AMETEK®
SENSORS, TEST & CALIBRATION

JOFRA RTCt Communication Protocol Manual

AMETEK
SENSORS, TEST & CALIBRATION

JOFRA RTCt Communication Protocol Manual

Table of contents

Introduction

This document describes the JSON based protocol used for communication between a PC and a JOFRA RTCt calibrator.

Consult https://www.json.org/json-en.html for details on JSON syntax in general.

Communication paradigm

The JSON based protocol is supported through both HTTP and the serial USB physical layers.

Master/slave

The JSON protocol is based on the Master/Slave principle. The PC is always to be regarded as the master and the calibrator as the slave. This means that the calibrator never starts the communication by itself, but solely replies to requests from the PC.

Read/write

When talking about writing and reading, the communication is seen from the PC's point of view, which means that writing is information from the PC to the calibrator, whereas reading is information from the calibrator to the PC.

Format

The communication is based on line based JSON string telegrams as specified in this document. The format of a telegram is dependent of it being a request (from the PC) or a reply (from the calibrator).

Request

Three request methods are available:

- **CALL**: Execute a command with no further request and response parameters.
- **GET**: Request information from the calibrator. Optionally includes more request parameters.
- **SET**: Set/update a specified parameter value.

The request strings are JSON strings, containing a leading curly brace ({), the request method (CALL/GET/SET) followed by a colon (:) and the command name. If any parameters are to be sent, they are listed using similar key-value pair syntax, each pair delimited by a comma (,). The telegram ends with a curly brace (}). The order of the parameters is not important.

Parameter names and text strings are always surrounded with double quotes (").

General request syntax:

```
{ <METHOD>: <COMMAND>, <PARAM1>: <PARAM1_VALUE> }
```

Example:

```
{ "GET": "SibPort", "SibPort": 1 }
```

Response

The reply has a leading curly brace ({) followed by the response content and an ending curly brace (}):

Replies from CALL commands include the parameter "CallResponse", a colon (:), and the command name, indicating successful execution.

Reply content from GET commands include the parameter "GetResponse", a colon (:), and the command name, followed by a comma and the (first) parameter name, a colon (:), and the returned value, for each returned parameter. Parameter key-value pairs are comma separated.

SET commands include the parameter "SetResponse", a colon (:), and the command name, indicating successful execution.

Errors return the text "Error", a colon (:), and the error description.

CALL: { "CallResponse": <COMMAND> }

SET: { "SetResponse": <COMMAND>, <PARAM>: <VALUE> }

GET: { "GetResponse": <COMMAND>, <PARAM>: <VALUE> }

ERROR: { "ERROR": <ERROR_DESCRIPTION> }

Example: To get the LogOn status, the master issues following command:

JSON: {"GET": "IsLoggedOn"}

The reply is:

{"GetResponse": "IsLoggedOn", "IsLoggedOn": true} in case of success

Notes:

No trailing question mark (?) for GET requests.

White spaces between delimiters and other control symbols are optional.

Strings are always surrounded with double quotes (").

Delimiter between key-value pairs is a comma.

The protocol commands and parameters are case sensitive.

Telegram packages are incased in { } symbols indicating the first and last character.

Example of error responses:

```
{"Error": "Invalid command or argument(s)"}
{"Error": "Telegram not allowed"}
{"Error": "Temperature out of range"}
```

Parameter value placeholders are written as <PARAMETER>, i.e. a leading 'less than' character (<), the parameter identifier in all-caps, and a trailing 'greater than' character (>).

If this parameter is a string, replace the whole parameter, including leading and trailing brackets, with the text string surrounded with double quotes:

String Example: `{ "ID": <ID_PARAMETER> }` becomes: `{ "ID": "MyID" }` for `<ID_PARAMETER> = "MyID"` .

Bool Example: `{ "State": <STATE_PARAMETER> }` becomes: `{ "State": true }` for `<STATE_PARAMETER> = true` .

Number Example: `{ "Counter": <COUNT_PARAMETER> }` becomes: `{ "Counter": 1234 }` for `<COUNT_PARAMETER> = 1234` .

Datatypes

Datatype	Source data width	Note
Number (integer)	16-bit signed	Some parameters are restricted to a small set of integer values, and could optionally be parsed into a fewer bit wide integer datatype.
Number (float)	64-bit double precision	Many floating point parameters only utilize single precision.
String	ASCII characters	Most temperatures and other displayed values are represented in formatted form – with fixed resolution.
Bool		

Protocol content variations

The protocol is implemented on all new models in the RTCt series, but the individual feature set of each model results in variations of the available commands and parameters. The following sections describe these modifying factors and the outcomes on the protocol.

Series

The RTCts display up to three (**0/1/2/3**) decimals.

The DLC (XDIFF) input is only available on RTCt series (B/C models only).

Input configuration

The RTCt series offer models with several input configurations, designated respectively A (No inputs), B (Reference and Sensor inputs) and C (Reference input only).

Commands and parameters related to sensor inputs (SIB1&2; SUT) are only relevant for B models.
Commands and parameters related to reference inputs (RIB; XREF/XDIFF) are only relevant for B and C models.

If a command related to one of the previously mentioned categories is issued to an instrument without the corresponding feature, the response should be an error stating 'Invalid command or argument(s)'.

In command requests and responses, some parameters are only relevant for specific models. The calibrator responds only with relevant parameters. Irrelevant/empty parameters are omitted.

Example: The "LiveSensors" command will return information on both READ, TRUE, SENSOR1, SENSOR2 and XDIFF on an RTCt -C model, but only information on READ on an A model. See details below.

Read all sensors: `{"GET": "LiveSensors"}`

Response on success:

A model	B model	C model
{"GetResponse": "LiveSensors", "READ": { … }, "NumberOfSetDecimals": <SET_DECIMALS>, "Unit": <TEMPERATURE_UNIT> }	{"GetResponse": "LiveSensors", "READ": { … }, "TRUE": { … }, "XDIFF": { … }, "NumberOfSetDecimals": <SET_DECIMALS>, "Unit": <TEMPERATURE_UNIT> }	{"GetResponse": "LiveSensors", "READ": { … }, "TRUE": { … }, "SENSOR1": { … }, "SENSOR2": { … }, "XDIFF": { … }, "NumberOfSetDecimals": <SET_DECIMALS>, "Unit": <TEMPERATURE_UNIT> }

The DLC (XDIFF) input is only available on RTCt series (B/C models only).

Temperature conversion

Temperature values are expressed in one of the following temperature units, via the parameter "Unit" which should always be present in a request or response JSON telegram which also includes one or more temperature parameters. All temperatures within a telegram must be stated in same unit.
Kelvin [K] = "KEL"
Celsius [°C] = "CEL"
Fahrenheit [°F] = "FAR"

Submitting a temperature with no accompanying temperature unit parameter results in an error response.

Telegram commands

Logon

In order to communicate with the calibrator, the master must issue the logon command. It is possible to read parameters when logged on.

If any command is issued without being logged on, a "Telegram not allowed" error response is received.

Call: `{"CALL": "LogOn"}`

Response: `{"CallResponse": "LogOn" }`

Logoff

The log off command disables the possibility to communicate with the calibrator. At logoff, Mode should be set to Local for the user interface of the calibrator is reenabled.

Call: `{"CALL": "LogOff"}`

Response: `{"CallResponse": "LogOff" }`

IsLoggedOn

This command returns the current logon state.

Read: {"GET": "IsLoggedOn"}

Response: {"GetResponse": "IsLoggedOn", "IsLoggedOn": <LOGGEDON> }

Table1: IsLoggedOn read response parameters

Parameter	Type: value(s)
<LOGGEDON>	Bool: true|false

Mode

Read: {"GET": "Mode"}

Response: {"GetResponse": "Mode", "Mode": <REMOTE_MODE>}

Table 2: Mode read response parameters

Parameter	Type: value(s)
<REMOTE_MODE>	String : "Local" | "Remote" | "Locked"

Write: {"SET": "Mode", "Mode": <REMOTE_MODE>}

Table 1: Mode write parameters

Parameter	Type: value(s)
<REMOTE_MODE>	String : "Local" | "Remote" | "Locked"

Response: {"SetResponse": "Mode"}

Examples:

{"GET": "Mode"}

{"SET": "Mode", "Mode": "Local"}

LiveSensors

This command returns the status and readings of the calibrator. All temperature values are in the unit expressed in the calibrator.

LiveSensors can retrieve either the complete array of sensor data or singularly.

Read all sensors: `{"GET": "LiveSensors"}`

Response on success:

```
{"GetResponse": "LiveSensors",
"READ": {
   "Name": <READ_NAME>,
   "ConvertToTemperature": <READ_CONVTOTEMP>,
   "Input": {
      "InputType": <READ_INPUTTYPE>,
      "InputValue": {"Value": <READ_INPUTVALUE>, "Unit": <READ_INPUTUNIT>},
      "TemperatureValue": {"Value": <READ_TEMPVALUE>, "Unit": <READ_TEMPUNIT>}
   },
   "NumberOfDecimals": <READ_DECIMALS>
},
"TRUE": {
   "Name": <TRUE_NAME>,
   "ConvertToTemperature": <TRUE_CONVTOTEMP>,
   "Input": {
      "InputType": <TRUE_INPUTTYPE>,
      "InputValue": {"Value": <TRUE_INPUTVALUE>, "Unit": <TRUE_INPUTUNIT>},
      "TemperatureValue": {"Value": <TRUE_TEMPVALUE>, "Unit": <TRUE_TEMPUNIT>}
   },
   "Stability": {
      "Tolerance": {"Value": <TRUE_STAB_TOLVALUE>, "Unit": <TRUE_STAB_TOLUNIT>},
      "RequiredSeconds": <TRUE_STAB_TIME>,
      "Seconds": <TRUE_STAB_ELAPSED>
   },
   "NumberOfDecimals": <TRUE_DECIMALS>,
   "SetFollows": <TRUE_SETFOLLOWS>
},
"SENSOR1": {
   "Name": <SENSOR1_NAME>,
   "ConvertToTemperature": <SENSOR1_CONVTOTEMP>,
   "Input": {
      "InputType": <SENSOR1_INPUTTYPE>,
      "InputValue": {"Value": <SENSOR1_INPUTVALUE>, "Unit": <SENSOR1_INPUTUNIT>},
      "TemperatureValue": {"Value": <SENSOR1_TEMPVALUE>, "Unit": <SENSOR1_TEMPUNIT>}
   },"CJOhms": {
      "InputType": <CJOHM_INPUTTYPE>,
      "InputValue": {"Value": <CJOHM_INPUTVALUE>, "Unit": <CJOHM_INPUTUNIT>},
      "TemperatureValue": {"Value": <CJOHM_TEMPVALUE>, "Unit": <CJOHM_TEMPUNIT>}
   },
   "Stability": {
      "Tolerance": {"Value": <SENSOR1_STAB_TOLVALUE>, "Unit":
<SENSOR1_STAB_TOLUNIT>},
      "RequiredSeconds": <SENSOR1_STAB_TIME>,
      "Seconds": <SENSOR1_STAB_ELAPSED>
   }
   "NumberOfDecimals": <SENSOR1_DECIMALS>
},
"SENSOR2": {
   "Name": <SENSOR2_NAME>,
   "ConvertToTemperature": <SENSOR2_CONVTOTEMP>,
   "Input": {
      "InputType": <SENSOR2_INPUTTYPE>,
      "InputValue": {"Value": <SENSOR2_INPUTVALUE>, "Unit": <SENSOR2_INPUTUNIT>},
      "TemperatureValue": {"Value": <SENSOR2_TEMPVALUE>, "Unit": <SENSOR2_TEMPUNIT>}
```

```
   },"CJOhms": {
      "InputType": <CJC_INPUTTYPE>,
      "InputValue": {"Value": <CJC_INPUTVALUE>, "Unit": <CJC_INPUTUNIT>},
      "TemperatureValue": {"Value": <CJC_TEMPVALUE>, "Unit": <CJC_TEMPUNIT>}
   },

   "Stability": {
      "Tolerance": {"Value": <SENSOR2_STAB_TOLVALUE>, "Unit":
<SENSOR2_STAB_TOLUNIT>},
      "RequiredSeconds": <SENSOR2_STAB_TIME>,
      "Seconds": <SENSOR2_STAB_ELAPSED>
   },
   "NumberOfDecimals": <SENSOR2_DECIMALS>
},
"XDIFF": {
   "Name": <XDIFF_NAME>,
   "ConvertToTemperature": <XDIFF_CONVTOTEMP>,
   "Input": {
      "InputType": <XDIFF_INPUTTYPE>,
      "InputValue": {"Value": <XDIFF_INPUTVALUE>, "Unit": <XDIFF_INPUTUNIT>},
      "TemperatureValue": {"Value": <XDIFF_TEMPVALUE>, "Unit": <XDIFF_TEMPUNIT>}
   },
   "Stability": {
      "Tolerance": {"Value": <XDIFF_STAB_TOLVALUE>, "Unit": <XDIFF_STAB_TOLUNIT>},
      "RequiredSeconds": <XDIFF_STAB_TIME>,
      "Seconds": <XDIFF_STAB_ELAPSED>
   },
   "NumberOfDecimals": <XDIFF_DECIMALS>,
   "SetFollows": <XDIFF_SETFOLLOWS>
},
"NumberOfSetDecimals": <SET_DECIMALS>
}
Note! ”XREF” & “XDIFF” -> B/C models only
      ”SENSOR1” & “SENSOR2” -> B models only
```

Table 2: Livesensors read all response parameters

<table>
<tr><th>Parameter</th><th>Type: value(s)</th></tr>
<tr><td><READ_NAME>
<TRUE_NAME>
<SENSOR1_NAME>
<SENSOR2_NAME>
<XDIFF_NAME></td><td>String eg.INT_RTD
Serial no. if extern

Selected sensor type eg.P100(90)385
Serial no.</td></tr>
<tr><td><READ_CONVTOTEMP>
<TRUE_CONVTOTEMP>
<SENSOR1_CONVTOTEMP>
<SENSOR2_CONVTOTEMP>
<XDIFF_CONVTOTEMP></td><td>Bool: true|false</td></tr>
<tr><td>
<READ_INPUTTYPE>
<TRUE_INPUTTYPE>
<XDIFF_INPUTTYPE></td><td>String:
“INT_RTD”,
“REF_RTD” | “INT_RTD” (depending on True selection.)
“REF_TC”</td></tr>
<tr><td><SENSOR1_INPUTTYPE>
<SENSOR2_INPUTTYPE>

<CJC_INPUTTYPE></td><td>String: "SENS_None" | "SENS_Voltage" | "SENS_Current" |
"SENS_Ohm400" | “SENS_Ohm4000” | "SENS_TC" |
"SENS_Switch"
”CJC”</td></tr>
<tr><td><READ_INPUTVALUE>
<TRUE_INPUTVALUE>
<XDIFF_INPUTVALUE></td><td>READ = Emtpy String</td></tr>
<tr><td><SENSOR1_INPUTVALUE>
<SENSOR2_INPUTVALUE></td><td>String
If InputType is "Switch", InputValue is: "Open" | "Closed"</td></tr>
<tr><td><READ_INPUTUNIT>
<TRUE_INPUTUNIT>
<XDIFF_INPUTUNIT>
<SENSOR1_INPUTUNIT></td><td>String: "mA" | "volt" | "mV" | "Ohm"</td></tr>
</table>

<SENSOR2_INPUTUNIT>	
<READ_TEMPVALUE> <TRUE_TEMPVALUE> <SENSOR1_TEMPVALUE> <SENSOR2_TEMPVALUE> <XDIFF_TEMPVALUE>	String Formatted using selected temperature resolution
<TRUE_STAB_TOLVALUE> <SENSOR1_STAB_TOLVALUE> <SENSOR2_STAB_TOLVALUE> <XDIFF_STAB_TOLVALUE>	String Formatted using selected temperature resolution
<TRUE_STAB_TIME> <SENSOR1_STAB_TIME> <SENSOR2_STAB_TIME>	Number (integer)
<TRUE_STAB_ELAPSED> <SENSOR1_STAB_ELAPSED> <SENSOR2_STAB_ELAPSED> <XDIFF_STAB_ELAPSED>	Number (integer) Time to stable (<0) or time it has been stable (>0).
<XDIFF_STAB_ELAPSED>	Number (integer) Time it has been stable (>0).
<READ_DECIMALS> <TRUE_DECIMALS> <SENSOR1_DECIMALS> <SENSOR2_DECIMALS> <XDIFF_DECIMALS> <SET_DECIMALS>	Number (integer): 0 | 1 | 2 | 3
<TRUE_SETFOLLOWS> <XDIFF_SETFOLLOWS>	Bool: true|false
<TRUE_STAB_TOLUNIT> <SENSOR1_STAB_TOLUNIT> <SENSOR2_STAB_TOLUNIT> <XDIFF_STAB_TOLUNIT>	String: "KEL" | "CEL" | "FAR"

Read single sensor:

```
{"GET": "LiveSensors", "Sensor": <SELECTED_SENSOR>}
```

Table 3: Livesensors read singular request parameters

Parameter	Type: value(s)
<SELECTED_SENSOR>	String: "READ" | "TRUE" | "SENSOR1" | "SENSOR2" | "XDIFF" *Value is not case sensitive*

Response on success:

READ `{"GET": "LiveSensors", "Sensor": “READ”}`

```
{"GetResponse": "LiveSensors",
"READ": {
  "Name": <READ_NAME>,
  "ConvertToTemperature": <READ_CONVTOTEMP>,
  "Input": {
    "InputType": <READ_INPUTTYPE>,
    "InputValue": {"Value": <READ_INPUTVALUE>, "Unit": <READ_INPUTUNIT>},
    "TemperatureValue": {"Value": <READ_TEMPVALUE>, "Unit": <READ_TEMPUNIT>}
  },
  "NumberOfDecimals": <READ_DECIMALS>
},
"NumberOfSetDecimals": <SET_DECIMALS>
}
```

SetTemperature

Sets/reads the SET-temperature.

Read: `{"GET": "SetTemperature"}`

Response:
`{"GetResponse": "SetTemperature", "SetTemperature": {"Value": <SETTEMP_MAXVALUE>, "Unit": <SETTEMP_MAXUNIT>}}`

Table 4: SetTemperature read response parameters

Parameter	Type: value(s)
`<SET_TEMPVALUE>`	`String` Formatted using selected temperature resolution
`<SET_TEMPUNIT>`	`String: "KEL" | "CEL" | "FAR"`

Write:
`{"SET": "SetTemperature", "SetTemperature": {"Value": <SET_TEMPVALUE>, "Unit": <SET_TEMPUNIT>}}`

Table 5: SetTemperature write parameters

Parameter	Type: value(s)
`<SET_TEMPVALUE>`	`String`
`<SET_TEMPUNIT>`	`String: "KEL" | "CEL" | "FAR"`

Response:
`{"SetResponse": "SetTemperature"}`

Example: `{"SET": "SetTemperature", "SetTemperature": {"Value": "300.0", "Unit": "CEL"}}`

Unit

This command sets/reads the temperature unit on the calibrator.

Read: `{"GET": "Unit"}`

Response:
`{"GetResponse": "Unit", "Unit": <TEMPERATURE_UNIT>}`

Table 6: Unit read response parameters

Parameter	Type: value(s)
`<TEMPERATURE_UNIT>`	`String: "KEL" | "CEL" | "FAR"`

Write:
`{"SET": "Unit", "Unit": <TEMPERATURE_UNIT>}`

Table 7: Unit write parameters

Parameter	Type: value(s)
`<TEMPERATURE_UNIT>`	`String: "KEL" | "CEL" | "FAR"`

Response:
`{"SetResponse": "Unit"}`

Example: `{"SET": "Unit", "Unit": "FAR"}`

StabilitySetup

This command sets/reads the stability criteria settings.

Read: `{"GET": "StabilitySetup"}`

Response:

```
{"GetResponse": "StabilitySetup",
"IREF": {
   "ExtendedTime":<IREF_STAB_EXTTIME>
},
"XREF": {
   "Tolerance":{"Value": <XREF_STAB_TOLVALUE>, "Unit": <XREF_STAB_TOLUNIT>},
   "RequiredSeconds":<XREF_STAB_TIME>
},
"XDIFF": {
   "Tolerance":{"Value": <XDIFF_STAB_TOLVALUE>, "Unit": <XDIFF_STAB_TOLUNIT>},
   "Enabled":<XDIFF_STAB_ENABLED>
},
"SENSOR1": {
   "Tolerance":{"Value": <SENSOR1_STAB_TOLVALUE>, "Unit": <SENSOR1_STAB_TOLUNIT>},
   "RequiredSeconds":<SENSOR1_STAB_TIME>,
   "Enabled":<SENSOR1_STAB_ENABLED>
},
"SENSOR2": {
   "Tolerance": {"Value": <SENSOR2_STAB_TOLVALUE>, "Unit": <SENSOR2_STAB_TOLUNIT>},
   "RequiredSeconds":<SENSOR2_STAB_TIME>,
   "Enabled":<SENSOR2_STAB_ENABLED>
},
}
Note! "XREF" & "XDIFF" -> B/C models only
      "SENSOR1" & "SENSOR2" -> B models only
```

Table 8: StabilitySetup read response parameters

Parameter	Type: value(s)
<IREF_STAB_TOLVALUE> <XREF_STAB_TOLVALUE> <XDIFF_STAB_TOLVALUE> <SENSOR1_STAB_TOLVALUE> <SENSOR2_STAB_TOLVALUE>	`String` Formatted using selected temperature resolution
<IREF_STAB_TIME> <XREF_STAB_TIME> <SENSOR1_STAB_TIME> <SENSOR2_STAB_TIME>	`Number (integer)`
<IREF_STAB_EXTTIME>	`Number (integer)`
<SENSOR1_STAB_ENABLED> <SENSOR2_STAB_ENABLED> <XDIFF_STAB_ENABLED>	`Bool: true|false`
<TRUE_STAB_TOLUNIT> <SENSOR1_STAB_TOLUNIT> <SENSOR2_STAB_TOLUNIT> <XDIFF_STAB_TOLUNIT>	`String: "KEL"	"CEL"	"FAR"`

Write:

```
{"SET": "StabilitySetup",
"IREF": {
   "Tolerance":{"Value": <SENSOR2_STAB_TOLVALUE>, "Unit": <IREF_STAB_TOLUNIT>},
   "RequiredSeconds":<IREF_STAB_TIME>,
   "ExtendedTime":<IREF_STAB_EXTTIME>
},
"XREF": {
   "Tolerance":{"Value": <XREF_STAB_TOLVALUE>, "Unit": <XREF_STAB_TOLUNIT>},
   "RequiredSeconds":<XREF_STAB_TIME>
},
"XDIFF": {
   "Tolerance":{"Value": <XDIFF_STAB_TOLVALUE>, "Unit": <XDIFF_STAB_TOLUNIT>},
   "Enabled":<XDIFF_STAB_ENABLED>
},
"SENSOR1": {
   "Tolerance": {"Value": <SENSOR1_STAB_TOLVALUE>, "Unit": <SENSOR1_STAB_TOLUNIT>},
   "RequiredSeconds":<SENSOR1_STAB_TIME>,
   "Enabled":<SENSOR1_STAB_ENABLED>
},
"SENSOR2": {
   "Tolerance":{"Value": <SENSOR2_STAB_TOLVALUE>, "Unit": <SENSOR2_STAB_TOLUNIT>},
   "RequiredSeconds":<SENSOR2_STAB_TIME>,
   "Enabled":<SENSOR2_STAB_ENABLED>
}
}
```

Table 9: StabilitySetup write parameters

Parameter	Type: value(s)
<IREF_STAB_TOLVALUE> <XREF_STAB_TOLVALUE> <XDIFF_STAB_TOLVALUE> <SENSOR1_STAB_TOLVALUE> <SENSOR2_STAB_TOLVALUE>	String Formatted using selected temperature resolution
<IREF_STAB_TIME> <XREF_STAB_TIME> <SENSOR1_STAB_TIME> <SENSOR2_STAB_TIME>	Number (integer)
<IREF_STAB_EXTTIME>	Number (integer)
<SENSOR1_STAB_ENABLED> <SENSOR2_STAB_ENABLED> <XDIFF_STAB_ENABLED>	Bool: true|false
<TRUE_STAB_TOLUNIT> <SENSOR1_STAB_TOLUNIT> <SENSOR2_STAB_TOLUNIT> <XDIFF_STAB_TOLUNIT>	String: "KEL" | "CEL" | "FAR"

Response:
```
{"SetResponse": "StabilitySetup"}
```

UserMinMaxSetTemperature

This command sets/reads the user defined temperature limits of the calibrator.

Read: `{"GET": "UserMinMaxSetTemperature"}`

Response:
```
{"GetResponse": "UserMinMaxSetTemperature", "MinSetTemperature": {"Value":
<SETTEMP_MINVALUE>, "Unit": <SETTEMP_MINUNIT>}, "MaxSetTemperature": {"Value":
<SETTEMP_MAXVALUE>, "Unit": <SETTEMP_MAXUNIT>}}
```

Table 10: UserMinMaxSetTemperature read response parameters

Parameter	Type: value(s)
<SETTEMP_MINVALUE>	`String` Formatted using selected temperature resolution
<SETTEMP_MAXVALUE>	`String` Formatted using selected temperature resolution
<SETTEMP_MINUNIT> <SETTEMP_MAXUNIT>	`String: "KEL" | "CEL" | "FAR"`

Write:

```
{"SET": "UserMinMaxSetTemperature", "MinSetTemperature": {"Value":
<SETTEMP_MINVALUE>, "Unit": <SETTEMP_MINUNIT>}, "MaxSetTemperature": {"Value":
<SETTEMP_MAXVALUE>, "Unit": <SETTEMP_MAXUNIT>}}
```

Table 11: UserMinMaxSetTemperature write parameters

Parameter	Type: value(s)
<SETTEMP_MINVALUE>	`String` Formatted using selected temperature resolution
<SETTEMP_MAXVALUE>	`String` Formatted using selected temperature resolution
<SETTEMP_MINUNIT> <SETTEMP_MAXUNIT>	`String: "KEL" | "CEL" | "FAR"`

Response:

```
{"SetResponse": "UserMinMaxSetTemperature"}
```

Example: `{"GET": "UserMinMaxSetTemperature"}`

Success `{"GetResponse": "UserMinMaxSetTemperature", "MinSetTemperature": {"Value": "-40.0" , "Unit": "CEL"}, "MaxSetTemperature": {"Value": "150.0" , "Unit": "CEL"}}`

Example: `{"SET": "UserMinMaxSetTemperature", "MinSetTemperature": {"Value": "263.15" , "Unit": "CEL"}, "MaxSetTemperature": {"Value": "373.15" , "Unit": "CEL"}}`

Success `{"SetResponse": "UserMinMaxSetTemperature"}`

SlopeRate

This command sets/reads the slope rate. The value is supplied in $X \cdot min^{-1}$, where *X* is the currently selected temperature unit, e.g., $°C \cdot min^{-1}$. The read command returns both the rate and current temperature unit. The write command has an optional temperature unit parameter.

Read: `{"GET": "SlopeRate"}`

Response:

```
{"GetResponse": "SlopeRate",
 "SlopeRate":
 {"Value": <SLOPE_RATEVALUE>,
  "Unit": <SLOPE_RATEUNIT>
 }
 "MaxSpeed" : <bool>
}
```

Table 12: SlopeRate read response parameters

Parameter	Type: value(s)

<SLOPE_RATEVALUE>	String Formatted using selected temperature resolution
<SLOPE_RATEUNIT>	String: "KEL" | "CEL" | "FAR"

Write:
```
{"SET": "SlopeRate",
 "SlopeRate":
 {"Value": <SLOPE_RATEVALUE>,
  "Unit": <SLOPE_RATEUNIT>
 }
 "MaxSpeed”: <bool>
}
```

Table 13: SlopeRate write parameters

Parameter	Type: value(s)
<SLOPE_RATEVALUE>	String Formatted using selected temperature resolution
<SLOPE_RATEUNIT>	String: "KEL" | "CEL" | "FAR"

Response:
```
{"SetResponse": "SlopeRate"}
```

SibPort

This command sets/reads the input used for SUT – sensor under test.

Note! For B models only.

Read:
```
{"GET": "SibPort", "SibPort": <SIBPORT>}
{"GET": "SibPort"}
```

Table 14: SibPort read request parameters

Parameter	Type: value(s)
<SIBPORT>	Number (integer): 0 | 1 | 2 *Optional* If parameter is omitted, or the value 0 is provided, both SibPort 1 and SibPort 2 are returned. 1: SibPort 1 2: SibPort 2

Response:

0 or empty
```
{"GetResponse": "SibPort", "SibPort1": <SIBPORT1>, "SibPort2": <SIBPORT2>}
```
1
```
{"GetResponse": "SibPort", "SibPort1": <SIBPORT1>}
```
2
```
{"GetResponse": "SibPort", "SibPort2": <SIBPORT2>}
```

Table 15: SibPort read response parameters

Parameter	Type: value(s)
`<SIBPORT1>` `<SIBPORT2>`	`String: "None" | "Voltage" | "Current" | "RTD" | "TC" | "Switch"`

Write:

Sensor 1 and Sensor 2:
```
{"SET": "SibPort", "SibPort1": <SIBPORT1>, "SibPort2": <SIBPORT2>}
```
Sensor 1:
```
{"SET": "SibPort", "SibPort1": <SIBPORT1>}
```
Sensor 2:
```
{"SET": "SibPort", "SibPort2": <SIBPORT2>}
```

Table 16: SibPort write parameters

Parameter	Type: value(s)
`<SIBPORT1>` `<SIBPORT2>`	`String: "None" | "Voltage" | "Current" | "RTD" | "TC" | "Switch"`

Response:
```
{"SetResponse": "SibPort"}
```

SibTCPort

This command sets/reads the SUT – sensor under test thermocouple (mV) type.

Note! For B models only.

Read:
```
{"GET": "SibTCPort", "SibPort": <SIBPORT>}
{"GET": "SibTCPort"}
```

Table 17: SibTCPort read request parameters

Parameter	Type: value(s)
`<SIBPORT>`	`Number (integer): 0 | 1 | 2` *Optional* If parameter is omitted, or the value 0 is provided, both SibPort 1 and SibPort 2 are returned. 1: SibPort 1 2: SibPort 2

Response:

0 or empty
```
{
   "GetResponse": "SibTCPort",
   "SibPort1": {
      "CompensationMode": <COMPENSATION_MODE1>,
      "ManualTemp": {"Value": <MANUAL_TEMPVALUE1>, "Unit": <MANUAL_TEMPUNIT1>},
      "SensorType": <SENSOR_TYPE1>
   },
   "SibPort2": {
      "CompensationMode": <COMPENSATION_MODE2>,
      "ManualTemp": {"Value": <MANUAL_TEMPVALUE2>, "Unit": <MANUAL_TEMPUNIT2>},
      "SensorType": <SENSOR_TYPE2>
   }
```

```
}
```

1

```
{"GetResponse": "SibTCPort", "SibPort1": {"CompensationMode": <COMPENSATION_MODE1>,
"ManualTemp": {"Value": <MANUAL_TEMPVALUE1>, "Unit": <MANUAL_TEMPUNIT1>},
"SensorType": <SENSOR_TYPE1>} }
```

2

```
{"GetResponse": "SibTCPort", "SibPort2": {"CompensationMode": <COMPENSATION_MODE2>,
"ManualTemp": {"Value": <MANUAL_TEMPVALUE2>, "Unit": <MANUAL_TEMPUNIT2>},
"SensorType": <SENSOR_TYPE2>} }
```

Table 18: SibTCPort read response parameters

Parameter	Type: value(s)
<COMPENSATION_MODE1> <COMPENSATION_MODE2>	String: "Auto" | "Manual"
<SENSOR_TYPE1> <SENSOR_TYPE2>	String: "B" | "E" | "J" | "K" | "L" | "N" | "R" | "S" | "T" | "U"
<MANUAL_TEMPVALUE1> <MANUAL_TEMPVALUE2>	String Formatted using selected temperature resolution
<MANUAL_TEMPUNIT1> <MANUAL_TEMPUNIT2>	String: "KEL" | "CEL" | "FAR"

Write:

Sensor 1 and Sensor 2:

```
{"SET": "SibTCPort", "SibPort1": {"CompensationMode": <COMPENSATION_MODE1>,
"ManualTemp": {"Value": <MANUAL_TEMPVALUE1>, "Unit": <MANUAL_TEMPUNIT1>},
"SensorType": <SENSOR_TYPE1>}, "SibPort2": {"CompensationMode":
<COMPENSATION_MODE2>, "ManualTemp": {"Value": <MANUAL_TEMPVALUE2>, "Unit":
<MANUAL_TEMPUNIT2>}, "SensorType": <SENSOR_TYPE2>}}
```

Sensor 1:

```
{"SET": "SibTCPort", "SibPort1": {"CompensationMode": <COMPENSATION_MODE1>,
"ManualTemp": {"Value": <MANUAL_TEMPVALUE1>, "Unit": <MANUAL_TEMPUNIT1>},
"SensorType": <SENSOR_TYPE1>} }
```

Sensor 2:

```
{"SET": "SibTCPort", "SibPort2": {"CompensationMode": <COMPENSATION_MODE2>,
"ManualTemp": {"Value": <MANUAL_TEMPVALUE2>, "Unit": <MANUAL_TEMPUNIT2>},
"SensorType": <SENSOR_TYPE2>} }
```

Table 19: SibTCPort write parameters

Parameter	Type: value(s)
<COMPENSATION_MODE1> <COMPENSATION_MODE2>	String: "Automatic" | "Auto" | "Manual" or Number (integer): 0 | 1 If a string is supplied, "Automatic" or "Auto" both refer to automatic compensation mode. If a number is supplied, 0=Automatic, 1=Manual compensation mode.
<SENSOR_TYPE1> <SENSOR_TYPE2>	String: "B" | "E" | "J" | "K" | "L" | "N" | "R" | "S" | "T" | "U"
<MANUAL_TEMPVALUE1> <MANUAL_TEMPVALUE2>	String Formatted using selected temperature resolution
<MANUAL_TEMPUNIT1> <MANUAL_TEMPUNIT2>	String: "KEL" | "CEL" | "FAR" Optional

Response:

```
{"SetResponse": "SibTCPort"}
```

SibRTDPort

This command sets/reads the setting for the SUT – sensor under test resistance temperature detector (ohm) type.

Note! For B models only.

Read:

```
{"GET": "SibRTDPort", "SibPort": <SIBPORT>}
{"GET": "SibRTDPort"}
```

Table 20: SibRTDPort read request parameters

<table>
<tr><th>Parameter</th><th>Type: value(s)</th></tr>
<tr><td><SIBPORT></td><td>Number (integer): 0 | 1 | 2
Optional
If parameter is omitted, or the value 0 is provided, both SibPort 1 and SibPort 2 are returned.
1: SibPort 1
2: SibPort 2</td></tr>
</table>

Response:

0 or empty

```
{
   "GetResponse": "SibRTDPort",
   "SibPort1": {     "Wires": <WIRES1>,
      "SensorType": <SENSOR_TYPE1>
   },
   "SibPort2": {
      "Wires": <WIRES2>, "SensorType": <SENSOR_TYPE2>
   }
}
```

1

```
{"GetResponse": "SibRTDPort", "SibPort1": {"Wires": <WIRES1>, "SensorType": 
<SENSOR_TYPE1>} }
```

2

```
{"GetResponse": "SibRTDPort", "SibPort2": {"Wires": <WIRES2>, "SensorType": 
<SENSOR_TYPE2>} }
```

Table 21: SibRTDPort read response parameters

<table>
<tr><th>Parameter</th><th>Type: value(s)</th></tr>
<tr><td><SENSOR_TYPE1>
<SENSOR_TYPE2></td><td>String: "P10(90)385" | …
See table **Fejl! Henvisningskilde ikke fundet.** – RTD type strings</td></tr>
<tr><td><WIRES1>
<WIRES2></td><td>Number (integer): 2 | 3 | 4</td></tr>
</table>

Sensor 1 and Sensor 2:

Write:

```
{"SET": "SibRTDPort", "SibPort1": {"Wires": <WIRES1>, "SensorType": <SENSOR_TYPE1>}, 
"SibPort2": {"Wires": <WIRES2>, "SensorType": <SENSOR_TYPE2>}}
```

Sensor 1:

```
{"SET": "SibRTDPort", "SibPort1": {"Wires": <WIRES1>, "SensorType": <SENSOR_TYPE1>}}
```

Sensor 2:

```
{"SET": "SibRTDPort", "SibPort2": {"Wires": <WIRES2>, "SensorType": <SENSOR_TYPE2>}}
```

Table 22: SibRTDPort write parameters

Parameter	Type: value(s)
`<SENSOR_TYPE1>` `<SENSOR_TYPE2>`	`String: "P10(90)385" | …` See table**Fejl! Henvisningskilde ikke fundet.** – RTD type strings
`<WIRES1>` `<WIRES2>`	`Number (integer): 2 | 3 | 4`

Response:

```
{"SetResponse": "SibRTDPort"}
```

SibVoltagePort

This command sets/reads the setting for the SUT – sensor under test voltage (V) type.

Note! For B models only.

Read:

```
{"GET": "SibVoltagePort", "SibPort": <SIBPORT>}
{"GET": "SibVoltagePort"}
```

Table 23: SibVoltagePort read request parameters

Parameter	Type: value(s)
`<SIBPORT>`	`Number (integer): 0 | 1 | 2` *Optional* If parameter is omitted, or the value 0 is provided, both SibPort 1 and SibPort 2 are returned. 1: SibPort 1 2: SibPort 2

Response:

0 or empty

```
{
   "GetResponse": "SibVoltagePort",
   "SibPort1": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
      "VoltageLow": {"Value": <VOLTAGE_LOWVALUE1>, "Unit": <VOLTAGE_LOWUNIT1>},
      "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE1>, "Unit": <VOLTAGE_HIGHUNIT1>}
   },
   "SibPort2": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
      "VoltageLow": {"Value": <VOLTAGE_LOWVALUE2>, "Unit": <VOLTAGE_LOWUNIT2>},
      "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE2>, "Unit": <VOLTAGE_HIGHUNIT2>}
   }
}
```

1

```
{"GetResponse": "SibVoltagePort", "SibPort1": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
      "VoltageLow": {"Value": <VOLTAGE_LOWVALUE1>, "Unit": <VOLTAGE_LOWUNIT1>},
      "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE1>, "Unit": <VOLTAGE_HIGHUNIT1>}
   }
```

```
}
```

2

```
{"GetResponse": "SibVoltagePort", "SibPort2": {
     "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
     "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
     "VoltageLow": {"Value": <VOLTAGE_LOWVALUE2>, "Unit": <VOLTAGE_LOWUNIT2>},
     "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE2>, "Unit": <VOLTAGE_HIGHUNIT2>}
  }
}
```

Table 24: SibVoltagePort read response parameters

Parameter	Type: value(s)
<TEMPERATURE_HIGHVALUE1> <TEMPERATURE_HIGHVALUE2> <TEMPERATURE_LOWVALUE1> <TEMPERATURE_LOWVALUE2>	`String` Formatted using selected temperature resolution
<VOLTAGE_HIGHVALUE1> <VOLTAGE_HIGHVALUE2> <VOLTAGE_LOWVALUE1> <VOLTAGE_LOWVALUE2>	`String` Voltages are in V (volt) or mV (millivolt), depending on the selected range.
<TEMPERATURE_LOWUNIT1> <TEMPERATURE_HIGHUNIT1> <TEMPERATURE_LOWUNIT2> <TEMPERATURE_HIGHUNIT2>	`String: "KEL" | "CEL" | "FAR"`
<VOLTAGE_LOWUNIT1> <VOLTAGE_HIGHUNIT1> <VOLTAGE_LOWUNIT2> <VOLTAGE_HIGHUNIT2>	`String: "V"`

Write:

Sensor 1 and Sensor 2:

```
{ "SET": "SibVoltagePort", "SibPort1": {
     "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
     "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
     "VoltageLow": {"Value": <VOLTAGE_LOWVALUE1>, "Unit": <VOLTAGE_LOWUNIT1>},
     "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE1>, "Unit": <VOLTAGE_HIGHUNIT1>}
  },
  "SibPort2": {
     "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
     "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
     "VoltageLow": {"Value": <VOLTAGE_LOWVALUE2>, "Unit": <VOLTAGE_LOWUNIT2>},
     "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE2>, "Unit": <VOLTAGE_HIGHUNIT2>}
  }
}
```

Sensor 1:

```
{"SET": "SibVoltagePort", "SibPort1": {
     "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
     "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
     "VoltageLow": {"Value": <VOLTAGE_LOWVALUE1>, "Unit": <VOLTAGE_LOWUNIT1>},
     "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE1>, "Unit": <VOLTAGE_HIGHUNIT1>}
  },
}
```

Sensor 2:
```
{"SET": "SibVoltagePort", "SibPort2": {
     "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
     "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
     "VoltageLow": {"Value": <VOLTAGE_LOWVALUE2>, "Unit": <VOLTAGE_LOWUNIT2>},
     "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE2>, "Unit": <VOLTAGE_HIGHUNIT2>}
  }
}
```

Table 25: SibVoltagePort write parameters

Parameter	Type: value(s)
<TEMPERATURE_HIGHVALUE1> <TEMPERATURE_HIGHVALUE2> <TEMPERATURE_LOWVALUE1> <TEMPERATURE_LOWVALUE2>	`String` Formatted using selected temperature resolution
<VOLTAGE_HIGHVALUE1> <VOLTAGE_HIGHVALUE2> <VOLTAGE_LOWVALUE1> <VOLTAGE_LOWVALUE2>	`String` Voltages are in V (volt) or mV (millivolt), depending on the selected range.
<TEMPERATURE_LOWUNIT1> <TEMPERATURE_HIGHUNIT1> <TEMPERATURE_LOWUNIT2> <TEMPERATURE_HIGHUNIT2>	`String: "KEL" | "CEL" | "FAR"`
<VOLTAGE_LOWUNIT1> <VOLTAGE_HIGHUNIT1> <VOLTAGE_LOWUNIT2> <VOLTAGE_HIGHUNIT2>	`String: "V"`

Response:
```
{"SetResponse": "SibVoltagePort"}
```

SibCurrentPort

This command sets/reads the setting for the SUT – sensor under test current (mA) type.

Note! For B models only.

Read:
```
{"GET": "SibCurrentPort", "SibPort": <SIBPORT>}
{"GET": "SibCurrentPort"}
```

Table 26: SibCurrentPort read request parameters

Parameter	Type: value(s)
<SIBPORT>	`Number (integer): 0 | 1 | 2` *Optional* If parameter is omitted, or the value 0 is provided, both SibPort 1 and SibPort 2 are returned. 1: SibPort 1 2: SibPort 2

Response:

0 or empty
```
{
  "GetResponse": "SibCurrentPort",
  "SibPort1": {
```

```
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE1>, "Unit": <CURRENT_LOWUNIT1>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE1>, "Unit": <CURRENT_HIGHUNIT1>},
   "LoopPower": <bool>
   },
   "SibPort2": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE2>, "Unit": <CURRENT_LOWUNIT2>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE2>, "Unit": <CURRENT_HIGHUNIT2>},
      "LoopPower": <bool>
   }
}
```

1

```
{"GetResponse": "SibCurrentPort", "SibPort1": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE1>, "Unit": <CURRENT_LOWUNIT1>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE1>, "Unit": <CURRENT_HIGHUNIT1>},
      "LoopPower": <bool>
   }
}
```

2

```
{"GetResponse": "SibVoltagePort", "SibPort2": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE2>, "Unit": <CURRENT_LOWUNIT2>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE2>, "Unit": <CURRENT_HIGHUNIT2>},
      "LoopPower": <bool>
   }
}
```

Table 27: SibCurrentPort read response parameters

Parameter	Type: value(s)
<TEMPERATURE_HIGHVALUE1> <TEMPERATURE_HIGHVALUE2> <TEMPERATURE_LOWVALUE1> <TEMPERATURE_LOWVALUE2>	String Formatted using selected temperature resolution
<CURRENT_HIGHVALUE1> <CURRENT_HIGHVALUE2> <CURRENT_LOWVALUE1> <CURRENT_LOWVALUE2>	String Currents are in mA (milliampere)
<TEMPERATURE_LOWUNIT1> <TEMPERATURE_HIGHUNIT1> <TEMPERATURE_LOWUNIT2> <TEMPERATURE_HIGHUNIT2>	String: "KEL" | "CEL" | "FAR"
<CURRENT_LOWUNIT1> <CURRENT_HIGHUNIT1> <CURRENT_LOWUNIT2> <CURRENT_HIGHUNIT2>	String: "mA"

Write:

Sensor 1 and Sensor 2:

```
{ "SET": "SibCurrentPort", "SibPort1": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE1>, "Unit": <CURRENT_LOWUNIT1>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE1>, "Unit": <CURRENT_HIGHUNIT1>}
   },
   "SibPort2": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE2>, "Unit": <CURRENT_LOWUNIT2>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE2>, "Unit": <CURRENT_HIGHUNIT2>}
   }
}
```

Sensor 1:

```
{"SET": "SibCurrentPort", "SibPort1": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE1>, "Unit":
<TEMPERATURE_LOWUNIT1>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE1>, "Unit":
<TEMPERATURE_HIGHUNIT1>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE1>, "Unit": <CURRENT_LOWUNIT1>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE1>, "Unit": <CURRENT_HIGHUNIT1>}
   },
}
```

Sensor 2:

```
{"SET": "SibCurrentPort", "SibPort2": {
      "TemperatureLow": {"Value": <TEMPERATURE_LOWVALUE2>, "Unit":
<TEMPERATURE_LOWUNIT2>},
      "TemperatureHigh": {"Value": <TEMPERATURE_HIGHVALUE2>, "Unit":
<TEMPERATURE_HIGHUNIT2>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE2>, "Unit": <CURRENT_LOWUNIT2>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE2>, "Unit": <CURRENT_HIGHUNIT2>}
   }
}
```

Table 28: SibVoltagePort write parameters

Parameter	Type: value(s)
<TEMPERATURE_HIGHVALUE1> <TEMPERATURE_HIGHVALUE2> <TEMPERATURE_LOWVALUE1> <TEMPERATURE_LOWVALUE2>	`String` Formatted using selected temperature resolution
<CURRENT_HIGHVALUE1> <CURRENT_HIGHVALUE2> <CURRENT_LOWVALUE1> <CURRENT_LOWVALUE2>	`String` Currents are in mA (milliampere)
<TEMPERATURE_LOWUNIT1> <TEMPERATURE_HIGHUNIT1> <TEMPERATURE_LOWUNIT2> <TEMPERATURE_HIGHUNIT2>	`String: "KEL" | "CEL" | "FAR"`
<CURRENT_LOWUNIT1> <CURRENT_HIGHUNIT1> <CURRENT_LOWUNIT2> <CURRENT_HIGHUNIT2>	`String: "mA"`

Response:

```
{"SetResponse": "SibCurrentPort"}
```

UseExternalReferenceSensor

This command sets/reads the setting for the True temperature. If "False", the internal reference (Read) temperature is displayed as "True".

Note! For B/C models only.

Read:
```
{"GET": "UseExternalReferenceSensor"}
```

Response:
```
{"GetResponse": "UseExternalReferenceSensor", "UseExternalReferenceSensor":
<USE_EXT_REF>}
```

Table 29: UseExternalReferenceSensor read response parameters

Parameter	Type: value(s)
`<USE_EXT_REF>`	`Bool: true	false`

Write:
```
{"SET": "UseExternalReferenceSensor", "UseExternalReferenceSensor": <USE_EXT_REF>}
```

Table 30: UseExternalReferenceSensor write parameters

Parameter	Type: value(s)
`<USE_EXT_REF>`	`Bool: true	false`

Response:
```
{"SetResponse": "UseExternalReferenceSensor"}
```

EnableSetFollowsXREF

This command sets/reads the "SET-follows-true" feature for the external reference sensor.

Note! For B/C models only.

Read:
```
{"GET": "EnableSetFollowsXREF"}
```

Response:
```
{"GetResponse": "EnableSetFollowsXREF", "EnableSetFollowsXREF": <XREF_SETFOLLOWS>}
```

Table 31: EnableSetFollowsXREF read response parameters

Parameter	Type: value(s)
`<XREF_SETFOLLOWS>`	`Bool: true	false`

Write:
```
{"SET": "EnableSetFollowsXREF", "EnableSetFollowsXREF": <XREF_SETFOLLOWS>}
```

Table 32: EnableSetFollowsXREF write parameters

Parameter	Type: value(s)
`<XREF_SETFOLLOWS>`	`Bool: true	false`

Response:
`{"SetResponse": "EnableSetFollowsXREF"}`

EnableSetFollowsXDIFF

This command sets/reads the "DLC" feature.

Note! For B/C models only.

Read:
`{"GET": "EnableSetFollowsXDIFF"}`

Response:
`{"GetResponse": "EnableSetFollowsXDIFF", "EnableSetFollowsXDIFF": <XDIFF_SETFOLLOWS>}`

Table 33: EnableSetFollowsXDIFF read response parameters

Parameter	Type: value(s)
`<XDIFF_SETFOLLOWS>`	`Bool: true | false`

Write:
`{"SET": "EnableSetFollowsXDIFF", "EnableSetFollowsXDIFF": <XDIFF_SETFOLLOWS>}`

Table 34: EnableSetFollowsXDIFF write parameters

Parameter	Type: value(s)
`<XDIFF_SETFOLLOWS>`	`Bool: true | false`

Response:
`{"SetResponse": "EnableSetFollowsXDIFF"}`

SensorConvertToTemperature

This command sets/reads the "Convert to temperature " feature for the sensor under test.

Note! For B models only.

Read:
`{"GET": "SensorConvertToTemperature", "SibPort": <SIBPORT>}`
`{"GET": "SensorConvertToTemperature"}`

Table 35: SensorConvertToTemperature read request parameters

Parameter	Type: value(s)
`<SIBPORT>`	`Number (integer): 0 | 1 | 2` *Optional* If parameter is omitted, or the value 0 is provided, both SibPort 1 and SibPort 2 are returned. 1: SibPort 1 2: SibPort 2

Response:
0 or empty

```
{"GetResponse": "SensorConvertToTemperature", "SibPort1": <SIBPORT1>, "SibPort2":
<SIBPORT2>}
```

1

```
{"GetResponse": "SensorConvertToTemperature", "SibPort1": <SIBPORT1>}
```

2

```
{"GetResponse": "SensorConvertToTemperature", "SibPort2": <SIBPORT2>}
```

Table 36: SensorConvertToTemperature read response parameters

Parameter	Type: value(s)
<SIBPORT1> <SIBPORT2>	Bool: true|false

Write:

Sensor 1 and Sensor 2:

```
{"SET": "SensorConvertToTemperature", "SibPort1": <SIBPORT1>, "SibPort2":
<SIBPORT2>}
```

Sensor 1:

```
{"SET": "SensorConvertToTemperature", "SibPort1": <SIBPORT1>}
```

Sensor 2:

```
{"SET": "SensorConvertToTemperature", "SibPort2": <SIBPORT2>}
```

Table 37: SensorConvertToTemperature write parameters

Parameter	Type: value(s)
<SIBPORT1> <SIBPORT2>	Bool: true|false

Response:

```
{"SetResponse": "SensorConvertToTemperature"}
```

XRefConvertToTemperature

This command sets/reads the "Convert to temperature " feature for the external reference sensor.

Note! For B/C models only.

Read:

```
{"GET": "XRefConvertToTemperature"}
```

Response:

```
{"GetResponse": "XRefConvertToTemperature", "XRefConvertToTemperature":
<XREF_CONVTOTEMP>}
```

Table 38: XRefConvertToTemperature read response parameters

Parameter	Type: value(s)
<XREF_CONVTOTEMP>	Bool: true | false

Write:

```
{"SET": "XRefConvertToTemperature", "XRefConvertToTemperature": <XREF_CONVTOTEMP>}
```

Table 39: XRefConvertToTemperature write parameters

Parameter	Type: value(s)
<XREF_CONVTOTEMP>	Bool: true | false

Response:
```
{"SetResponse": "XRefConvertToTemperature"}
```

StirrerSpeed

This command sets/reads the stirrer speed (RTCt-168).

Read:
```
{"GET": "StirrerSpeed"}
```

Response:
```
{"GetResponse": "StirrerSpeed", "StirrerSpeed": <STIRRER_SPEED>}
```

Table 40: StirrerSpeed read response parameters

Parameter	Type: value(s)
<STIRRER_SPEED>	Number (integer): [0-100]

Write:
```
{"SET": "StirrerSpeed", "StirrerSpeed": <STIRRER_SPEED>}
```

Table 41: StirrerSpeed write parameters

Parameter	Type: value(s)
<STIRRER_SPEED>	Number (integer): [0-100]

Response:
```
{"SetResponse": "StirrerSpeed"}
```

RibXREFPort

This command loads a segment of the intelligent probes data inserted in the Reference sensor port

Read:
```
{"GET": "RibXREFPort"}
```

Response:
```
{"GetResponse": "RibXREFPort",
"SerialNumber": <XREF_SERIAL_NUMBER>,
"MaxTemperature": {"Value": <XREF_MAXTEMPVALUE>, "Unit": <XREF_MAXTEMPUNIT>},
"MinTemperature": {"Value": <XREF_MINTEMPVALUE>, "Unit": <XREF_MINTEMPUNIT>},
"CalibrationDate": <XREF_CALIBRATION_DATE>,
"CalibrationInitials": <XREF_CALIBRATION_INITIALS>,
"CalibrationPeriod": <XREF_CALIBRATION_PERIOD>,
"CoefficientsType": <XREF_COEF_TYPE>,
"Coefficients": <XREF_COEFF>}
```

CvD:
```
<XREF_COEFF> = {
   "R0": <XREF_COEF_R0>,
   "A": <XREF_COEF_A>,
   "B": <XREF_COEF_B>,
   "C": <XREF_COEF_C>
}
```

ITS-90:
```
<XREF_COEFF> = {
```

```
   "RTPW": <XREF_COEF_RTPW>,
   "ALR": <XREF_COEF_ALR>,
   "BLR": <XREF_COEF_BLR>,
   "CLR": <XREF_COEF_CLR>,
   "C2LR": <XREF_COEF_C2LR>,
   "C3LR": <XREF_COEF_C3LR>,
   "C4LR": <XREF_COEF_C4LR>,
   "C5LR": <XREF_COEF_C5LR>,
   "AHR": <XREF_COEF_AHR>,
   "BHR": <XREF_COEF_BHR>,
   "CHR": <XREF_COEF_CHR>,
   "DHR": <XREF_COEF_DHR>,
   "W": <XREF_COEF_W>
}
```

Table 42: RibXREFPort read response parameters

Parameter	Type: value(s)
<XREF_SERIAL_NUMBER>	String
<XREF_MAXTEMPVALUE> <XREF_MINTEMPVALUE>	String Formatted using selected temperature resolution
<XREF_MAXTEMPUNIT> <XREF_MINTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<XREF_CALIBRATION_DATE>	String: "YYYY-MM-DD" YYYY = 4 digit year MM = 2 digit month (January = 01, December = 12) DD = 2 digit day of month Example: May 3rd, 2021: "2021-05-03"
<XREF_CALIBRATION_INITIALS>	String
<XREF_CALIBRATION_PERIOD>	Number (integer) Calibration period in days
<XREF_COEF_TYPE>	Number (integer): 0 | 1 0=CVD 1=ITS90
<XREF_COEF_RTPW>	Number (float)
<XREF_COEF_ALR>	Number (float)
<XREF_COEF_BLR>	Number (float)
<XREF_COEF_CLR>	Number (float)
<XREF_COEF_C2LR>	Number (float)
<XREF_COEF_C3LR>	Number (float)
<XREF_COEF_C4LR>	Number (float)
<XREF_COEF_C5LR>	Number (float)
<XREF_COEF_AHR>	Number (float)
<XREF_COEF_BHR>	Number (float)
<XREF_COEF_CHR>	Number (float)
<XREF_COEF_DHR>	Number (float)
<XREF_COEF_W>	Number (float)
<XREF_COEF_R0>	Number (float)
<XREF_COEF_A>	Number (float)
<XREF_COEF_B>	Number (float)
<XREF_COEF_C>	Number (float)

RibXDiffPort

This command loads a segment of the intelligent probes data inserted in the Differential sensor port

Note! For B/C models only.

Read:

```
{"GetResponse": "RibXDIFFPort",
"SerialNumber": <XDIFF_SERIAL_NUMBER>,
"MaxTemperature": {"Value": <XDIFF_MAXTEMPVALUE>, "Unit": <XDIFF_MAXTEMPUNIT>},
```

```
"MinTemperature": {"Value": <XDIFF_MINTEMPVALUE>, "Unit": <XDIFF_MINTEMPUNIT>},
"CalibrationDate": <XDIFF_CALIBRATION_DATE>,
"CalibrationInitials": <XDIFF_CALIBRATION_INITIALS>,
"CalibrationPeriod": <XDIFF_CALIBRATION_PERIOD>,
"CoefficientsType": <XDIFF_COEF_TYPE>,
"Coefficients": {
   "A": <XDIFF_COEF_A>,
   "B": <XDIFF_COEF_B>,
   "C": <XDIFF_COEF_C>
}
}
```

Table 43: RibXDiffPort read response parameters

Parameter	Type: value(s)
<XDIFF_SERIAL_NUMBER>	String
<XDIFF_MAXTEMPVALUE> <XDIFF_MINTEMPVALUE>	String Formatted using selected temperature resolution
<XDIFF_MAXTEMPUNIT> <XDIFF_MINTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<XDIFF_CALIBRATION_DATE>	String: "YYYY-MM-DD" YYYY = 4 digit year MM = 2 digit month (January = 01, December = 12) DD = 2 digit day of month Example: May 3rd, 2021: "2021-05-03"
<XDIFF_CALIBRATION_PERIOD>	Number (integer) Calibration period in days
<XDIFF_COEF_TYPE>	Number (integer): 0..14 0 B 1 E 2 J 3 K 4 N 5 R 6 S 7 T 8 Non-standard 9 L 10 U 11 XK 12 BP 13 C 14 CJC-only
<XDIFF_COEF_A>	Number (float)
<XDIFF_COEF_B>	Number (float)
<XDIFF_COEF_C>	Number (float)

CalibratorDevice

The returns data elements identifying the instrument type, configuration settings and calibration dates.

Read: `{"GET": "CalibratorDevice"}`

Response on success:

```
{"GetResponse": "CalibratorDevice",
"SerialNumber": <SERIAL_NUMBER>,
"ProtocolVersion": <PROTOCOL_VERSION>,
"SWVersion": <MB_SWVERSION>,
"HWVersion": <MB_HWVERSION>,
"ModelId": <MODEL_ID>,
"Model": <MODEL_NAME>,
"ModelVariant": <MODEL_VARIANT>,
"CBSWVersion": <CB_SWVERSION>,
"CBHWVersion": <CB_HWVERSION>,
"HasSilentMode": <OPT_SILENT_MODE>,
"HasStirrer": <OPT_STIRRER>,
"HasFPSC": <OPT_FPSC>,
"FactoryMinSetTemperature": {"Value": <FACTORY_SETTEMP_MINVALUE>, "Unit":
<FACTORY_SETTEMP_MINUNIT>},
"FactoryMaxSetTemperature": {"Value": <FACTORY_SETTEMP_MAXVALUE>, "Unit":
<FACTORY_SETTEMP_MAXUNIT>},
"MinSetTemperature": {"Value": <SETTEMP _MINVALUE>, "Unit": <SETTEMP_MINUNIT>},
"MaxSetTemperature": {"Value": <SETTEMP _MINVALUE>, "Unit": <SETTEMP_MINUNIT>},
"MainsFrequency": <MAINS_FREQUENCY>,
"MainsFrequencyAccepted": <FREQUENCY_OK>,
"EnableReferenceInputBoardFailed": <ENABLE_RIB_FAILED>,
"EnableSensorInputBoardFailed": <ENABLE_SIB_FAILED>,
"IsReferenceInputBoardCalibrated": <RIB_CALIBRATED>,
"IsSensorInputBoardCalibrated": <SIB_CALIBRATED>
}
```

Table 44: CalibratorDevice read response parameters

Parameter	Type: value(s)
<SERIAL_NUMBER>	String (e.g. "123456-12345")
<PROTOCOL_VERSION>	Number (float)
<MB_SWVERSION>	String (e.g. "1.0.1257")
<MB_HWVERSION>	Number (integer)
<MODEL_ID>	Number (integer)
<MODEL_NAME>	String (e.g. "RTCt-157 B")
<MODEL_VARIANT>	String: "A" | "B" | "C"
<CB_SWVERSION>	String (e.g. "2.57")
<CB_HWVERSION>	Number (integer)
<OPT_SILENT_MODE>	Bool: true|false
<OPT_STIRRER>	Bool: true|false
<OPT_FPSC>	Bool: true|false
<FACTORY_SETTEMP_MAXVALUE> <FACTORY_SETTEMP_MINVALUE>	String Formatted using selected temperature resolution
<FACTORY_SETTEMP_MAXUNIT> <FACTORY_SETTEMP_MINUNIT>	String: "KEL" | "CEL" | "FAR"
<SETTEMP_MAXVALUE> <SETTEMP_MINVALUE>	String Formatted using selected temperature resolution
<SETTEMP_MAXUNIT> <SETTEMP_MINUNIT>	String: "KEL" | "CEL" | "FAR"
<MAINS_FREQUENCY>	Number (integer): 0 | 1 | 2 0 = Any 1 = 50 Hz only 2 = 60 Hz only
<FREQUENCY_OK>	Bool: true|false
<ENABLE_RIB_FAILED>	Bool: true|false
<ENABLE_SIB_FAILED>	Bool: true|false

<RIB_CALIBRATED>	Bool: true|false
<SIB_CALIBRATED>	Bool: true|false

Write: -

RibXREFPortProbe

This command reads and programs the relevant data for an RTD type intelligent probe connected to the Reference sensor port.

Read:

```
{"GET": "RibXREFPortProbe"}
```

Response:

```
{"GetResponse": "RibXREFPortProbe",
"ManufacturerId": <XREF_MANUFACTURER_ID>,
"ModelNumber": <XREF_MODEL_NUMBER>,
"VersionLetter": <XREF_VERSION_LETTER>,
"VersionNumber": <XREF_VERSION_NUMBER>,
"SerialNumber": <XREF_SERIAL_NUMBER>,
"Template": <XREF_TEMPLATE>,
"MinTemperature": {"Value": <XREF_MINTEMPVALUE>, "Unit": <XREF_MINTEMPUNIT>},
"MaxTemperature": {"Value": <XREF_MAXTEMPVALUE>, "Unit": <XREF_MAXTEMPUNIT>},
"MinElecVal": {"Value": <XREF_ELECVAL_MINVALUE>, "Unit": <XREF_ELECVAL_MINUNIT>},
"MaxElecVal": {"Value": <XREF_ELECVAL_MAXVALUE>, "Unit": <XREF_ELECVAL_MAXUNIT>},
"CoefficientsType": <XREF_COEF_TYPE>,
"SignalType": <XREF_SIGNAL_TYPE>,
"MapMethod": <XREF_MAP_METHOD>,
"CalibrationDate": <XREF_CALIBRATION_DATE>,
"CalibrationInitials": <XREF_CALIBRATION_INITIALS>,
"CalibrationPeriod": <XREF_CALIBRATION_PERIOD>,
"SensorMeasID": <XREF_MEAS_ID>,
"SensorResponseTime   ": <XREF_RESPONSE_TIME>,
"SensorExcitation": <XREF_EXCITATION>,
"SensorMaxExcitation": <XREF_MAX_EXCITATION>,
"SensorSelfHeating": <XREF_SELF_HEATING>,
"Coefficients": <XREF_COEFF>
}
```

CvD:

```
<XREF_COEFF> = {
   "R0": <XREF_COEF_R0>,
   "A": <XREF_COEF_A>,
   "B": <XREF_COEF_B>,
   "C": <XREF_COEF_C>
}
```

ITS-90:

```
<XREF_COEFF> = {
   "RTPW": <XREF_COEF_RTPW>,
   "ALR": <XREF_COEF_ALR>,
   "BLR": <XREF_COEF_BLR>,
   "CLR": <XREF_COEF_CLR>,
   "C2LR": <XREF_COEF_C2LR>,
   "C3LR": <XREF_COEF_C3LR>,
   "C4LR": <XREF_COEF_C4LR>,
   "C5LR": <XREF_COEF_C5LR>,
   "AHR": <XREF_COEF_AHR>,
   "BHR": <XREF_COEF_BHR>,
   "CHR": <XREF_COEF_CHR>,
   "DHR": <XREF_COEF_DHR>,
   "W": <XREF_COEF_W>
}
```

Table 45: RibXREFPortProbe read response parameters

Parameter	Type: value(s)
<XREF_MANUFACTURER_ID>	Number (integer)
<XREF_MODEL_NUMBER>	Number (integer)
<XREF_VERSION_NUMBER>	Number (integer)
<XREF_VERSION_LETTER>	String (1 character)
<XREF_SERIAL_NUMBER>	Number (integer)
<XREF_TEMPLATE>	Number (integer)
<XREF_MINTEMPVALUE> <XREF_MAXTEMPVALUE>	String
<XREF_MAXTEMPUNIT> <XREF_MINTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<XREF_ELECVAL_MINVALUE> <XREF_ELECVAL_MMAXVALUE>	Number (float)
<XREF_ELECVAL_MINUNIT> <XREF_ELECVAL_MAXUNIT>	String "Ohm"
<XREF_COEF_TYPE>	Number (integer): 0 | 1 0=CVD 1=ITS90
<XREF_SIGNAL_TYPE>	Number (integer): VoltageSensor = 0, CurrentSensor = 1, ResistanceSensor = 2, BridgeSensor = 3, LVDTSensor = 4, PotentiometricVoltageDividerSensor = 5, PulseSensor = 6, VoltageActuator = 7, CurrentActuator = 8, PulseActuator = 9
<XREF_MAP_METHOD>	Number (integer): Linear = 0, Inverse_m_xb = 1, Inverse_bm_x = 2, Inverse_x_bm = 3, Thermocouple = 4, Thermistor = 5, RTD = 6, Bridge = 7
<XREF_CALIBRATION_DATE>	String: "YYYY-MM-DD" YYYY = 4 digit year MM = 2 digit month (January = 01, December = 12) DD = 2 digit day of month Example: May 3rd, 2021: "2021-05-03"
<XREF_CALIBRATION_INITIALS>	String
<XREF_CALIBRATION_PERIOD>	Number (integer) Calibration period in days
<XREF_MEAS_ID>	Number (integer)
<XREF_RESPONSE_TIME>	Number (float)
<XREF_EXCITATION>	Number (float)
<XREF_EXTICATION_MAX>	Number (float)
<XREF_SELF_HEATING>	Number (float)
<XREF_COEF_RTPW>	Number (float)
<XREF_COEF_ALR>	Number (float)
<XREF_COEF_BLR>	Number (float)
<XREF_COEF_CLR>	Number (float)
<XREF_COEF_C2LR>	Number (float)
<XREF_COEF_C3LR>	Number (float)
<XREF_COEF_C4LR>	Number (float)

<XREF_COEF_C5LR>	Number (float)
<XREF_COEF_AHR>	Number (float)
<XREF_COEF_BHR>	Number (float)
<XREF_COEF_CHR>	Number (float)
<XREF_COEF_DHR>	Number (float)
<XREF_COEF_W>	Number (float)
<XREF_COEF_R0>	Number (float)
<XREF_COEF_A>	Number (float)
<XREF_COEF_B>	Number (float)
<XREF_COEF_C>	Number (float)

Write:

```
{"SET": "RibXREFPortProbe",
"ManufacturerId": <XREF_MANUFACTURER_ID>,
"ModelNumber": <XREF_MODEL_NUMBER>,
"VersionLetter": <XREF_VERSION_LETTER>,
"VersionNumber": <XREF_VERSION_NUMBER>,
"SerialNumber": <XREF_SERIAL_NUMBER>,
"Template": <XREF_TEMPLATE>,
"MinTemperature": {"Value": <XREF_MINTEMPVALUE>, "Unit": <XREF_MINTEMPUNIT>},
"MaxTemperature": {"Value": <XREF_MAXTEMPVALUE>, "Unit": <XREF_MAXTEMPUNIT>},
"MinElecVal": {"Value": <XREF_ELECVAL_MINVALUE>, "Unit": <XREF_ELECVAL_MINUNIT>},
"MaxElecVal": {"Value": <XREF_ELECVAL_MAXVALUE>, "Unit": <XREF_ELECVAL_MAXUNIT>},
"CoefficientsType": <XREF_COEF_TYPE>,
"SignalType": <XREF_SIGNAL_TYPE>,
"MapMethod": <XREF_MAP_METHOD>,
"CalibrationDate": <XREF_CALIBRATION_DATE>,
"CalibrationInitials": <XREF_CALIBRATION_INITIALS>,
"CalibrationPeriod": <XREF_CALIBRATION_PERIOD>,
"SensorMeasID": <XREF_MEAS_ID>,
"SensorResponseTime": <XREF_RESPONSE_TIME>,
"SensorExcitation": <XREF_EXCITATION>,
"SensorMaxExcitation": <XREF_MAX_EXCITATION>,
"SensorSelfHeating": <XREF_SELF_HEATING>,
"Coefficients": <XREF_COEFF>
}
```

CvD:

```
<XREF_COEFF> = {
   "R0": <XREF_COEF_R0>,
   "A": <XREF_COEF_A>,
   "B": <XREF_COEF_B>,
   "C": <XREF_COEF_C>
}
```

ITS-90:

```
<XREF_COEFF> = {
   "RTPW": <XREF_COEF_RTPW>,
   "ALR": <XREF_COEF_ALR>,
   "BLR": <XREF_COEF_BLR>,
   "CLR": <XREF_COEF_CLR>,
   "C2LR": <XREF_COEF_C2LR>,
   "C3LR": <XREF_COEF_C3LR>,
   "C4LR": <XREF_COEF_C4LR>,
   "C5LR": <XREF_COEF_C5LR>,
   "AHR": <XREF_COEF_AHR>,
   "BHR": <XREF_COEF_BHR>,
   "CHR": <XREF_COEF_CHR>,
   "DHR": <XREF_COEF_DHR>,
   "W": <XREF_COEF_W>
}
```

Table 46: RibXREFPortProbe write parameters

Parameter	Type: value(s)
Consult Table 45: RibXREFPortProbe read response parameters	

Response on success:

```
{"SetResponse": "RibXREFPortProbe"}
```

RibXDIFFPortProbe

This command reads and programs the relevant data for a Thermocouple type intelligent probe connected to the DLC sensor port.

Note! For B/C models only.

Read:

```
{"GET": "RibXDIFFPortProbe"}
```

Response:

```
{"GetResponse": "RibXDIFFPortProbe",
"ManufacturerId": <XDIFF_MANUFACTURER_ID>,
"ModelNumber": <XDIFF_MODEL_NUMBER>,
"VersionLetter": <XDIFF_VERSION_LETTER>,
"VersionNumber": <XDIFF_VERSION_NUMBER>,
"SerialNumber": <XDIFF_SERIAL_NUMBER>,
"Template": <XDIFF_TEMPLATE>,
"MinTemperature": {"Value": <XDIFF_MINTEMPVALUE>, "Unit": <XDIFF_MINTEMPUNIT>},
"MaxTemperature": {"Value": <XDIFF_MAXTEMPVALUE>, "Unit": <XDIFF_MAXTEMPUNIT>},
"MinElecVal": {"Value": <XDIFF_ELECVAL_MINVALUE>, "Unit": <XDIFF_ELECVAL_MINUNIT>},
"MaxElecVal": {"Value": <XDIFF_ELECVAL_MAXVALUE>, "Unit": <XDIFF_ELECVAL_MAXUNIT>},
"CoefficientsType": <XDIFF_COEF_TYPE>,
"SignalType": <XDIFF_SIGNAL_TYPE>,
"MapMethod": <XDIFF_MAP_METHOD>,
"CalibrationDate": <XDIFF_CALIBRATION_DATE>,
"CalibrationInitials": <XDIFF_CALIBRATION_INITIALS>,
"CalibrationPeriod": <XDIFF_CALIBRATION_PERIOD>,
"SensorMeasID": <XDIFF_MEAS_ID>,
"SensorResponseTime": <XDIFF_RESPONSE_TIME>,
"BuiltInRTDForCJ": <XDIFF_INBUILT_RTD>,
"ColdJunctionSource": <XDIFF_CJ_SOURCE>,
"SensorExcitation": <XDIFF_EXCITATION>,
"SensorMaxExcitation": <XDIFF_MAX_EXCITATION>,
"SensorSelfHeating": <XDIFF_SELF_HEATING>,
"SensorImpedance": <XDIFF_IMPEDANCE>,
"Coefficients": {
   "ColdJunction": {
      "R0": <XDIFF_COEF_CJ_R0>,
      "A": <XDIFF_COEF_CJ_A>,
      "B": <XDIFF_COEF_CJ_B>,
      "C": <XDIFF_COEF_CJ_C>
   },
   "TC": {
      "A": <XDIFF_COEF_TC_A>,
      "B": <XDIFF_COEF_TC_B>,
      "C": <XDIFF_COEF_TC_C>
   }
}
```

Table 47: RibXDIFFPortProbe read response parameters

Parameter	Type: value(s)
<XDIFF_MANUFACTURER_ID>	Number (integer)
<XDIFF_MODEL_NUMBER>	Number (integer)
<XDIFF_VERSION_NUMBER>	Number (integer)
<XDIFF_VERSION_LETTER>	String (1 character)
<XDIFF_SERIAL_NUMBER>	Number (integer)
<XDIFF_TEMPLATE>	Number (integer) =0 (TC)
<XDIFF_MINTEMPVALUE> <XDIFF_MAXTEMPVALUE>	String
<XDIFF_MAXTEMPUNIT> <XDIFF_MINTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<XDIFF_ELECVAL_MAXVALUE> <XDIFF_ELECVAL_MINVALUE>	Number (float)
<XDIFF_ELECVAL_MAXUNIT> <XDIFF_ELECVAL_MINUNIT>	String Thermcouple/DLC "Voltage"
<XDIFF_COEF_TYPE>	Number (integer): 0..14 0 B 1 E 2 J 3 K 4 N 5 R 6 S 7 T 8 Non-standard 9 L 10 U 11 XK 12 BP 13 C 14 CJC-only
<XDIFF_SIGNAL_TYPE>	Number (integer): VoltageSensor = 0, CurrentSensor = 1, ResistanceSensor = 2, BridgeSensor = 3, LVDTSensor = 4, PotentiometricVoltageDividerSensor = 5, PulseSensor = 6, VoltageActuator = 7, CurrentActuator = 8, PulseActuator = 9

<XDIFF_MAP_METHOD>	Number (integer):
	Linear = 0,
	Inverse_m_xb = 1, // m/(x+b)
	Inverse_bm_x = 2, // (b+m/x)
	Inverse_x_bm = 3, // 1/(b+m/x)
	Thermocouple = 4,
	Thermistor = 5,
	RTD = 6,
	Bridge = 7

<XDIFF_CALIBRATION_DATE>	String: "YYYY-MM-DD" YYYY = 4 digit year MM = 2 digit month (January = 01, December = 12) DD = 2 digit day of month Example: May 3rd, 2021: "2021-05-03"
<XDIFF_CALIBRATION_INITIALS>	String
<XDIFF_CALIBRATION_PERIOD>	Number (integer) Calibration period in days
<XDIFF_MEAS_ID>	Number (integer)
<XDIFF_RESPONSE_TIME>	Number (float)
<XDIFF_INBUILT_RTD>	Bool: true|false
<XDIFF_CJ_SOURCE>	Number (integer): 0 | 1 | 2

	0 = "No CJ required" 1 = "Built in RTD" (Sensor compensated for 0°C CJ) 2 = "CJ Required"
<XDIFF_EXCITATION>	Number (float)
<XDIFF_EXTICATION_MAX>	Number (float)
<XDIFF_SELF_HEATING>	Number (float)
<XDIFF_IMPEDANCE>	Number (float)
<XDIFF_COEF_CJ_R0>	Number (float)
<XDIFF_COEF_CJ_A>	Number (float)
<XDIFF_COEF_CJ_B>	Number (float)
<XDIFF_COEF_CJ_C>	Number (float)
<XDIFF_COEF_TC_A>	Number (float)
<XDIFF_COEF_TC_B>	Number (float)
<XDIFF_COEF_TC_C>	Number (float)

Write:

```
{"SET": "RibXDIFFPortProbe",
"ManufacturerId": <XDIFF_MANUFACTURER_ID>,
"VersionLetter": <XDIFF_VERSION_LETTER>,
"VersionNumber": <XDIFF_VERSION_NUMBER>,
"SerialNumber": <XDIFF_SERIAL_NUMBER>,
"Template": <XDIFF_TEMPLATE>,
"MinTemperature": {"Value": <XDIFF_MINTEMPVALUE>, "Unit": <XDIFF_MINTEMPUNIT>},
"MaxTemperature": {"Value": <XDIFF_MAXTEMPVALUE>, "Unit": <XDIFF_MAXTEMPUNIT>},
"MinElecVal": {"Value": <XDIFF_ELECVAL_MINVALUE>, "Unit": <XDIFF_ELECVAL_MINUNIT>},
"MaxElecVal": {"Value": <XDIFF_ELECVAL_MAXVALUE>, "Unit": <XDIFF_ELECVAL_MAXUNIT>},
"CoefficientsType": <XDIFF_COEF_TYPE>,
"SignalType": <XDIFF_SIGNAL_TYPE>,
"MapMethod": <XDIFF_MAP_METHOD>,
"CalibrationDate": <XDIFF_CALIBRATION_DATE>,
"CalibrationInitials": <XDIFF_CALIBRATION_INITIALS>,
"CalibrationPeriod": <XDIFF_CALIBRATION_PERIOD>,
"SensorMeasID": <XDIFF_MEAS_ID>,
"SensorResponseTime": <XDIFF_RESPONSE_TIME>,
"BuiltInRTDForCJ": <XDIFF_INBUILT_RTD>,
"ColdJunctionSource": <XDIFF_CJ_SOURCE>,
"SensorExcitation": <XDIFF_EXCITATION>,
"SensorMaxExcitation": <XDIFF_MAX_EXCITATION>,
"SensorSelfHeating": <XDIFF_SELF_HEATING>,
"Coefficients": {
   "ColdJunction": {
      "R0": <XDIFF_COEF_CJ_R0>,
      "A": <XDIFF_COEF_CJ_A>,
      "B": <XDIFF_COEF_CJ_B>,
      "C": <XDIFF_COEF_CJ_C>
   },
   "TC": {
      "A": <XDIFF_COEF_TC_A>,
      "B": <XDIFF_COEF_TC_B>,
      "C": <XDIFF_COEF_TC_C>
   }
}
```

Table 48: RibXDIFFPortProbe write parameters

Parameter	Type: value(s)
Consult Table 47: RibXDIFFPortProbe read response parameters	

Response on success:

```
{"SetResponse": "RibXDIFFPortProbe"}
```

DisplayResolution

Writes/reads the number of decimals used to display Set, True, Read and SUT value on the calibrator.

The read returns the resolution for the selected data.
The SENSOR resolution is identical for both channels.

Read:
```
{"GET": "DisplayResolution", "Type": <DISPLAY_TYPE>}
{"GET": "DisplayResolution"}
```

Table 49: DisplayResolution read request parameters

<table>
<tr><th>Parameter</th><th>Type: value(s)</th></tr>
<tr><td><DISPLAY_TYPE></td><td>String:
A models: "SETTEMP" | "READ"
B models: "SETTEMP" | "TRUE" | "READ" | "SENSOR"
C models: "SETTEMP" | "TRUE" | "READ"</td></tr>
</table>

Note! A/B/C models differ!

Response:

Empty (all)

A models:
```
{"GetResponse": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>, "READ":
<READ_DECIMALS>}
```

B models:
```
{"GetResponse": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>, "TRUE":
<TRUE_DECIMALS>, "READ": <READ_DECIMALS>, "SENSOR": <SENSOR_DECIMALS>}
```

C models:
```
{"GetResponse": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>, "TRUE":
<TRUE_DECIMALS>, "READ": <READ_DECIMALS>}
```

SETTEMP
```
{"GetResponse": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>}
```
TRUE (B/C models only!)
```
{"GetResponse": "DisplayResolution", "TRUE": <TRUE_DECIMALS>}
```
READ
```
{"GetResponse": "DisplayResolution", "READ": <READ_DECIMALS>}
```
SENSOR (B models only!)
```
{"GetResponse": "DisplayResolution", "SENSOR": <SENSOR_DECIMALS>}
```

Table 50: DisplayResolution read response parameters

<table>
<tr><th>Parameter</th><th>Type: value(s)</th></tr>
<tr><td><SETTEMP_DECIMALS>
<TRUE_DECIMALS>
<READ_DECIMALS>
<SENSOR_DECIMALS></td><td>Number (integer): 0 | 1 | 2 | 3</td></tr>
</table>

Note! A/B/C models differ!

Write:

All

A models:
```
{"SET": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>, "READ": <READ_DECIMALS>}
```
B models:

```
{"SET": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>, "TRUE": <TRUE_DECIMALS>,
"READ": <READ_DECIMALS>, "SENSOR": <SENSOR_DECIMALS>}
```

C models:

```
{"SET": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>, "TRUE": <TRUE_DECIMALS>,
"READ": <READ_DECIMALS>}
```

SETTEMP

```
{"SET": "DisplayResolution", "SETTEMP": <SETTEMP_DECIMALS>}
```

TRUE (B/C models only!)

```
{"SET": "DisplayResolution", "TRUE": <TRUE_DECIMALS>}
```

READ

```
{"SET": "DisplayResolution", "READ": <READ_DECIMALS>}
```

SENSOR (B models only!)

```
{"SET": "DisplayResolution", "SENSOR": <SENSOR_DECIMALS>}
```

Any combination of SETTEMP, TRUE, READ, and SENSOR may be used (according to current A/B/C model).

Table 51: DisplayResolution write parameters

Parameter	Type: value(s)
<SETTEMP_DECIMALS> <TRUE_DECIMALS> <READ_DECIMALS> <SENSOR_DECIMALS>	Number (integer): 0 | 1 | 2 | 3

Note! A/B/C models differ!

Response:

```
{"SetResponse": "DisplayResolution"}
```

NetworkConfiguration

Managers calibrators network configuration. Write requires DHCP must be false to change remaining parameters.

Read:

```
{"GET": "NetworkConfig",”Interface”,<INTERFACE_VALUE>}
```

Response:

```
{"GetResponse": "NetworkConfig", "Interface": <INTERFACE_VALUE>,
 "Configuration":{
   "DHCP":Boolean,"Address":<IPADRESS>,
   "Netmask":<NETMASK>,
   "Gateway":<GATEWAY>}}
```

Write:

```
{"SET": "NetworkConfig", "Interface": <INTERFACE_VALUE>,
   "Configuration":{
    "DHCP":Boolean,"Address":<IPADRESS>,
    "Netmask":<NETMASK>,
    "Gateway":<GATEWAY>}}
```

Table 552: Network write parameters

Parameter	Type: value(s)
<INTERFACE_VALUE>	"Wired" "Wireless"
<IPADRESS>	Valid Ip address format
<NETMASK>	Valid Netmask for the Ip address
<GATEWAY>	Valid Gateway format for the Ip address.

Response:
```
{"SetResponse": "mConfig"}
```

SilentMode

Configures calibrator’s silent mode.

Read:
```
{"GET": "SilentMode"}
```

Response:
```
{"GetResponse": "SilentMode", "SilentMode": <bool>}
```

Table 58a: OptionsCode write parameters

Write:
```
{"SET": "SilentMode", "SilentMode": <bool>}
```

Table 53a: OptionsCode write parameters

Parameter	Type: value(s)
`<SilentMode>`	bool

Response:
```
{"SetResponse": "SilentMode"}
```

ListWorkOrders

Lists all Work Orders on the calibrator and summary list of their results. Details are omitted.

`Note: If <TEST_TYPE> Empty then default = “Calibration”.`

Read:

```
{
            "GET": "ListWorkOrders",
            "TestType": <TEST_TYPE>
}
```

Parameter	Type: value(s)
`<TEST_TYPE>`	String: Calibration/AutoStep/Switch

Response:
```
{
    "GetResponse": "ListWorkOrders",
    "WorkOrders": [
        {
            "WorkOrder_GUID": <WORKORDER_GUID>,
            "TestType": <TEST_TYPE>,
            "ID": <WORKORDER_ID>,
            "ResultCounter": <RESULT_COUNTER>,
            "Steps": <Steps>,
            "Slope": <Steps>,             "ResultsSummary": [
                {
                    "Result_GUID": <RESULT_GUID>,
                    "ID": <WORKORDER_ID>,
                    "Start": <DATETIME_START>,
                    "End": <DATETIME_END>,
                    "Pass": <PASS_FAIL>,
```

```
                    "AsFound": <ASFOUND_ASLEFT>
                    "Steps": <Steps>,
                    "Deviation": <Deviation>,

                }
            ],
            "SibInputPorts": [
             {
               "ID": <String>,
               "Sensor_ID": <String>,
               "SibPortNo": <PortNo>,
               "SensorType": <SUT_TYPE>
            }
           ],
        }
    ]
}
```

Table 59: ListWorkOrders read response parameters

Parameter	Type: value(s)
<WORKORDER_GUID>	String
<TEST_TYPE>	String: Calibration/AutoStep/Switch
<WORKORDER_ID>	String
<RESULT_COUNTER>	Number (integer)
<RESULT_GUID>	String
<ASFOUND_ASLEFT>	Bool
<PASS_FAIL>	Bool
<DATETIME_START>	String
<Steps>	Integer
<Deviation>	Number (Float)
<PortNo>	Integer

WorkOrder

Only the **WRITE** command is relevant here. Sends a single workorder [WorkOrderInfo] to the calibrator. Almost identical to WorkOrderResult command with following main structural differences.

1. Result details are not included in WorkOrder. { **Fejl! Henvisningskilde ikke fundet.**}
2. List of 2 <SENSOR_UNDER_TEST> Input 1 & 2. (only one in WorkOrderResult)
3. List of 2 <SENSOR_UNDER_TEST> Input 1 & 2. (only one in WorkOrderResult)
4.

Note: Due to complexity, SET syntax is displayed with nested items, indicated with green text color. Essentially, the GET response is built up from a "header" containing the Work Order setup (like a template), including all sensors.

, followed by an overview of the Calibration Results related to this particular Work Order setup. For details on the Calibration Results, use **WorkOrderResults** command.

Read: N/A

Write : (Writes a single workorder [WorkOrder] to the calibrator.)

```
{"SET": "WorkOrder",
   "WorkOrder":
   {
      "TestType": <TEST_TYPE>,
      "GUID": <WORKORDER_GUID>,
      "ID": <WORKORDER_ID>,
```

```
      "CompanyName": <COMPANY_NAME>,
      "Notes": <WO_NOTES>,
      "Sensors": [ { SENSOR_UNDER_TEST <SENSOR_UNDER_TEST> } ],
      "Scenario": { SCENARIO
     <SCENARIO> },
      "Calibrator": { CALIBRATOR
     <CALIBRATOR> },
      "ManualHeatSource": { MANUAL_HEATSOURCE
     <MANUAL_HEATSOURCE> },
      "ManualTrueRead": { MANUAL_TRUEREAD
     <MANUAL_TRUEREAD> },
      "ReferenceSensor": { REFERENCE_SENSOR
     <REFERENCE_SENSOR> },
      "DLCSensor": { REFERENCE_SENSOR
     <REFERENCE_SENSOR> },
      "SibInputPorts":[ { SENSOR_INPUT
     <SENSOR_INPUT> } ],
      "Procedure": { PROCEDURE_INFO
     <PROCEDURE_INFO> },
      "Checksum": { <CHECKSUM> }
     }
}
```

Response:

```
{"SetResponse": " WorkOrder "}
```

Table 6054: WorkOrder read response parameters

Parameter	Type: value(s)
<WORKORDER_GUID>	String
<TEST_TYPE>	String: "Calibration"
<WORKORDER_ID>	String
<COMPANY_NAME>	String
<WORKORDER_NOTES>	String
<CHECKSUM>	Number (integer)

SENSOR_UNDER_TEST

```
<SENSOR_UNDER_TEST> = {
   "ID": <SUT_ID>,
   "SibPortNo": <SENSORINPUT_SIBPORT>,
   "Manufacturer": <SUT_MANUFACTURER>,
   "SerialNumber": <SUT_SERIAL_NUMBER>,
   "TypeDescription": <SUT_DESCRIPTION>,
   "PassFailTolerance": {"Value": <SUT_PASSFAIL_TOLVALUE>, "Unit":
<SUT_PASSFAIL_TOLUNIT>},
   "PassFailTolerancePercent": <SUT_PASSFAIL_TOL_PCT>,
   "TemperatureLimits": {
      "TemperatureMin": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureMax": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
   },
   "TagNo1": <SUT_TAG1>,
   "TagNo1Info": <SUT_TAG1_VAL>,
   "TagNo2": <SUT_TAG2>,
   "TagNo2Info": <SUT_TAG2_VAL>,
   "TagNo3": <SUT_TAG3>,
   "TagNo3Info": <SUT_TAG3_VAL>,
   "CertificateNotes": <SUT_CERTNOTES>,
   "Notes": <SUT_NOTES>
}
```

Table 6155: WorkOrder read response parameters for SENSOR_UNDER_TEST

Parameter	Type: value(s)
<SUT_ID>	String
<SENSORINPUT_SIBPORT>	Number (int)
<SUT_MANUFACTURER>	String
<SUT_DESCRIPTION>	String
<SUT_TYPE>	String: "None" | "Voltage" | "Current" | "RTD" | "TC" | "Switch"
<SUT_PASSFAIL_TOLVALUE>	String Formatted using selected temperature resolution
<SUT_PASSFAIL_TOLUNIT>	String: "KEL" | "CEL" | "FAR"
<SUT_PASSFAIL_TOL_PCT>	Number (float)
<SUT_TAG1>	String
<SUT_TAG1_VAL>	String
<SUT_TAG2>	String
<SUT_TAG2_VAL>	String
<SUT_TAG3>	String
<SUT_TAG3_VAL>	String
<SUT_CERTNOTES>	String
<SUT_NOTES>	String

SCENARIO

```
<SCENARIO> = {
   "ManualHeatingSource": <SCENARIO_MANUAL_SOURCE>,
   "ReferenceSensor": <SCENARIO_REF_SENSOR>,
   "SutSensor1": <SCENARIO_SUT_SENSOR1>,
   "SutSensor2": <SCENARIO_SUT_SENSOR2>
} NOTE! To suppress manual reading for SIB2 defined as NONE,
  set "SutSensor2":="CalibratorInput";
```

Table 6256: WorkOrder read response parameters for SCENARIO

Parameter	Type: value(s)
<SCENARIO_MANUAL_SOURCE>	Bool
<SCENARIO_REF_SENSOR>	String: "Internal"|"External"|"ExternalManual"
<SCENARIO_SUT_SENSOR1> <SCENARIO_SUT_SENSOR2>	String: "CalibratorInput" | "SwitchInput" | "ManualReading" | ManualDevice"

CALIBRATOR

```
<CALIBRATOR> = {
   "ID": <CALIBRATOR_ID>,
   "SerialNumber": <CALIBRATOR_SERIAL_NUMBER>,
   "Manufacturer": <CALIBRATOR_MANUFACTURER>,
   "Model": <CALIBRATOR_MODEL_NAME>,
   "ModelId": <CALIBRATOR_MODEL_ID>,
   "NumSetDecimals": <CALIBRATOR_SET_DECIMALS>,
   "NumReadDecimals": <CALIBRATOR_READ_DECIMALS>,
   "TemperatureLimits" {
      "TemperatureMin": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureMax": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>}
   },
   "CalibrationDate": <CALIBRATOR_CALIBRATION_DATE>,
   "CertificateNotes": <CALIBRATOR_CERTNOTES>,
   "Notes": <CALIBRATOR_NOTES>,
   "Uncertainty": <CALIBRATOR_UNCERTAINTY>
}
```

Table 6357: WorkOrder read response parameters for CALIBRATOR

Parameter	Type: value(s)
<CALIBRATOR_ID>	String
<CALIBRATOR_SERIAL_NUMBER>	String
<CALIBRATOR_MANUFACTURER>	String
<CALIBRATOR_MODEL_ID>	Number (integer)
<CALIBRATOR_MODEL_NAME>	String (e.g. "RTCt-157 B")
<CALIBRATOR_SET_DECIMALS>	Number (integer): 0|1|2|3
<CALIBRATOR_READ_DECIMALS>	Number (integer): 0|1|2|3
<CALIBRATOR_MAXTEMPVALUE> <CALIBRATOR_MINTEMPVALUE>	String Formatted using selected temperature resolution
<CALIBRATOR_MAXTEMPUNIT> <CALIBRATOR_MINTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<CALIBRATOR_CERTNOTES>	String
<CALIBRATOR_NOTES>	String
<CALIBRATOR_UNCERTAINTY>	String
<CALIBRATOR_CALIBRATION_DATE>	String

MANUAL_HEATSOURCE

```
<MANUAL_HEATSOURCE> = {
   "Manufacturer": <MANUALSOURCE_MANUFACTURER>,
   "ID": <MANUALSOURCE_ID>,
   "Serial": <MANUALSOURCE_SERIAL_NUMBER>,
   "Type": <MANUALSOURCE_TYPE>,
   "TemperatureLimits: {
      "TemperatureMin": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureMax": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>}
   },
   "NumDecimals": <MANUALSOURCE_DECIMALS>,
   "CertificateNotes": <MANUALSOURCE_CERTNOTES>,
   "Notes": <MANUALSOURCE_NOTES>,
}
```

Table 6458: WorkOrder read response parameters for MANUAL_HEATSOURCE

Parameter	Type: value(s)
<MANUALSOURCE_MANUFACTURER>	String
<MANUALSOURCE_ID>	String
<MANUALSOURCE_SERIAL_NUMBER>	String
<MANUALSOURCE_TYPE>	String
<MANUALSOURCE_DECIMALS>	Number (integer)
<MANUALSOURCE_CERTNOTES>	String
<MANUALSOURCE_NOTES>	String

MANUAL_TRUEREAD

```
<MANUAL_TRUEREAD> = {
   "Manufacturer": <MANUALTRUE_MANUFACTURER>,
   "ID": <MANUALTRUE_ID>,
   "Serial": <MANUALTRUE_SERIAL_NUMBER>,
   "Type": <MANUALTRUE_TYPE>,
   "TemperatureLimits" {
      "TemperatureMin": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureMax": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>}
   },
   "NumDecimals": <MANUALTRUE_DECIMALS>,
   "ChannelNo": <MANUALTRUE_CHANNELNO>,
   "CertificateNotes": <MANUALTRUE_CERTNOTES>,
   "Notes": <MANUALTRUE_NOTES>
}
```

Table 6559: WorkOrder read response parameters for MANUAL_TRUEREAD

Parameter	Type: value(s)
<MANUALTRUE_MANUFACTURER>	String
<MANUALTRUE_ID>	String
<MANUALTRUE_SERIAL_NUMBER>	String
<MANUALTRUE_TYPE>	String
<MANUALTRUE_DECIMALS>	Number (integer)
<MANUALTRUE_CHANNELNO>	Number (integer)
<MANUALTRUE_CERTNOTES>	String
<MANUALTRUE_NOTES>	String

REFERENCE_SENSOR

```
<REFERENCE_SENSOR> = {
   "ID": <REF_ID>,
   "Manufacturer": <REF_MANUFACTURER>,
   "Serial": <REF_SERIAL_NUMBER>,
   "Description": <REF_DESCRIPTION>,
   "TemperatureLimits": {
      "TemperatureMin": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureMax": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
   },
   "Stability": {
      "Tolerance": {"Value": <REF_STAB_TOLVALUE>, "Unit": <REF_STAB_TOLUNIT>},
      "RequiredSeconds": <REF_STAB_TIME>
   },
   "NumDecimals": <REF_DECIMALS>,
   "ConvertToTemp": <REF_CONVERTTOTEMP>,
   "ReferenceSensorSetup": {
      "ManufacturerId": <XREF_MANUFACTURER_ID>,
      "ModelNumber": <XREF_MODEL_NUMBER>,
      "VersionLetter": <XREF_VERSION_LETTER>,
      "VersionNumber": <XREF_VERSION_NUMBER>,
      "SerialNumber": <XREF_SERIAL_NUMBER>,
      "Template": <XREF_TEMPLATE>,
      "MinTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "MaxTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "MinElecVal": {"Value": <XREF_ELECVAL_MINVALUE>, "Unit":
<XREF_ELECVAL_MINUNIT>},
      "MaxElecVal": {"Value": <XREF_ELECVAL_MAXVALUE>, "Unit":
<XREF_ELECVAL_MAXUNIT>},
      "CoefficientsType": <XREF_COEF_TYPE>,
      "SignalType": <XREF_SIGNAL_TYPE>,
      "MapMethod": <XREF_MAP_METHOD>,
      "CalibrationDate": <XREF_CALIBRATION_DATE>,
      "CalibrationInitials": <XREF_CALIBRATION_INITIALS>,
      "CalibrationPeriod": <XREF_CALIBRATION_PERIOD>,
      "SensorMeasID": <XREF_MEAS_ID>,
      "SensorResponseTime": <XREF_RESPONSE_TIME>,
      "SensorExcitation": <XREF_EXCITATION>,
      "SensorMaxExcitation": <XREF_MAX_EXCITATION>,
      "SensorSelfHeating": <XREF_SELF_HEATING>,
      "Coefficients": <XREF_COEF>
   },
   "CertificateNotes": <REF_CERTNOTES>,
   "Notes": <REF_NOTES>
}
```

The `"CoefficientsType": <XREF_COEF_TYPE>` parameter determines which type of coefficients to use: 0=CVD, 1=ITS-90.

CVD

```
<XREF_COEF> = {"R0": <XREF_COEF_R0>, "A": <XREF_COEF_A>, "B": <XREF_COEF_B>, "C":
<XREF_COEF_C>}
```

ITS-90

```
<XREF_COEF> = {"RTPW": <XREF_COEF_RTPW>, "ALR": <XREF_COEF_ALR>, "BLR":
<XREF_COEF_BLR>, "CLR": <XREF_COEF_CLR>, "C2LR": <XREF_COEF_C2LR>, "C3LR":
<XREF_COEF_C3LR>, "C4LR": <XREF_COEF_C4LR>, "C5LR": <XREF_COEF_C5LR>, "AHR":
<XREF_COEF_AHR>, "BHR": <XREF_COEF_BHR>, "CHR": <XREF_COEF_CHR>, "DHR":
<XREF_COEF_DHR>, "W": <XREF_COEF_W>}
}
```

Table 6660: WorkOrder read response parameters for REFERENCE_SENSOR

Parameter	Type: value(s)
<REF_ID>	String
<REF_MANUFACTURER>	String
<REF_SERIAL_NUMBER>	String
<REF_DESCRIPTION>	String
<REF_DECIMALS>	Number (integer): 0|1|2|3
<REF_CONVERTTOTEMP>	Bool
<REF_STAB_TOLVALUE>	String Formatted using selected temperature resolution
<REF_STAB_TOLUNIT>	String: "KEL" | "CEL" | "FAR"
<REF_STAB_TIME>	Number (integer) Seconds
<XREF_MANUFACTURER_ID>	String
<XREF_MODEL_NUMBER>	Number (integer)
<XREF_VERSION_LETTER>	String
<XREF_VERSION_NUMBER>	Number (integer)
<XREF_SERIAL_NUMBER>	String
<XREF_TEMPLATE>	Number (integer): 1=TC 2=RTD 3=Thermister
<XREF_ELECVAL_MINVALUE> <XREF_ELECVAL_MAXVALUE>	Number (float)
<XREF_ELECVAL_MINUNIT> <XREF_ELECVAL_MAXUNIT>	String
<XREF_COEF_TYPE>	Number (integer): 0 | 1 0: CvD 1: ITS-90
<XREF_SIGNAL_TYPE>	Number (integer): 2
<XREF_MAP_METHOD>	
<XREF_CALIBRATION_DATE>	String
<XREF_CALIBRATION_INITIALS>	String
<XREF_CALIBRATION_PERIOD>	Number (integer) Days
<XREF_MEAS_ID>	Number (integer)
<XREF_RESPONSE_TIME>	Number (float)
<XREF_EXCITATION>	Number (float)
<XREF_MAX_EXCITATION>	Number (float)
<XREF_SELF_HEATING>	Number (float)
<REF_CERTNOTES>	String
<REF_NOTES>	String
<XREF_COEF_RTPW>	Number (float)
<XREF_COEF_ALR>	Number (float)
<XREF_COEF_BLR>	Number (float)
<XREF_COEF_CLR>	Number (float)
<XREF_COEF_C2LR>	Number (float)
<XREF_COEF_C3LR>	Number (float)
<XREF_COEF_C4LR>	Number (float)
<XREF_COEF_C5LR>	Number (float)
<XREF_COEF_AHR>	Number (float)
<XREF_COEF_BHR>	Number (float)
<XREF_COEF_CHR>	Number (float)

<XREF_COEF_DHR>	Number (float)
<XREF_COEF_W>	Number (float)
<XREF_COEF_R0>	Number (float)
<XREF_COEF_A>	Number (float)
<XREF_COEF_B>	Number (float)
<XREF_COEF_C>	Number (float)

XDIFF_SENSOR

```
<XDIFF_SENSOR> = {
   "Enabled": <XDIFF_ENABLED>,
   "SerialNumber": <XDIFF_SERIAL_NUMBER>,
   "ReferenceModel": <XDIFF_MODEL>,
   "TemperatureLimits": {
      "MinTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "MaxTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
   },
   "Stability": {
      "Tolerance": {"Value": <XDIFF_STAB_TOLVALUE>, "Unit": <XDIFF_STAB_TOLUNIT>}
   },
   },
   "ReferenceSensorSetup": {
      "ManufacturerId": <XDIFF_MANUFACTURER_ID>,
      "ModelNumber": <XDIFF_MODEL_NUMBER>,
      "VersionLetter": <XDIFF_VERSION_LETTER>,
      "VersionNumber": <XDIFF_VERSION_NUMBER>,
      "SerialNumber": <XDIFF_SERIAL_NUMBER>,
      "Template": <XDIFF_TEMPLATE>,
      "MinTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "MaxTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "MinElecVal": {"Value": <XDIFF_ELECVAL_MINVALUE>, "Unit":
<XDIFF_ELECVAL_MINUNIT>},
      "MaxElecVal": {"Value": <XDIFF_ELECVAL_MAXVALUE>, "Unit":
<XDIFF_ELECVAL_MAXUNIT>},
      "CoefficientsType": <XDIFF_COEF_TYPE>,
      "SignalType": <XDIFF_SIGNAL_TYPE>,
      "MapMethod": <XDIFF_MAP_METHOD>,
      "CalibrationDate": <XDIFF_CALIBRATION_DATE>,
      "CalibrationInitials": <XDIFF_CALIBRATION_INITIALS>,
      "CalibrationPeriod": <XDIFF_CALIBRATION_PERIOD>,
      "SensorMeasID": <XDIFF_MEAS_ID>,
      "SensorResponseTime": <XDIFF_RESPONSE_TIME>,
      "BuiltInRTDForCJ": <XDIFF_INBUILT_RTD>,
      "ColdJunctionSource": <XDIFF_CJ_SOURCE>,
      "SensorExcitation": <XDIFF_EXCITATION>,
      "SensorMaxExcitation": <XDIFF_MAX_EXCITATION>,
      "SensorSelfHeating": <XDIFF_SELF_HEATING>,
      "SensorImpedance": <XDIFF_IMPEDANCE>,
      "Coefficients": {
         "ColdJunction": {
            "R0": <XDIFF_COEF_CJ_R0>,
            "A": <XDIFF_COEF_CJ_A>,
            "B": <XDIFF_COEF_CJ_B>,
            "C": <XDIFF_COEF_CJ_C>
         },
         "TC": {
            "A": <XDIFF_COEF_TC_A>,
            "B": <XDIFF_COEF_TC_B>,
            "C": <XDIFF_COEF_TC_C>
         }
      }
   }
   "NumDecimals": <XDIFF_DECIMALS>,
   "ConvertToTemp": <XDIFF_CONVERTTOTEMP>,
}
```

Table 6761: WorkOrder read response parameters for XDIFF_SENSOR

Parameter	Type: value(s)
<XDIFF_ENABLED>	Bool
<XDIFF_MODEL>	String
<XDIFF_SERIAL_NUMBER>	String
<XDIFF_DECIMALS>	Number (integer): 0|1|2|3
<XDIFF_CONVERTTOTEMP>	Bool
<XDIFF_STAB_TOLVALUE>	String Formatted using selected temperature resolution
<XDIFF_STAB_TOLUNIT>	String: "KEL" | "CEL" | "FAR"
<XDIFF_CALIBRATION_DATE>	String
<XDIFF_MANUFACTURER_ID>	Number (integer)
<XDIFF_VERSION_NUMBER>	Number (integer)
<XDIFF_VERSION_LETTER>	String (1 character)
<XDIFF_SERIAL_NUMBER>	Number (integer)
<XDIFF_TEMPLATE>	Number (integer) =0 (TC)
<XDIFF_MINTEMPVALUE> <XDIFF_MAXTEMPVALUE>	String **Formatted using selected temperature resolution**
<XDIFF_MAXTEMPUNIT> <XDIFF_MINTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<XDIFF_ELECVAL_MAXVALUE> <XDIFF_ELECVAL_MINVALUE>	Number (float)
<XDIFF_ELECVAL_MAXUNIT> <XDIFF_ELECVAL_MINUNIT>	String Thermcouple/DLC “Voltage”. RTD “Ohm”
<XDIFF_COEF_TYPE>	Number (integer): 0..14 0 B 1 E 2 J 3 K 4 N 5 R 6 S 7 T 8 Non-standard 9 L 10 U 11 XK 12 BP 13 C 14 CJC-only
<XDIFF_SIGNAL_TYPE>	Number (integer): VoltageSensor = 0, CurrentSensor = 1, ResistanceSensor = 2, BridgeSensor = 3, LVDTSensor = 4, PotentiometricVoltageDividerSensor = 5, PulseSensor = 6, VoltageActuator = 7, CurrentActuator = 8, PulseActuator = 9

<XDIFF_MAP_METHOD>	Number (integer):
	Linear = 0,
	Inverse_m_xb = 1, // m/(x+b)
	Inverse_bm_x = 2, // (b+m/x)
	Inverse_x_bm = 3, // 1/(b+m/x)
	Thermocouple = 4,
	Thermistor = 5,
	RTD = 6,
	Bridge = 7

<XDIFF_CALIBRATION_DATE>	String: "YYYY-MM-DD" YYYY = 4 digit year MM = 2 digit month (January = 01, December = 12) DD = 2 digit day of month

	Example: May 3rd, 2021: "2021-05-03"
<XDIFF_CALIBRATION_INITIALS>	String
<XDIFF_CALIBRATION_PERIOD>	Number (integer) Calibration period in days
<XDIFF_MEAS_ID>	Number (integer)
<XDIFF_RESPONSE_TIME>	Number (float)
<XDIFF_INBUILT_RTD>	Bool: true|false
<XDIFF_CJ_SOURCE>	Number (integer): 0 | 1 | 2 0 = "No CJ required" 1 = "Built in RTD" (Sensor compensated for 0°C CJ) 2 = "CJ Required"
<XDIFF_EXCITATION>	Number (float)
<XDIFF_EXTICATION_MAX>	Number (float)
<XDIFF_SELF_HEATING>	Number (float)
<XDIFF_IMPEDANCE>	Number (float)
<XDIFF_COEF_CJ_R0>	Number (float)
<XDIFF_COEF_CJ_A>	Number (float)
<XDIFF_COEF_CJ_B>	Number (float)
<XDIFF_COEF_CJ_C>	Number (float)
<XDIFF_COEF_TC_A>	Number (float)
<XDIFF_COEF_TC_B>	Number (float)
<XDIFF_COEF_TC_C>	Number (float)

SENSOR_INPUT

```
<SENSOR_INPUT> = {
   "ID": <SENSORINPUT_DEVICEID>,
   "Manufacturer": <SENSORINPUT_MANUFACTURER>,
   "SerialNumber": <SENSORINPUT_SERIAL_NUMBER>,
   "NumDecimals": <SENSORINPUT_DECIMALS>,
   "ConvertToTemp": <SENSORINPUT_CONVERTTOTEMP>,
   "Stability": {
      "Tolerance": {"Value": <SENSORINPUT_STAB_TOLVALUE>, "Unit":
<SENSORINPUT_STAB_TOLUNIT>},
      "RequiredSeconds": <SENSORINPUT_STAB_TIME>,
      "Enabled": <SENSORINPUT_STAB_ENABLED>
   }
   "SensorType": <SENSORINPUT_SIBPORT_TYPE>,
   "SibPortNo": <SENSORINPUT_SIBPORT>,
   "VoltageOption": <SENSORINPUT_VOLTAGE_TYPE>,
   "CurrentOption": <SENSORINPUT_CURRENT_TYPE>,
   "RTDOption": <SENSORINPUT_RTD_TYPE>,
   "TCOption": <SENSORINPUT_TC_TYPE>,
   "SwitchOption": {},
   "CertificateNotes": <SENSORINPUT_CERTNOTES>,
   "Notes": <SENSORINPUT_NOTES>
}
```

The <SENSORINPUT_SIBPORT_TYPE> parameter determines which of the following <SENSORINPUT_xxx_TYPE> objects to include.

SENSORINPUT_VOLTAGE_TYPE

```
<SENSORINPUT_VOLTAGE_TYPE> = {
      "TemperatureLow": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureHigh": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "VoltageLow": {"Value": <VOLTAGE_LOWVALUE>, "Unit": <VOLTAGE_LOWUNIT>},
      "VoltageHigh": {"Value": <VOLTAGE_HIGHVALUE>, "Unit": <VOLTAGE_HIGHUNIT>}
}
```

SENSORINPUT_CURRENT_TYPE

```
<SENSORINPUT_CURRENT_TYPE> = {
```

```
      "TemperatureLow": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "TemperatureHigh": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "CurrentLow": {"Value": <CURRENT_LOWVALUE>, "Unit": <CURRENT_LOWUNIT>},
      "CurrentHigh": {"Value": <CURRENT_HIGHVALUE>, "Unit": <CURRENT_HIGHUNIT>},
      "LoopPower": <LOOP_POWER>
   },
}
```

SENSORINPUT_RTD_TYPE

```
<SENSORINPUT_RTD_TYPE> = {
      "Wires": <WIRES>,
      "SensorType": <RTD_TYPE>
}
```

SENSORINPUT_TC_TYPE

```
<SENSORINPUT_TC_TYPE> = {
      "CompensationMode": <COMPENSATION_MODE>,
      "ManualTemp": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
      "SensorType": <TC_TYPE>
}
```

Table 6862: WorkOrder read response parameters for SENSOR_INPUT

Parameter	Type: value(s)
<SENSOR_ID>	String
<SENSOR_MANUFACTURER>	String
<SENSOR_SERIAL_NUMBER>	String
<SENSOR_DESCRIPTION>	String
<SENSOR_DECIMALS>	Number (integer): 0|1|2|3
<SENSOR_CONVERTTOTEMP>	Bool
<SENSOR_UNCERTAINTY>	?
<SENSOR_INPUTTEXT>	String User input
<SENSOR_SIBPORT>	Number (integer): 1|2
<SENSOR_SIBPORT_TYPE>	String: "None" | "Voltage" | "Current" | "RTD" | "TC" | "Switch"
<SENSOR_CERTNOTES>	String
<SENSOR_NOTES>	String
<SENSOR_STAB_TOLVALUE>	String Formatted using selected temperature resolution
<SENSOR_STAB_TOLUNIT>	String: "KEL" | "CEL" | "FAR"
<SENSOR_STAB_TIME>	Number (integer) Seconds
<SENSOR_STAB_ENABLED>	Bool
<VOLTAGE_HIGHVALUE> <VOLTAGE_LOWVALLUE>	Number (float) Voltages are in V (volt)
<VOLTAGE_HIGHUNIT> <VOLTAGE_LOWUNIT>	String "volt"
<CURRENT_HIGHVALUE> <CURRENT_LOWVALUE>	Number (float) Currents are in mA (milliampere).
<CURRENT_HIGHUNIT> <CURRENT_LOWUNIT>	String "mA"
<RTD_TYPE>	String: "P10(90)385" | … RTD type table.
<TC_TYPE>	String: "B" | "E" | "J" | "K" | "L" | "N" | "R" | "S" | "T" | "U"
<WIRES>	Number (integer): 2|3|4
<COMPENSATION_MODE>	String: "Auto" | "Manual"
<LOOP_POWER>	Bool

PROCEDURE_INFO

```
<PROCEDURE_INFO> = {
   "StepList": [
      {
         "SetTemperature": {
            "Value": <PROCEDURE_STEP_SETTEMPVALUE>,
            "Unit": <PROCEDURE_STEP_SETTEMPUNIT>
         },
         "Slope": {
            "Value": <PROCEDURE_STEP_SLOPEVALUE>,
            "Unit": <PROCEDURE_STEP_SLOPEUNIT>
         },
         "HeatSource": <PROCEDURE_STEP_HEATSOURCE>
      }
   ],
   "TwoWay": <PROCEDURE_TWOWAY>,
   "HoldTime": <PROCEDURE_HOLDTIME>,
   "KeepLast": <PROCEDURE_KEEPLAST>,
   "CoolingTemperature": {
      "Value": <PROCEDURE_COOLINGTEMPVALUE>,
      "Unit": <PROCEDURE_COOLINGTEMPUNIT>
   },
   "UseSetFollowsTrue": <PROCEDURE_USESFT>,
   "UseDLCCorrection": <PROCEDURE_USEDLC>
   }
}
```

Table 6963: WorkOrder read response parameters for PROCEDURE_INFO

Parameter	Type: value(s)
<PROCEDURE_STEP_SETTEMPVALUE>	String Formatted using selected temperature resolution
<PROCEDURE_STEP_SETTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<PROCEDURE_STEP_SLOPEVALUE>	String Formatted using selected temperature resolution
<PROCEDURE_STEP_SLOPEUNIT>	String: "KEL" | "CEL" | "FAR"
<PROCEDURE_STEP_HEATSOURCE>	String:
<PROCEDURE_TWOWAY>	Bool: true|false
<PROCEDURE_HOLDTIME>	Number (integer)Mintues
<PROCEDURE_KEEPLAST>	Bool: true|false
<PROCEDURE_COOLINGTEMPVALUE>	String Formatted using selected temperature resolution
<PROCEDURE_COOLINGTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<PROCEDURE_USESFT>	Bool: true|false
<PROCEDURE_USEDLC>	Bool: true|false
<PROCEDURE_DLCSTABVALUE>	String Formatted using DLC resolution
<PROCEDURE_SLOPERATEVALUE>	String Formatted using selected temperature resolution
<PROCEDURE_SLOPERATEUNIT>	String: "KEL" | "CEL" | "FAR"

Write : (Writes a single workorder [WorkOrderInfo] to the calibrator.)

```
{
   "SET": "WorkOrder",
   "WorkOrder":
   {
      "TestType": <TEST_TYPE>,
      "GUID": <WORKORDER_GUID>,
      "ID": <WORKORDER_ID>,
```

```
      "CompanyName": <COMPANY_NAME>,
      "Notes": <WORKORDER_NOTES>,
    "Sensors": [{SENSOR_UNDER_TEST <SENSOR_UNDER_TEST>}],
    "Scenario": { SCENARIO
  <SCENARIO> },
    "Calibrator": { CALIBRATOR
  <CALIBRATOR> },
    "ManualHeatSource": { MANUAL_HEATSOURCE
  <MANUAL_HEATSOURCE> },
    "ManualTrueRead": { MANUAL_TRUEREAD
  <MANUAL_TRUEREAD> },
    "ReferenceSensor": { REFERENCE_SENSOR
  <REFERENCE_SENSOR> },
    "DLCSensor": { REFERENCE_SENSOR
  <REFERENCE_SENSOR> },
    "SibInputPorts": [{ SENSOR_INPUT
  <SENSOR_INPUT> }],
    "Procedure": { PROCEDURE_INFO
  <PROCEDURE_INFO> },
      "Checksum": { <CHECKSUM> }
  }
}
```

Consult previous tables for details on parameters and nested objects. Note: CalibrationResults are omitted for the Write method.

Response:

```
{"SetResponse": "WorkOrder"}
```

DeleteWorkOrder

Deletes a single workorder from the calibrator's list of current jobs.

Write:

```
{"SET": "DeleteWorkOrder", "GUID": <WORKORDER_GUID>, "TestType": <TEST_TYPE>}
```

Table 7064: WorkOrderResults read parameters

Parameter	Type: value(s)
<WORKORDER_GUID>	String
<TEST_TYPE>	String : Calibration/AutoStep/Switch

Response:

```
{"SetResponse": "DeleteWorkOrder"}
```

WorkOrderResult

Returns calibration results for a particular Work Order on the calibrator.

Note:

1. Due to complexity, the response syntax is displayed with nested items, indicated with green text color. Consult details from the 'read' method above.

The Calibration Results contain a complete description of the Work Order setup ,the particular sensor as well as the particular Work Order test result.

Read:

```
{
            "GET": "WorkOrderResult",
            "WorkOrder_GUID": <WORKORDER_GUID>,
            "TestType": <TEST_TYPE>,
            "Result_GUID": <RESULT_GUID>
}
```

Table 7165: WorkOrderResults read parameters

Parameter	Type: value(s)
<WORKORDER_GUID>	String
<TEST_TYPE>	String : Calibration/AutoStep/Switch
<RESULT_GUID>	String

Response:

```
{
  "GetResponse": "WorkOrderResult",
  "WorkOrderResult":
       {
          "TestType": <TEST_TYPE>,
          "WorkOrder_GUID": <WORKORDER_GUID>,
          "Result_GUID": <RESULT_GUID>,
          "ID": <WORKORDER_ID>,
          "Company": <COMPANY_NAME>,
          “Notes”: <NOTES>,
          "Calibrator": { CALIBRATOR
  <CALIBRATOR> },
          "ReferenceSensor": { REFERENCE_SENSOR
  <REFERENCE_SENSOR> },
          "DLCSensor": { REFERENCE_SENSOR
  <REFERENCE_SENSOR> },
          "SensorUnderTest": { <SENSOR_UNDER_TEST<SENSOR_UNDER_TEST> },
          "Scenario": { SCENARIO
  <SCENARIO> },
          "ManualHeatSource": { MANUAL_HEATSOURCE
  <MANUAL_HEATSOURCE> },
          "ManualTrueRead": { MANUAL_TRUEREAD
  <MANUAL_TRUEREAD> },
          "Procedure": { PROCEDURE_INFO
  <PROCEDURE_INFO> },
          "SensorInput": { SENSOR_INPUT
  <SENSOR_INPUT> }
          "ManualTemperatureDuring": <MANUAL_TEMP_DURING>,
          "ManualCJTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
          "AmbientTemperature": {"Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
          "Start": <DATETIME_START>,
          "End": <DATETIME_END>,
          "AsFound": <ASFOUND_ASLEFT>,
          "StepResultList": { Fejl! Henvisningskilde ikke fundet.},

     }
}
```

```
<CALIBRATION_STEP_RESULTS> =
{
  "StepNumber": <STEP_NUMBER>,
  "RefSensorTemperature": {
    "InputType": <REF_INPUTTYPE>,
    "InputValue": <REF_VALUE>
    "TemperatureValue": { "Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
  },
```

```
  "SUTTemperature": {
    "InputType": <SUT_INPUTTYPE>,
    "InputValue": <SUT_VALUE>
    "TemperatureValue": { "Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
  },
  "SUTManualTemperature": {
    "InputType": <SUT_MANUAL_INPUTTYPE>,
    "InputValue": <SUT_MANUAL_VALUE>
    "TemperatureValue": { "Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
  },
  "SETTemperature": { "Value": <TEMPVALUE>, "Unit": <TEMPUNIT>},
  "Pass": <STEP_PASS>,
  "TimeStamp": <STEP_TIMESTAMP>,
  "StepSkipped": <STEP_SKIPPED>
}
```

Table 7266: WorkOrderResults read response parameters

Parameter	Type: value(s)
<RESULT_GUID>	String
<PROCEDURE_ID>	String
<WORKORDER_ID>	String
<COMPANY_NAME>	String
<WORKORDER_NOTES>	String
<WORKORDER_CHECKSUM>	Number (integer)
<NOTES>	String
<MANUAL_TEMP_DURING>	Bool
<MANUAL_CJTEMPVALUE>	String Formatted using selected temperature resolution
<MANUAL_CJTEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<AMBIENT_TEMPVALUE>	String Formatted using selected temperature resolution
<AMBIENT_TEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<ASFOUND_ASLEFT>	Bool
<DATETIME_START>	String
<DATETIME_END>	String
<REF_INPUTTYPE>	String "INT_RTD", "REF_RTD" | "INT_RTD" (depending on True selection.)
<TEMPVALUE>	String Formatted using selected temperature resolution
<TEMPUNIT>	String: "KEL" | "CEL" | "FAR"
<REF_VALUE>	Number (float)
<SUT_MANUAL_INPUTTYPE>	String
<SUT_MANUAL_VALUE>	Number (Float)
<SUT_INPUTTYPE>	String: "SENS_None" | "SENS_Voltage" | "SENS_Current" | "SENS_Ohm400" | "SENS_Ohm4000" | "SENS_TC" | "SENS_Switch"
<SUT_VALUE>	Number (Float)
<STEP_NUMBER>	Number (integer)
<STEP_PASS>	Bool: true|false
<STEP_TIMESTAMP>	String
<STEP_SKIPPED>	Bool
<RESULT_ASFOUND_ASLEFT>	Bool
<RESULT_MANUAL_TEMP_DURING>	Bool: true|false
<RESULT_REF_SERIAL_NUMBER>	String

Table 7367: WorkOrder read response parameters for CALIBRATION_RESULTS_LIST

Parameter	Type: value(s)
<RESULT_COUNTER>	Number (integer)
<RESULT_UNIQUE_ID>	String
<WORKORDER_ID>	String
<DATETIME_START>	String
<DATETIME_END>	String

DeleteWorkOrderResult

Deletes results of a workorder from the calibrator's list of current jobs.

Write:

```
{
            "SET": "DeleteWorkOrderResult",
            "WorkOrder_GUID": <WORKORDER_GUID>,
            "TestType": <TEST_TYPE>,
            "Result_GUID": <RESULT_GUID>
}
```

Response:

```
{"SetResponse": "DeleteWorkOrderResult"}
```

DeleteAllWorkOrderResults

Deletes all the results for a particular workorder from the list of current jobs.

Write:

```
{"SET": "DeleteAllWorkOrderResults", "GUID": <WORKORDER_GUID>, "TestType":
<TEST_TYPE>}
```

Response:

```
{"SetResponse": "DeleteAllWorkOrderResults"}
```

RTD type table.

'P10(90)385'
'P50(90)385'
'P100(90)385'
'P200(90)385'
'P400_90_385'
'P500_90_385'
'P1000(90)385'
'P50(90)391'
'P100(90)392'
'M50(90)428'
'M100(90)428'
'H120(90)672'
'Pt-100MILL'
'YSI-400'

SENSORS, TEST & CALIBRATION

www.ametekcalibration.com

UK
Tel +44 (0)1243 833 302
stc.uk@ametek.com

France
Tel +33 (0)1 30 68 89 40
general.lloyd-instruments@ametek.fr

Germany
Tel +49 (0)2159 9136 510
info.mct-de@ametek.de

Denmark
Tel +45 4816 8000
jofra@ametek.com

USA - **California**
Tel +1 (800) 444 1850
crystal@ametek.com

USA - **Florida**
Tel +1 (800) 527 9999
cal.info@ametek.com

India
Tel +91 22 6196 8200
jofra@ametek.com

Singapore
Tel +65 6484 2388
jofra@ametek.com

China - Beijing
Tel +86 10 8526 2111 ext. 19
stc.china@ametek.com