Arkeo Multichannel API

v0.4

21-07-2025

Contents

Introduction

This API facilitates remote control of the Arkeo Multichannel using TCP on port 6340, enabling users to perform a variety of commands manage measurements.

A command must sent as a JSON string in the form
`{'command': command, 'parameter': parameter}`
Each command must be preceded by a 4-byte integer indicating the total length of the command. After each command, a reply is always sent back. Its format depends on the command (see list below). If an unknown command is sent, the string "`Not a valid command`" is send instead.

Note: Currently only a single client is allowed to connect to the multichannel software.

Summary of available commands

Command	Description
SetActiveChannel	Set the channel upon which all subsequent actions are performed
GetActiveChannel	Get the active channel
SetChannelSettings	Modify the settings
GetChannelSettings	Retrieve the settings
StartChannel	Start measurement
StopChannel	Stop the measurement process
ForceJV	Force a JV measurement
GetChannelState	Retrieve the current state
GetLatestJV	Get the voltage and current of last performed JV
GetIV	Get the live voltage and current of all channels
GetSensors	Get the live sensor values of all connected sensors

List of commands

Command: SetActiveChannel

Description: Set the channel upon which all succeeding actions are performed.
Parameters:

- channel_id (integer): The ID of the channel.

Response:

- channel_id (integer): The ID of the channel.

Command: GetActiveChannel

Description: Get the active channel.
Parameters:

Response:

- channel_id (integer): The ID of the channel.

Command: SetChannelSettings

Description: Set the JSON configuration string for active channel.
Parameters:

- settings (JSON string): The configuration settings in JSON format (see chapter: The JSON settings string).

Response:

- status (string): OK

Command: GetChannelSettings

Description: Retrieve the JSON configuration string of active channel.
Parameters:

Response:

- settings (JSON string): The configuration settings in JSON format (see chapter: The JSON settings string).

Command: StartChannel

Description: Start the measurement process for active channel.
Parameters:

Response:

- status (string): OK

Command: StopChannel

Description: Stop the measurement process for active channel.
Parameters:

Response:

- status (string): OK

Command: ForceJV

Description: Force a JV measurement on active channel if channel is in tracking mode.
Parameters:

- channel_id (integer): The ID of the active channel.

Response:

- status (string): OK

Command: GetChannelState

Description: Retrieve the current state of a specific channel (e.g., running, stopped, JV, tracking).
Parameters:

Response:

- state (string): JSON of the state of the active channel (see chapter: The JSON state string)

Command: GetLatestJV

Description: Retrieve the voltage and current values of the latest JV.
Parameters:

Response:

- JV Data (string):
  v_fw1|j_fw 1|...|v_fw n|j_fw n|| v_rv1|j_rv 1|...|v_rv n|j_rv n

Note:

Forward and Reverse scans are separated by || (double pipe character)
Voltage and Current arrays are interleaved and separated by | (single pipe character)
Voltage unit: V

Current unit A/cm^2

Command: GetIV

*Description***:** Retrieve the live voltage and current values of all channels.
*Parameters***:**

*Response***:**

- Voltage and Current values (string):
  v1|j1|...|vn|jn

Note:

Voltage and Current arrays are interleaved and separated by | (single pipe character)
Voltage unit: V
Current unit A/cm^2

Command: GetSensors

*Description***:** Retrieve the live voltags of all sensors.
*Parameters***:**

*Response***:**

- Sensor values (string):
  v1|...|vn|

Note:

Sensor voltages are separated by | (single pipe character)
Voltage unit: V

The JSON settings string

The settings of a channel are represented in a JSON string. When using the **Command: SetChannelSettings**, this is value that the software expects. It is recommended to use the **Command: GetChannelSettings** to read the actual settings and modify those settings.

Example

Below is an example of a channel settings string. A detailed description can be found on the next page.


```
{
  "Index":"1A",
  "Enable":true,
  "User":"Cicci Research",
  "Device":"Si cell",
  "Channel":{
    "VoltageLimit":"10 V",
    "CurrentLimit":0,
    "InvertedStructure":false
  },
  "JV":{
    "Vmin (V)":-0.1,
    "Vmax (V)":1.2,
    "Step (mV)":20,
    "ScanRate (mV/s)":100,
    "VocDetect":true,
    "Overvoltage (%)":0,
    "ScanOrder":"FW then RV"
  },
  "Tracking":{
    "TrackEnable":true,
    "Algorithm":"MPPT",
    "Perturbation (V)":0.02,
    "ConstantOutput":0,
    "SaveInterval (s)":10,
    "jvInterval":{"Value":10,"Unit":"min"},
    "TestDuration":{"Value":100,"Unit":"hours"}
  },
  "Cell":{
    "Type":"Cell",
    "Area (cm2)":1,
    "NrCells":1,
    "NrW cells":1,
    "W-cellArea (cm2)":1
  },
  "Note":""
}
```

Description

PARAMETER	DESCRIPTION	EXAMPLE	UNIT	TYPE
Index	Unique channel identifier	1A		string
Enable	Enables or disables the channel	true		boolean
User	Name of the user	User		string
Device	Device Name	Sample		string
Channel				object
- VoltageLimit[1]	Maximum Voltage	10 V		enum
- CurrentLimit	Current Range	0		integer
- InvertedStructure	Inverts the applied voltage	false		boolean
JV				object
- Vmin (V)	Minimum voltage	-0.1	V	float
- Vmax (V)	Maximum voltage	2	V	float
- Step	Voltage increment per step	20	mV	integer
- ScanRate (mV/s)	Rate at which the voltage is applied	100	mV/s	float
- VocDetect	Enables the detection of open-circuit voltage	true		boolean
- ScanOrder[2]	Order of scanning	*FW then RV*		enum
Tracking				object
- TrackEnable	Enables tracking	true		boolean
- Algorithm[3]	Specifies the algorithm used for tracking	MPPT		enum
- Perturbation (V)	Voltage differential for the tracking algorithm	0.01	V	float
- ConstantOutput	Setting to maintain a constant output	0.2	*	float
- SaveInterval (s)	Time between saved data poitns	10	s	integer
- jvInterval.Value	Time between JV scans	10		float
- jvInterval.Unit[4]	Unit for JV interval	min		enum
- TestDuration.Value	Duration of the tracking test	100		float
- TestDuration.Unit[4]	Unit for Test Duration	1		enum
Cell				object
- Type[5]	Cell Type	Cell		enum
- Area (cm2)	Area of the cell	1	cm^2	float
- NrCells	Number of cells	1		integer
- NrW cells	Number of W cells	1		integer
- W-cellArea (cm2)	The area of each W cell	1	cm^2	float
Note	Free to use field for comments			string

Notes for settings JSON

Detailed explanation and enums of the settings JSON

1. Voltage Limit

The voltage limit of each board is 10 V. 20 V can be reached by connecting 2 boards in parallel. Connect the positive contacts of 2 boards to your device and the negative contacts to each other.

2. Scan Order enum

0: FW then RV
1: RV then FW
2: Forward Only
3: Reverse Only

3. Tracking Algorithm enum

0: Open circuit
1: Short circuit
2: MPPT
3: MPPT-Stab
4: MPPT INC
5: Fixed Voltage
6: Fixed Voltage (no track)
7: Fixed Current
8: JV

4. Time unit enum

0: seconds
1: minutes
2: hours

5. Cell Type enum

0: Cell
1: Parallel Module
2: Z Module
3: W Module

The JSON state string

The state of a channel is represented in a JSON string. This string is read-only and is obtained using the **Command: GetChannelState**.

Example

Below is an example of a state string. A detailed description can be found below.

```
{"Enable":false,
  "Channel":"1A",
  "User":"User",
  "Measurement":"JV",
  "Direction":"Forward",
  "State":"Running",
}
```

Description

PARAMETER	DESCRIPTION	EXAMPLE	DATA TYPE
Enable	Enables or disables the channel	true	Boolean
Channel	ID of the channel	1A	String
User	Name of the user	User	String
Measurement	State of the Measurement	JV	String
Direction	Current direction of the JV	Forward	String
State	Current state of the channel – Idle – Ready to start – Running – Paused – Stopped – Error	Running	String

Examples

Python

```python
import socket
import json

def exampleCommand(command, data=''):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
        sock.connect(('localhost', 6340))

        message = json.dumps({'command': command, 'data': data})

        # Prepare the length of the string
        length = len(message)
        length_bytes = length.to_bytes(4, byteorder='big')  # 4 bytes to
represent the length

        sock.sendall(length_bytes) # Send the length of the string
        sock.sendall(message.encode('utf-8')) # Send the string

        # Receive response
        response = sock.recv(1024)
        print('Received:', response.decode('utf-8'))

exampleCommand('GetChannelSettings')
```

Changelog

v0.1
Initial version

v0.2
Added ForceJV and GetChannelState

v0.3
Added GetLatestJV, GetIV and GetSensors

v0.4
Reorganized the settings JSON and clarified the documentation
Added JSON schema validation