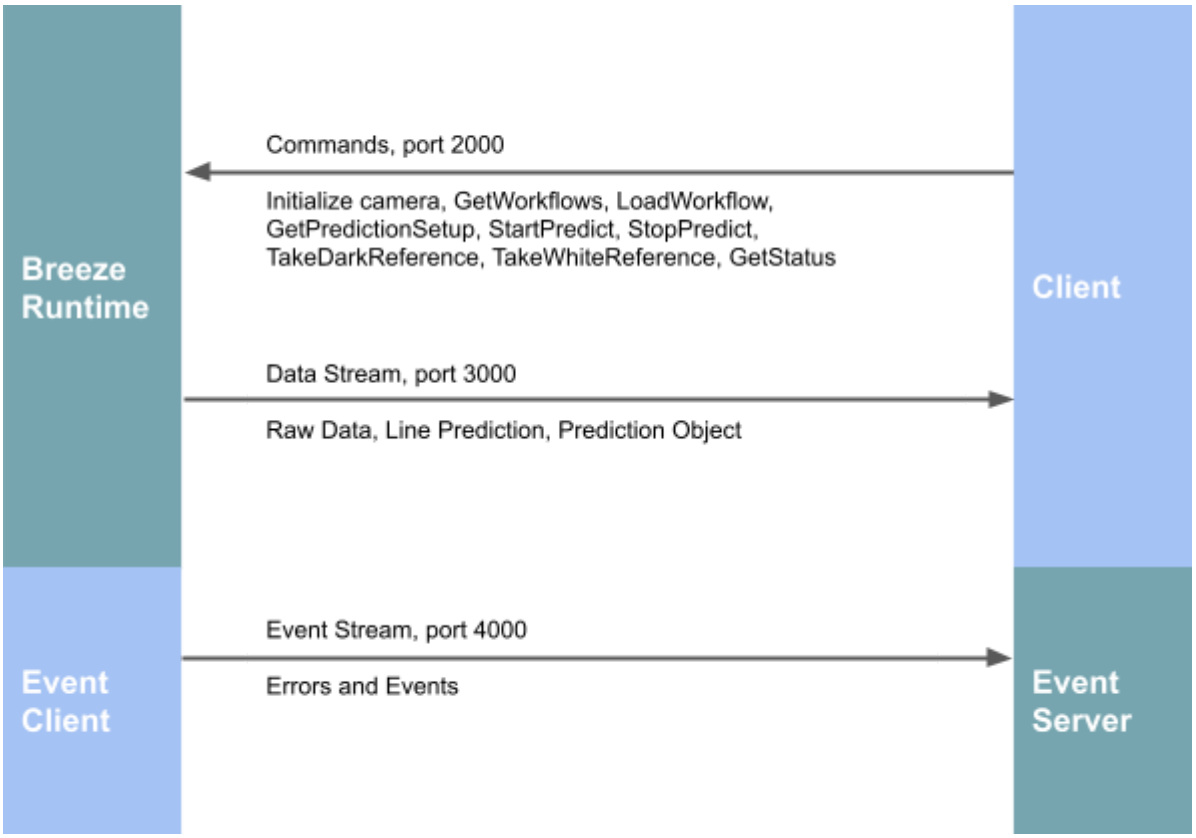


7. View Breeze Runtime status under the “Status” tab



Appendix A: Event Server

Breeze Runtime can be used in conjunction with an **optional** event server. If BreezeRuntime.exe is started with a ”/e” argument, events and errors from Breeze Runtime will be sent over this separate channel. Breeze Runtime will act as the client to the event server in this case. Note that all errors except those in command responses will be sent over this channel. The message format is unindented JSON ended with CR + LF (ASCII 13 + ASCII 10).



When Breeze Runtime is started it will repeatedly try to connect to the event server. If the event server disconnects it will start to try to connect again.

There are two optional arguments to the /e switch, where the IP address or IP address + port can be specified, e.g. “/e:127.0.0.1:4000” or “/e:127.0.0.1”. The default IP address is 127.0.0.1 and the default port is 4000.

Time format

Using the event server will also change the system time format to ROS time. This will be indicated in the reply from command `GetStatus`. This will change the time value in the data coming on the data stream. The time value represents the time that have elapsed since 12:00:00 midnight on January 1, 1970, UTC, where the upper unsigned 32 bits is the no of seconds and the lower unsigned 32 bits is the no of nanoseconds.

Event changes

The event `BreezeRuntimeStarted` will be sent when Breeze Runtime has started and connected to the event server. For a complete list of events, please see the section Command Errors and Events.

Events specific to the event server

Event Code	Constant	Description
2000	BreezeRuntimeStarted	Breeze Runtime has started and connected to Event Server

White reference intensity

Using the event server will default to `useWhiteRefIntensity`

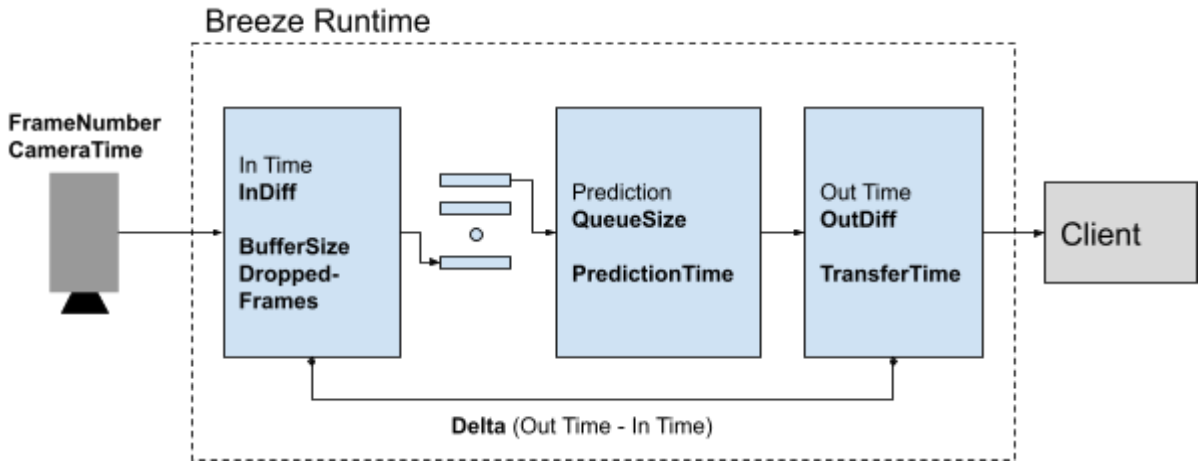
Appendix B: Diagnostics

Command

```
StartDiagnostic(File = Absolute file path)
StopDiagnostic()
```

Data.csv is stored in the file path when StopDiagnostic is called.

The diagnostic information overview



Name	Description	Unit
`Index`	Incrementing number started at 1	Number
`Seconds`	Seconds from diagnostic start	Seconds
`FrameNumber`	Frame number from camera	Number
`CameraTime`	*Not implemented*	
`InDiff`	Time difference between two In frames from camera	Ms
`DroppedFrames`	Number of dropped frames	Number
`BufferSize`	Elements in camera line buffer	Number
`QueueSize`	Elements in prediction frame queue	Number
`PredictionTime`	Prediction time	Ms
`OutDiff`	Time difference between two Out frames	Ms

`Delta`	Time difference between In and Out Time	Ms
`TransferTime`	Transfer time to client	Ms

Logging

The Breeze Runtime log file is located in the {home directory}/.Evince folder and is called BreezeRuntime.log

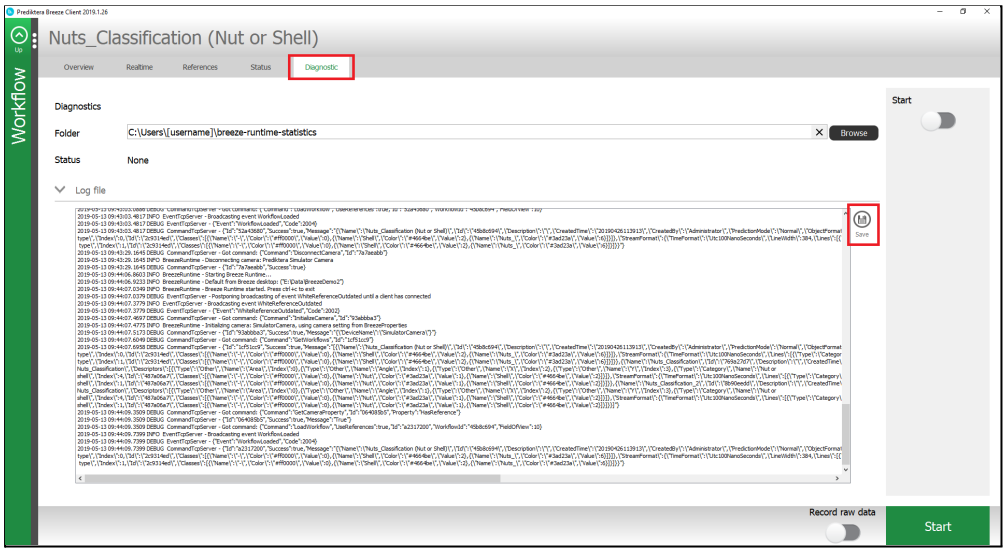
The default logging is in Debug level in log file and Info level for console.

For more logging this level can be changed to for example Trace in the **NLog.config** file. This file is located in the {program files}/Prediktera/Breeze/Runtime folder.

Current settings in NLog.config file:

```
<rules>
  <logger name="*" minlevel="Debug" writeTo="logfile" />
  <logger name="*" minlevel="Info" writeTo="console" />
</rules>
```

The log file logging can also be accessed from Breeze Client under Diagnostics. This log information is automatically updated when changed and can be exported by clicking on the save button.



Appendix C: Code example

The code example can be found in the Breeze installation directory in the Runtime/ExampleCode folder.

For example: *C:\Program Files\Prediktera\Breeze\Runtime\ExampleCode*

Preconditions

- Breeze Tutorial – Classification of nuts – step 1 basic is finished.
- Breeze Tutorial – Quantification of powder samples is finished.
- Breeze runtime workflows has been exported from Breeze as in the Breeze Runtime Tutorial.
- Prediktera Simulator Camera has been configured in Breeze.
- When Breeze Runtime starts, make sure it uses the same workspace as the tutorials.
- To run an example, set the preferred example as startup project in Visual Studio and press play.

Standard procedure for capturing data

1. Initialize camera
2. Load workflow
    a. Get a list of all workflows, then load the one you want or if you already know the id of the workflow you can load it right away.
3. Take dark reference
    a. Close shutter
    b. Take the reference
    c. Open shutter
4. Take white reference
    a. Put the white reference under the camera, then take the white reference.
5. Start predicting
    a. Listen to events and/or data streams.
6. Stop predicting
7. Disconnect the camera

Connect to Breeze Runtime for sending commands

Create a new TcpClient and connect to the host where Breeze Runtime is running on the command port which by default is 2000.
Send json commands to Breeze Runtime as a byte array and wait for a response.

Example commands and replies

Initialize camera:
`"{"Command":"InitializeCamera","Id":"1","DeviceName":"","RequestedPort":3000}"`
Reply: `"{"Id":"1","Success":true}"`

Get workflows: `"{"Command":"GetWorkflows","Id":"2","IncludeTestWorkflows":false}"`

Reply with a message array:

`"{"Id":"2","Success":true,"Message":"[{"Name":"Nuts_Classification","Id":"bd2c9b32","Description":"","CreatedTime":"20190417152441","CreatedBy":"Administrator","PredictionMode":"Normal","ObjectFormat":{"Id":"2d81bdbe","Name":"Sample - Nuts_Classification","Descriptors":[{"Type":"Category","Name":"Nut or Shell","Index":0,"Id":"6a2876d0","Classes":[{"Name":"-","Color":"#ff0000","Value":0},{"Name":"Nut","Color":"#3ad23a","Value":1},{"Name":"Shell","Color":"#4664be","Value":2}]}]},"StreamFormat":{"TimeFormat":"Utc100NanoSeconds","Lines":[{"Type":"Category","Name":"SampleCategory","Index":0,"Id":"db8d7f96","Classes":[{"Name":"-","Color":"#ff0000","Value":0},{"Name":"Sample","Color":"#3ad23a","Value":1}]},{"Type":"Category","Name":"Nut or shell","Index":1,"Id":"6a2876d0","Classes":[{"Name":"-","Color":"#ff0000","Value":0},{"Name":"Nut","Color":"#3ad23a","Value":1},{"Name":"Shell","Color":"#4664be","Value":2}]}]}}]"}"`

Pseudo code

```
public string SendCommandToServer(byte[] data)
{
    using var tcpClient = new TcpClient(hostName, 2000);
    using var stream = tcpClient.GetStream();
    const string lineEnding = "\r\n";

    // Send command
    stream.Write(data, 0, data.Length);
    stream.Write(Encoding.ASCII.GetBytes(lineEnding), 0, lineEnding.Length);

    // Read response
    while (!stream.DataAvailable)
    {
                Thread.Sleep(1);
    }

    string responseStr;
    using var memoryStream = new MemoryStream();

    var response = new byte[tcpClient.ReceiveBufferSize];
    int readByteCount;
    while (stream.DataAvailable && (readByteCount = stream.Read(response, 0,
              response.Length)) > 0)
    {
                memoryStream.Write(response, 0, readByteCount);
    }
    responseStr = Encoding.ASCII.GetString(memoryStream.ToArray(), 0, (int)
              memoryStream.Length).Replace(lineEnding, "");

    return responseStr;
}
```

Listening to server events (optional)

Start a new tread and create a new TcpClient and connect to the host where Breeze Runtime is running on the server event port which by default is 2500.

Pseudo code

```
Task.Run(() =>
            {
                try
                {
                    ListenToServerEvents();
                }
                catch (Exception e)
                {
                    HandleException(e);
                }
            });

public void ListenToServerEvents()
{
        using var tcpClient = new tcpClient(hostName, 2500);
        using var stream = tcpClient.GetStream();
        while (!_stopListeningToServerEvents)
        {
                if (!stream.DataAvailable)
                {
                        Thread.Sleep(100);
                        continue;
                }
                using var streamReader = new StreamReader(stream, Encoding.UTF8);
                var json = streamReader.ReadLine();
                var runtimeEvent = RuntimeEvent.FromJson(json);
                if (runtimeEvent.Event == "PredictionObject")
                {
                        var predictionObjectJson = runtimeEvent.Message;
                        var predictionObject =
                                PredictionObject.FromJson(predictionObjectJson,_workflowSetup.Ob
                                                jectFormat);
                        HandlePredictionObject(predictionObject);
                }
        }
}
```

Listening to data stream (optional)

Start a new tread and create a new TcpClient and connect to the host where Breeze Runtime is running on the stream port which by default is 3000.

Pseudo code

```
Task.Run(() =>
            {
                try
                {
                    ListenToDataStream();
                }
                catch (Exception e)
                {
                    HandleException(e);
                }
            });

public void ListenToDataStream()
{
        using var tcpClient = new tcpClient(hostName, 3000);
        using var stream = tcpClient.GetStream();

        const int streamTypeFieldSize = sizeof(byte);
        const int inTimeFieldSize = sizeof(long);
        const int outTimeFieldSize = sizeof(long);
        const int frameNumberSize = sizeof(long);
        const int dataBodyFieldSize = sizeof(int);

        const int headerSize = streamTypeFieldSize +
               inTimeFieldSize +
               outTimeFieldSize +
               frameNumberSize +
               dataBodyFieldSize;
        const int dataBodySize = BitConverter.ToInt32(header,
               streamTypeFieldSize +
               inTimeFieldSize +
               outTimeFieldSize +
               frameNumberSize);
        while (!stopped)
        {
               byte[] header = ReadHeaderFromStream(stream, headerSize);
               byte[] data = ReadDataFromStream(stream, dataBodySize);
               HandleData(data, GetStreamType(header));
        }
}
```

Appendix D: Breeze properties

BreezeProperties.xml measurement settings with default settings used by Breeze Runtime
Note. BreezePropertiex.xml is located in the Breeze workspace root folder.

```
<measurement>
   <whiteRefValid>0</whiteRefValid>
   <darkRefValid>0</darkRefValid>
   <referenceFrames>25</referenceFrames>
   <useWhiteRefIntensity>true</useWhiteRefIntensity>
   <runtimeQueueSize>1000</runtimeQueueSize>
   <usePipelineRateKeeper>true</usePipelineRateKeeper>
   <referenceQuality>
      <edgePercent>0.1</edgePercent>
      <darkMaxSignal>0.7</darkMaxSignal> // Percent of max signal
      <whiteMinSignal>0.5</whiteMinSignal> // Percent of max signal
      <whiteMaxSignal>0.99</whiteMaxSignal> // Percent of max signal
      <whitePixelVariation>0.05</whitePixelVariation>
      <maxLineVariation>0.05</maxLineVariation>
   </referenceQuality>
</measurement>
```

Appendix E: Generic camera

InitializeCamera command

1. Await "camera.isinitialized" is true (Timeout 5 seconds)
2. Await "camera.temperature.stable" is true (Timeout 5 seconds)
3. Check if "camera.image.istestimage" is true
4. Open Shutter
5. Set pre-processing enabled
6. Await pre-processing is true (Timeout 5 seconds)
7. Set "preprocessing.batch.frames" to 1
8. Set "processing.batch.frames" to 1
9. Set spectral and spatial binning modes
10. Read wavelength table
11. Set "networking.multicast.datagramsize" to 8972
12. Set "networking.performancemode.enable" to true
13. Set "output.stream.frames.udp.resendthreshold" to 1
14. Set "output.stream.frames.udp" to true

Initialize command (Re-Initialize predicting)

1. StopPredict (if predicting)
2. For number of Tries (If not successful wait TimeBetweenTrialSec)
    a. Disconnect camera (if connected)
    b. Call InitializeCamera command
    c. Wait for Camera is stable
3. Check white reference Ok
4. Start prediction

Streaming Error

If no UDP packet is received for 5 seconds -> CameraErrorCode ("Camera not streaming")
Action -> Call Initialize

Appendix F: Detection Technology camera

Failed to obtain energy levels Error

If the energy levels are not satisfied with the timeout time according to the information in the heartbeat received every second:

```
| latestHearBeatPacket.V1 - 24 | < 2.4
| latestHearBeatPacket.V2 - 3.3 | < 0.165
| latestHearBeatPacket.V3 - 2.5 | < 0.125
| latestHearBeatPacket.V4 - 1.1 | < 0.055
```

Streaming Error

If no UDP packet is received for 5 seconds -> CameraErrorCode ("Camera not streaming")
Action -> Call Initialize

Heartbeat Error

If a UDP heartbeat packet is received with `ErrorId ≠ 0x00` -> CameraErrorCode (Heartbeat error message)
Action -> Call Initialize

Breeze Runtime mandatory fields

Name	Min	Max	Step	Unit
GeneratorVoltage	0	160	1	kV
GeneratorCurrent	0	10	0.1	mA