UIDEP Definition Version 2.3

Universal Immssion Data Exchange Protocol

Universal communication system for air monitoring network components like central servers, data loggers, analyzers etc.

Authors: Ing. Erich Kitzmüller, Ambient Informtion Processing GmbH, and Dipl. Ing. Alfred Jakob

10.03.2022

Contents

Revision history

Version	Date	Changes
2.0	2019-07-25	Complete redesign of the specification
2.1	2019-11-21	Recommendations for operating modes, operating statuses etc.
2.2	2020-04-10	Representation of operating modes for devices, where each component has a separate operating mode; operating mode ‘G‘
2.3	2022-03-10	Modified the return port for the answers to the device detection broadcast

Preamble

The current de-facto standard for the communication between analyzers and data loggers is the so-called Bavaria-Hessia protocol, which works with serial cables as well as with LAN connections (TCP, UDP). Coming from the times of serial communication, it has several serious limitations, regarding the number of transferable components and statuses. This is now causing some problems, since network operators need, in addition of the pollutant or meteorlogical parameter the device measures, more internal parameters, like temperature and pressure in some modules, and statuses of the analyzers for quality assurance purposes.

UIDEP provides a modern standard, which has no imminent limints regarding the number of components and statuses, and it based on widely used protocols and formats – HTTP, JSON, REST – to make it easy to implement.

This protocol covers the communication between analyzer and data logger as well as other cases of communication, especially the data transfer between the monitoring network central and data loggers and the transfer of data from the monitoring network central to interested parties.

UIDEP therefore means the following

- a JSON structure to represent measured values and other data, which are transfered within a monitoring network
- transfering data in said JSON structure over HTTP as a network protocol (additionally there is a UDP broadcast for the automatic recognition of devices within the local network)
- said JSON structure as a file format for the transfer of data in files (e.g. from one network to another)

1. Overview on Features

In order to replace the tried-and-true Bavaria-Hessia-Protocol, it is obviously necessary to provide all functions of the Bavaria-Hessia-Protokoll in UIDEP as well. Furthermore, some new features are introduced, especially for the automatic device detection in the local network of a monitoring station.

Features regarding the communication between analyzer and data logger:

- Polling of the currently measured values and statuses
- Switching the operating mode of the analyzer (especially for automatic function checks)
- Changing the communication settings of the analyzer
- Clock synchronisation
- Detection of UIDEP-enabled analyzers in the local network of the monitoring station
- Querying the capabilities of a device

Features regarding the communication between data logger and monitoring network center:

- Polling of currently measured values and aggregated values (e.g. half-hour medium values)
- Polling of the results of the automatic function checks (often called „calibration“)
- Controling the analyzers that are connected to the data logger
- Configuring the analyzers that are connected to the data logger
- Configuring the data logger
- Sending additional files from the data logger to the monitoring network center
- Event notification in case of unusual conditions
- Clock synchronisation

Features regarding the communication between the monitoring network center and arbitrary recipients (e.g. other monitoring networks, internet services etc.)

- Polling of aggregated values

2. Protocol Structure

The data structure spans over several hierachical levels; depending on the source and the destination of the communications, some levels may be unnecessary and are therefore omitted.
This are the levels of the hierachy:

- Organisation (monitoring network operator)
- Station (point of measurements)
- Device/Analyzer
- Sensor/Component/Channel
- Value

Example of JSON-Data[1], which a data logger receives from an analyzer:

```
{
  "Device":"Horiba APNA/370",
  "SN":"12345678",
  "Components": [
    {
      "ID":"178",
      "Value": 6.543
    },
    {
      "ID":"179",
      "Value": 2.345,
      "ErrSts": ["F"]
    }
  ]
}
```

1Spaces and line breaks are optional in JSON; in this document, we include them for improved readability

When the data logger sends this data to the monitoring network center at a later time, it adds some information on a new level of hierachy, and it also amends the exiting levels with new fields:


```
{
  "Station": "AIP-Teststation",
  "Time": "2015-05-19T11:30:00+01:00",
  "AvgTime": 1800,
  "Devices": [
    {
      "Device":"Horiba APNA/370",
      "SN":"12345678",
      "Components": [
        {
          "Component":"NO2",
          "ID":"178",
          "Unit": "ppb",
          "Value": 6.543,
          "MinValue": 0.112,
          "MaxValue": 11.778,
          "StdDev": 1.44,
          "Valid": true
        },
        {
          "Component":"NO",
          "ID":"179",
          "Unit": "ppb",
          "Value": 2.345,
          "MinValue": 0.012,
          "MaxValue": 4.778,
          "StdDev": 0.74,
          "Valid": true
        }
      ]
    },
    {
      "Device":"Horiba APOA/370",
      "SN":"12345679",
      "Components": [
        {
          "Component":"O3",
          "ID":"211",
          "Unit": "ppb",
          "Value": 16.543,
          "MinValue": 3.112,
          "MaxValue": 31.778,
          "StdDev": 2.44,
          "OpSts": ["N"],
          "Valid": true
        }
      ]
    }
  ]
}
```

The following rules are in effect:

I. Each source only provides fields whose correct contents it knows; all other fields are omitted. For example, a device without a clock does not provide a (fictional) timestamp.
II. Fields with empty content may and shall be omitted completely.
III. The order of fields within an object is not relevant, there is no mandatory order and each possible order leads to the same interpretation.
IV. Each field is allocated to exactly one level of hierachy.
V. Fields of a lower hierachical level may be pulled up to a higher level if their value is identical for all objects on the lower level. For example, the time of measurement can be provided on the level of „station“ if its value is the same for all „components“. This avoids unnecessary repetitions.
VI. If there is only one object on a hierachical level (e.g. only on component in an analyzer), it can be combined with the parent object on the level above. (Flattening)

Example for rule I:

Wrong:

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "Components":[
    {
      "ID":"178",
      "Value": 6.543,
      "Time": ""
    }
  ]
}
```

Correct:

```
{
  "Device":"Horiba APNA/370",
  "SN":"12345678",
  "Components":[
    {
      "ID":"178",
      "Value": 6.543
    }
  ]
}
```

Example for rule II:

Representation variant 1 (valid but not recommended):

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "Components":[
    {
      "ID":"178",
      "Value": 6.543,
      "ErrSts":[],
      "OpSts":[]
    }
  ]
}
```

Representation variant 1 – same meaning as representation variant 1:

```
{
  "Device":"Horiba APNA/370",
  "SN":"12345678",
  "Components":[
    {
      "ID":"178",
      "Value": 6.543
    }
  ]
}
```

Example for rule III:

Representation variant 1:

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "Components":[
    {
      "ID":"178",
      "Value": 6.543,
      "ErrSts":["A","B"],
      "OpSts":["M"]
    }
  ]
}
```

Representation variant 2 – same meaning as representation variant 1:

```
{
  "SN":"12345678",
  "Components":[
    {
      "Value":6.543,
      "OpSts":["M"],
      "ErrSts":["B","A"],
      "ID":"178"
    }
  ],
  "Device":"ACME Sensor XYZ"
}
```

Example for rule V:

Representation variant 1:

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "Components":[
    {
      "ID":"178",
      "Value":6.543,
      "Time":"2015-05-19T11:30:00.000+01:00",
      "ErrSts":["A","B"],
      "OpSts":["M"]
    },
    {
      "ID":"179",
      "Value":1.345,
      "Time":"2015-05-19T11:30:00.000+01:00",
      "ErrSts":["A","B"],
      "OpSts":["M"]
    }
  ]
}
```

Representation variant 2 – same meaning as representation variant 1 :

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "ErrSts":["A","B"],
  "Time":"2015-05-19T11:30:00.000+01:00",
  "OpSts":["S"],
  "Components":[
    {
      "ID":"178",
      "Value": 6.543
    },
    {
      "ID":"179",
      "Value": 1.345
    }
  ]
}
```

Example for rule VI:

Representation variant 1:

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "Components": [
    {
      "ID":"178",
      "Value": 6.543,
      "Time": "2015-05-19T11:30:00.000+01:00",
      "ErrSts": ["A","B"],
      "OpSts": ["M"]
    }
  ]
}
```

Representation variant 2 – same meaning as representation variant 1:

```
{
  "Device":"ACME Sensor XYZ",
  "SN":"12345678",
  "ID":"178",
  "Value": 6.543,
  "Time": "2015-05-19T11:30:00.000+01:00",
  "ErrSts": ["A","B"],
  "OpSts": ["M"]
}
```

3. Data Transfer

UIDEP-enabled devices provide services for data transfer, which are designed REST-ful (REST = Representional State Transfer).

This means:

- The services are stateless, no need to save some state across requests
- Services are provided in a hierachcal structure
- http-Requests GET, POST, PUT, DELETE are used to request, create, change, delete data
- Intermediare agents in the network (firewall, gateways etc.) have no effect on the interface

Delimitation: Authentification, encryption and credentials management
Those aspects of the communication are outside of the scope of this specification. When such features are required, we recommend: HTTP-BASIC authentification or providing username and password as HTTP parameters in the URL. For an effective encryption, transfer UIDEP over HTTPS; use client and server certificates to safely authenticate the communication partners. Restrictions regarding the privileges of users and roles have to be arranged between the communications partners, and they are not within the scope of the UIDEP protocol specification.

Hint: the URL paths in the following descriptions, e.g. „/values/simple“, are always relative to the base URL of the device. The base URL can be determined by the UIDEP device detection protocol, as it is returned in the field DevURL.

3.1. Services provided by an analyzer for polling the measured values

Supported http-method: GET

Basic measured values: /values/simple

Measured values with additional information: /values/complex

Without further request parameters, this service returns the values of all components of the analyzer for the most recent measurement.

Optional request parameters:

?component=*id*	request data from this component only
?start=*YYYY-MM-DD-hh-mm-ss*	request historical data since *start* (incl.)
?end=*YYYY-MM-DD-hh-mm-ss*	request historical data till *end* (incl.)
?avgtime=*n*	request for averaged values (averaging interval *n* given in seconds)
?valid=true	only return valid measured values

Example:

```
{
  "Device":"Horiba APNA/370",
  "SN":"12345678",
  "Components": [
    {
      "ID":"178",
      "Value": 6.543
    }
  ]
}
```

Hints:

- /values/simple should return compact data with basic information only
- /values/complex should return all available supplementary information as well

- it is recommended that /values/simple only returns the primary components (pollutants, meteorological values etc.) while /values/complex also returns values of internal sensors (flow, pressure, lamp voltage etc.)
- Operating status and error statuses, as far as they are necessary for the correct interpretation of the measured values, shall be returned by /values/simple
- “ID“ is a unique, not necessarly descriptive identifier for each component. It is permited to proved a description of the component in the “Component“ field (z.B. chemical formula of the pollutant). Many devices that support the Bavaria-Hessia protocol allow an assignment of a three-digit number for each component (or for the first component). This number, represented as a String in JSON, is a possible value for “ID“; any other unique string is allowed as well. It is recommended that users have a way to configure at least the IDs of the primary components returned by /values/simple. The supplimentary components provided by /values/complex can have a preset unique ID, e.g. „Flow4“ for the forth flow sensor.

Recommendation for operating statuses and error statuses:

During normal operation, the measured values should have no operating status and no error status. This means, a status meaning „normal operation of the device“ is not recommended.

Status are be represented by strings of arbitrary length; to avoid possible character set issues, it is recommended that only characters in the ASCII-range 33-126 are used (prefered upper-case letters, digits, lower-case letters).

Recommendations for often used operating statuses:

- "W": The device is in maintenance mode (usually initiated by a switch operated by the maintenance technician)
- "S": The device is in the automatic function check, span gas is inhaled
- "Z": The device is in the automatic function check, zero gas is inhaled
- "R": The device is operating in an extended measuring range. (It is not recommended to provide a status for the first respectively only operating range)

Recommendations for often used error statuses:

- "T": a temperature sensor of the device reports an inacceptible value
- "P": a pressure sensor of the device reports an inacceptible value
- "F": a pressure sensor of the device reports an inacceptible value
- "E": a problem regarding the power supply has been detected
- "L": a problem with the lamp has been detected (for analyzers using this measurement method)

For operating statuses and error statuses, which can appear on several places in the analyzer, it is recommended to add a number to the given letter for uniqueness.

Example:

- "R3": „Range 3“ - the devices operates in the third range (operating status)
- "S2": „Span 2“ - The device is in the automatic function check, the second span gas is inhaled (operating status)
- "T4": „Temperature 4“ - the 4th temperature sensor reports an inacceptable value

3.2. Services provided by the an analyzer for controling the operating mode

Supported http-methods: GET (request operating mode), PUT (change operating mode)
Read/write the operating mode: /mode

Notice: While operatings modes are conceptually allocated to the *components* level, rule V. allows to represent the mode on the *device* level when it affects all components of the device equally. (In prior versions of the UIDEP specification up to V2.1, the operating mode was conceptually allocated to the *device* level)

Example:

```
{
  "OperatingMode": "M"
}
```

Example for a device, where each component has a separate operating mode (e.g. calibrators):

```
{
  "Components": [
     {"ID":"178", "OperatingMode":"Z"},
     {"ID":"179", "OperatingMode":"S"}
  ]
}
```

When using the PUT method, the list of components can contain only some of the components of the device; the operating modes of all components not contained in the list are left unchanged.

Here are the recommendations for often used operating modes. Generally it is recommended that operating modes an the resulting operating statuses use corresponding identifiers, where applicable.

- "M": „Measuring“ - normal measurement operation
- "S": „Span“ - automatic function check, span gas is inhaled
- "Z": „Zero“ - automatic function check, zero gas is inhaled
- "G": „Gas phase titration“ - calibration, calibration gas is produced by GPT

Just like in the description of operating status, it is recommended to append a number to the letter to make the identifiers for related modes unique:

- "S1": span gas for the first test point is inhaled
- "S2": span gas for the second test point is inhaled
- "R1": first measurement range is active
- "R2": second measurement range is active

3.3. Services provided by an analyzer for device configuration

Supported http-methods: GET (read configuration), PUT (change configuration)
Read/write the configuration: /configuration

Example:

```
{
  "DevConfiguration": [
    {
      "SettingName":"foo",
      "DataType":"string",
      "MaxLength":10,
      "CurrentSetting":"bar"
    },
    {
      "SettingName":"xyz",
      "DataType":"integer",
      "MinSetting":0,
      "MaxSetting":100,
      "CurrentSetting":50
    }
  ]
}
```

Recommended type names for “DataType“:

- “string“ - string; allowed length is given in “MaxLength“
- “integer“ - integer number; the allowed range is given in “MinSetting“ and “MaxSetting“
- “float“ - floating point number; the allowed range is given in “MinSetting“ and “MaxSetting“
- “boolean“ - logcal value; possible values are *true* and *false*

Configuration data with a data type not listed above are only maintainable with a vendor specific program.

Devices which are not configurable by UIDEP should return the following response to a GET request (deliberately ignoring rule II on page 7):

```
 {
   "DevConfiguration": []
 }
```

When changing the configuration with a PUT request, only the parameters to be changed shall be supplied; the fields “DataType“, “MaxLength“, “MinSetting“ and “MaxSetting“ can be provided, but the analyzer shall ignore them.

3.4. Services provided by a data logger for polling aggregated values

Supported http-method: GET
Basic measured values: /values/simple
Measured values with additional information: /values/complex
Function check results: /functioncheck

Without further request parameters, this service returns the values of all components of the analyzers for the most recent aggregation period.
Optional request parameters:

?start=*YYYY-MM-DD-hh-mm-ss*	request historical data since *start* (incl.)
?end=*YYYY-MM-DD-hh-mm-ss*	request historical data till *end* (incl.)
?avgtime=*n*	request for averaged values (averaging interval *n* given in seconds)

Example:

```
{
  "Station": "AIP-Teststation",
  "Time": "2015-05-19T11:30:00.000+01:00",
  "AvgTime": 1800,
  "AmsType": "GEMI IOX",
  "Version": "3.02",
  "Devices": [
    {
      "Device":"Horiba APNA/370",
      "SN":"12345678",
      "Components": [
        {
          "Component":"NO2",
          "ID":"178",
          "Unit": "ppb",
          "Value": 6.543,
          "MinValue": 0.112,
          "MaxValue": 11.778,
          "StdDev": 1.44,
          "Valid": true,
         "ChkTime": "2015-05-19T11:30:00.000+01:00",
          "Refs": [
            {
              "RefNo": 0,
              "NominalValue": 0.0,
              "CheckValue": 0.2
            },
            {
              "RefNo": 1,
              "NominalValue": 120.0,
              "CheckValue": 118.85
             },
            {
              "RefNo": 2,
              "NominalValue": 200.0,
              "CheckValue": 201.7
            }
          ]
        }
      ]
    }
  ]
}
```

Comments given in 3.1 are applicable here.

3.5. Services provided by a data logger for configuration

Supported http-methods: GET (read configuration), PUT (change configuration)
Read/write the configuration: /configuration

Example:

```
{
    "Station": "AIP-Teststation",
    "StationConfiguration": [
      {
        "SettingName":"foo",
        "DataType":"string",
        "MaxLength":10,
        "CurrentSetting":"bar"
      },
      {
        "SettingName":"xyz",
        "DataType":"integer",
        "MinValue":0,
        "MaxValue":100,
        "CurrentSetting":50
      }
    ],
    "Devices": [
      {
        "Device": "Horiba APNA/370",
        "SN": "12345678",
        "DevConfiguration": [
           {
            "SettingName":"bar",
            "DataType":"string",
            "MaxLength":10,
            "CurrentSetting":"foo"
          },
          {
            "SettingName":"abc",
            "DataType":"integer",
            "MinValue":0,
            "MaxValue":60,
            "CurrentSetting":20
          }
        ]
      }
    ]
  }
}
```

Comments given in 3.3 are applicable here.

3.6. Services provided by a data logger for the transfer of documents

Supported http-methods: GET (request list, download document), POST (upload document)
List of documents: /documents (only GET allowed)
Download/Upload document: /documents/*filename*

Example for GET /documents:

```
{
  "Documents": [
    "somedocument.pdf",
    "anotherdocument.zip"
  ]
}
```

3.7. Service provided by the monitoring network center for the reception of event notifications

Supported http-method: POST
Send event notification: /eventnotification

Example:

```
{
  "Station": "AIP-Teststation",
  "Time": "2015-05-19T11:37:15.000+01:00",
  "EventType": "Invalid function check",
  "EventText": "Span gas bottle empty",
  "Device":"Horiba APNA/370",
  "SN":"12345678",
  "Components": [
    {
      "ID":"178"
    }
  ]
}
```

Recommended event types (field “EventType”):

- “Operating Status” → Information about changed operating status
- “Limit exceedance” → Information about values exceeding a limit
- “Device change” → Information about a exchanged analyzer
- “Invalid function check” → Notification about a failed function check
- “Restart” → Notification about a system restart (analzer or data logger)
- “System” → Other messages automatically generated by the system
- “Manual” → Notification manually generated by the user
- “Document transfer” → There are new documents waiting to be transferred

3.8. Services provided for requesting the capabilities of a device

Supported http-method: GET
List of capabilities: /capabilities

Example:


```
{
  "DevCapabilities": {
    "Descriptors": [
      "DevURL", "DevCapabilities", "Descriptors", "ChkTime",
      "CheckValue", "Component", "Components", "CurrentSetting",
      "DataType", "DevConfiguration", "Device", "ErrSts",
      "EventText", "EventType", "HistoricalData",
      "ID", "IntSts", "LastCalibration", "MaxSetting", "MeasuredValues",
      "MinSetting", "NominalValue", "NotificationURL", "OperatingModes",
      "OperatingMode", "OpSts", "RefNo", "Refs", "Services",
      "SN", "StoragePeriod", "Time", "Unit", "Valid", "Value",
      "Version"
    ],
    "Services": [
      "values/simple", "values/complex", "configuration",
      "capabilities"
    ],
    "UIDEPVersion":"2.1"
  },
  "DevConfiguration": [
    {
      "SettingName": "SomeSetting",
      "DataType":"integer",
      "MinSetting":0,
      "MaxSetting":100,
      "CurrentSetting":42
    },
    {
      "SettingName": "AnotherSetting",
      "DataType":"string",
      "MaxLength":50,
      "CurrentSetting":"FooBar"
    }
  ],
  "LastCalibration": "2015-05-19T11:37:15.000+01:00",
  "Features": {
    "TimeSynchronized": true,
    "HistoricalData": true,
    "Aggregations": [
      {
        "AvgTime": 5,
        "StoragePeriod": 86400
      },
      {
        "AvgTime": 300,
        "StoragePeriod": 864000
      },
      {
        "AvgTime": 1800,
        "StoragePeriod": 86400000
      }
    ]
  },
  "OperatingModes": [
    {
      "OperatingMode": "M",
```

```
        "Description": "Measuring"
      },
      {
        "OperatingMode": "Z",
        "Description": "Zero Function Check"
      },
      {
        "OperatingMode": "S",
        "Description": "Span Function Check"
      },
      {
        "OperatingMode": "P",
        "Description": "Power saving mode (Standby)"
      }
    ],
    "OperatingStatuses": [
      {
        "Status": "Z",
        "Description": "Zero function check active"
      },
      {
        "Status": "S",
        "Description": "Span function check active"
      },
      {
        "Status": "U",
        "Description": "Device in startup procedure"
      },
      {
        "Status": "D",
        "Description": "Device in shutdown procedure"
      }
    ]
    "ErrorStatuses": [
      {
        "Status": "T",
        "Description": "Operating temperature not in acceptable range"
      },
      {
        "Status": "N",
        "Description": "No response from sensor unit"
      }
    ]
  }
```

Specifying "Descriptors" is optional (as it is only here for compability with prior version of UIDEP); it is recommended to omit this field. Recipients of UIDEP data must accept the presence of "Descriptors" but are not required to check its contents. If "Descriptors" is provided, it shall contain all UIDEP descriptors this device supports.

3.9. Service for clock synchronisation

Supported http-method: GET
Current timestamp: /currenttime

Example:

```
{
  "CurrentTime": "2015-11-18T10:22:05.000+01:00",
  "TimeSynchronized": true
}
```

This service allows comparing the system‘s own clock with the clock of the communication partner, to check if the time difference is within acceptable limits.

If there is a time difference, the system can choose to

- ignore the time difference
- compensate for the difference in the timestamp of measured values
- discard the values

This concerns only the requests for historical data. When requesting current measured values, the system can simply use its own clock as time reference and ignore the communication partner‘s timestamp.

4. List der Descriptors

Descriptor	Type	Description	Level
Active	Boolean	Active-Flag	Device
AmsType	String	Type of automated measuring system (data logger)	Station
AvgTime	Integer	Averaging interval in seconds	Value
ChkTime	String (ISO-Timestamp)	Timestamp of a function check	Function check
CheckValue	Float	Value of a function check**	FC-Ref
Component	String	Component (Parameter) - Description	Comp.
Components	Array	List of Components	Device
Configuration	Object	Configuration data (device specific)	Org., Station, Device, Comp.
CurrentSetting	variable	Current value of a configuration setting	Config.
CurrentTime	String (ISO-Timestamp)	Current time of the device‘s internal clock	Device
DataType	String	Data type of a configuration setting	Config.
Description	String	Description of a state of mode	Mode, State
Descriptors	Array	List of the supported descriptors	Device
Device	String	Device type	Device
Devices	Array	List of devices per station	Station
DevURL	String	Base URL for UIDEP related services of the device	Device
Documents	Array<String>	List of available documents	Station
ErrorStatuses	Array	List of possible error status	Device
ErrSts	Array<String>	List of applicable error status	Value
EventText	String	Descriptive text of an event notification	EN
EventType	String	Type of an event notification	EN
Features	Object	List of supported features	Device
FunctionChecks	Array	List of function checks	Comp.
ID	String	Component-ID (unique, not necessarily descriptive)	Comp.
IntSts	Array<String>	List of applicable internal statuses	Value
LastCalibration	String (ISO-Timestamp)	Date of the most recent “real“ calibration	Device
MaxLength	Integer	Maximum length of the configuration setting of type String	Conf.
MaxSetting	Float	Maximum allowed value of the configuration setting	Conf.
MaxValue	variable	Largest measured value within the averaging period	Value
MeasuredValues	Array	List of measured values	Comp.
MinSetting	Gleitkomma	Minimum allowed value of the configuration setting	Conf.

`MinValue`	variabel	Least measured value within the averaging period	Value
`NominalValue`	Float	Nominal value of the function check**	FC-Ref
`NotificationURL`	String	URL of the network center (for event notifications)	Org.
`OperatingMode`	String	Operating Mode („M“=Measurement, „Z“=Zero, „S“=Span, …)	Comp.
`OperatingModes`	Array	List of possible operating modes	Device
`OperatingStatuses`	Array	List of possible operating statuses	Device
`OpSts`	Array<String>	List of applicable operating statuses	Value
`Organisation`	String	Name of the organisation	Org.
`RefNo`	Integer	Number of the test point (0=Zero, 1=Span, …)**	FC-Ref
`Refs`	Array	List of the test points of an automatic function check	FC
`Services`	Array	List of supported services	Device, Station
`SN`	String	Serial number	Device
`Station`	String	Station name (unique identifier)	Station
`StationURL`	String	Base URL for UIDEP related services of the data logger	Station
`Status`	String	Status code for self description	Status
`StdDev`	Float	Standard deviation within the averaging period	Value
`StoragePeriod`	Integer	Storage duration of aggregated values (in seconds)	Aggr.
`Time`	String (ISO-time stamp)	Timestamp of measurement / end of aggregation period	Value
`TimeSynchronized`	Boolean	True if the device‘s clock is synchronized with an official time source (NTP or similar)	Device
`UIDEPVersion`	String	Supported UIDEP version	Device
`Unit`	String	Unit	Comp.
`Valid`	Boolean	Validity flag	Value
`Value`	Float	Measured value / Averaged value	Value
`Version`	String	Software version	Station

** … If there is more than one test point, RefNo refers to the applicable test point

5. Device detection protocol

The device detection protocol allows automatic detection and integration of analyzers in the local LAN of a station.

The main purpose is to allow the data logger to automatically detect and configure new analyzers in the network. It‘s also conceivable that one analyzer automatically detects another analyzer and uses it to compensate the measured values for cross-sensitivity.

Delimitation: Detection of data loggers from the network center, IPv6

For the detection protocol to work, both the sender and the reciever of the UDP broadcast must reside in the same network segment. This is usually the case for a station-internal network, but not for data loggers and the monitoring network center. By switching from IPv4 broadcasts to IPv6 multicasts, this limit could be overcome in the future. For that reason, this description includes the response of a data logger to such a message.

Device detection is implemented by a UDP broadcast on port 4120 having this content (ASCII):

```
UIDEP-DEVICE-DISCOVERY
```

UIDEP-enabled devices, which support the detection protocol, respond with a UDP message to the sender[2]; this response contains the basic information about the device in JSON format (UTF-8 encoded).

These desciptors can be used:

```
Device, SN, DevURL, Version, Components, ID, Component
```

A data logger responding to such a request can also use these descriptors:

```
Station, StationURL, AmsType, Version, DeviceList
```

Example for an analyzer‘s response:

```
{
  "Device":"Horiba APNA/370",
  "SN":"12345678",
  "DevURL":"http://192.168.11.15:8080/uidep",
  "Components": [
    {
      "ID":"178",
      "Component": "NO2"
    },
    {
      "ID":"179",
      "Component": "NO"
    }
  ]
}
```

2 Up to version V2.2 of the UIDEP specification, the responses to the broadcast were to be sent to port 4120 of the sender, but the majority of implementers let their devices return the response to the port from which the broadcase was sent, which is also easier to implement on the side of the data logger. Therefore, from V2.3 of the UIDEP specification, the response is to be sent to the port from which the broadcast was sent.

Example for a data logger‘s response:

```
{
  "Station": "AIP-Teststation",
  "AmsType": "Horiba IOX",
  "Version": "3.02",
  "StationURL":"http://192.168.11.2/foobar/uidep",
  "Devices": [
    {
      "Device":"Horiba APNA/370",
      "SN":"12345678",
      "DevURL":"http://192.168.11.15:8080/uidep",
      "Components": [
        {
          "ID":"178",
          "Component": "NO2"
        },
        {
          "ID":"179",
          "Component": "NO"
        }
      ]
    }
  ]
}
```

The response must be valid JSON (UTF-8 encoded); the examples above include optional spaces and line breaks for readability.

Please note: while Port 4120 is fixed for the detection protocol, the UIDEP web services can use any port – it‘s just necessary to specify the port in the DevURL field of the answer if it is not the HTTP default port 80. In the example above, port 8080 is specified for the web services.

For technical reasons, a maximum packet size for UDP response packets must be specified. There, the maximum size of UDP response packets for the UIDEP device detection protocol is limited to 1000 bytes.

To match this limit, a device can – at it‘s own option – choose to either:

- include all components of the device
- include only those components returned in /values/simple
- omit all components (for example, a particle classifier can have more than 100 primary components which would alone exceed the limit of 1000 bytes)

It‘s recommended to choose the variant that provides a maximum of components while staying safely within the limit.