Optional Capabilities for Vendor Device Certification

The device certification process requires the device to follow integration best practices. This means that more fields are mandatory for a certified device compared to platform minium requirements. In the following section, optional capabilities and behavior are described. The optional capabilities require the Mandatory Capabilites for Vendor Device Certification

- ☑ Foundation Capabilities ☑ c8y_Agent
  - ☑ Device Information
    - ☑ c8y_IsDevice
    - ☑ name
    - ☑ type
    - ☑ c8y_Hardware
    - ☑ c8y_Firmware
    - ☑ c8y_RequiredAvailability
    - ☑ c8y_SupportedOperations
  - ☑ External ID
  - ☑ Sending Operational Data
    - ☑ Measurements
    - ☑ Events
    - ☑ Alarms
- ☐ Optional Capabilities
  - ☑ Gateways
    - ☑ Child Device Types
  - ☑ Log File Retrieval
  - ☑ Device Configuration
    - ☑ Text Based Configuration
    - ☑ File Based Configuration
  - ☑ Managing Device Software
  - ☑ Managing Device Firmware
  - ☑ Managing Device Firmware
  - ☑ Device Profile
  - ☑ Restart
  - ☐ Measurement Request
  - ☐ Shell
  - ☐ Cloud Remote Access
  - ☑ Location & Tracking

Optional Capabilities

All sections below are **optional**. If a device partner decides to certify Optional Capabilities, they are documented in the certificate and shown on the device partner portal. Customer can filter and search for devices that support certain capabilities. Therefore, it is recommended to certify all capabilities (aka. "Optional Capabilities") offered by the device. The capabilities are listed below in descending order of importance based

on Software AG's experience. To indicate that a device wants to certify an Optional Capabilities, it has to add the respective element to the list of supported operations in the inventory object of the device. All Optional Capabilities that receive operations require the fragment `com_cumulocity_model_Agent` to be present in the device managed object in the inventory as described in Foundation Capability Fragments.

Fragment	Content	Required for optional capability
`c8y_RequiredAvailability`	Minimal communication interval to determine if device is offline	No
`c8y_SupportedOperations`	Many optional operations are directly triggering the dynamic UI by invoking the respective operation tabs	Yes, if the Optional Capability is an operation
`com_cumulocity_model_Agent`	Empty fragment. Declares that the device is able to receive operations	Yes, for root devices and gateways that support operations; No, for devices and gateways that don't support operations; Must not be used for child devices;

Example structure of the device Managed Object in the inventory:

```
"c8y_IsDevice": {},
"com_cumulocity_model_Agent": {},
"name": "edge-agent-eabdcf5d344b4a52a4ffab13fe3d11cb",
"type": "c8y_EdgeAgent",
"c8y_Agent": {
    "name": "DeviceAgent",
    "version": "1.0",
    "url": ""
},
"c8y_Firmware": {
    "name": "raspberrypi-bootloader",
    "version": "1.20140107-1",
    "url": "31aab9856861b1a587e2094690c2f6e272712cb1"
},
"c8y_Hardware": {
    "model": "BCM2708",
    "revision": "000e",
    "serialNumber": "00000000e2f5ad4d"
},
"c8y_RequiredAvailability": {
    "responseInterval": 6
},
"c8y_SupportedOperations": [
    "c8y_Software",
    "c8y_Firmware",
    "c8y_LogfileRequest",
    "c8y_Configuration",
```

```
        "c8y_SendConfiguration"
    ],
    "c8y_Position": {
            "lng": 8.6449,
            "lat": 49.819199
    },
    "c8y_SoftwareList": [{
        "name": "adduser",
        "version": "3.118",
        "url": ""
    }],
    "c8y_Configuration": {
        "config": "mqtt.url=mqtt.eu-
latest.cumulocity.com\nmqtt.port=8883\nmqtt.tls=true\nmqtt.cert_auth=false\nmqtt.c
lient_cert=/root/.cumulocity/certs/chain-
cert.pem\nmqtt.client_key=/root/.cumulocity/certs/device-cert-private-
key.pem\nmqtt.cacert=/etc/ssl/certs/ca-
certificates.crt\nmqtt.ping.interval.seconds=60\nagent.name=dm-example-
device\nagent.type=c8y_dm_example_device\nagent.main.loop.interval.seconds=10\nage
nt.requiredinterval=10\nagent.loglevel=INFO"
    },
    "c8y_SupportedLogs": [
            "agentlog"
    ]
```

c8y_RequiredAvailability

Minimal communication interval to determine if device is offline. For details and examples, compare device availability section of the documentation.

Fragment	Mandatory
responseInterval	No

Example structure in device inventory:

```
"c8y_RequiredAvailability": {
    "responseInterval": 6
}
```

c8y_SupportedOperations

The fragment c8y_SupportedOperations is used to identify which operations (and hence certifiable Optional Capabilities) are supported by the device. This fragment is optional. If not present, no Optional Capabilities will be certified.

Fragment	Mandatory
c8y_SupportedOperations	No

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_Software",
    "c8y_Firmware"
]
```

Gateways

For details and examples, compare child operations section of the documentation.

Child Device Types

Cumulocity uses the concept of *child device types* to distinguish the capabilities of child devices *behind* a gateway device. For example, a child device connected via a simple analog wire connection (like a temperature sensor) may only be able to send measurements (the temperature), while a child device connected via OPC-UA is able to send measurements, events, alarms, and log files in addition to the ability to process incoming requests to upgrade its firmware. In this case, the child device type `Analog` is only supporting the minimum requirements for certification without any *optional capabilities*. The child device type `OPC-UA` is supporting the *foundation capabilities* as well at the optional capabilities `Logs` and `Firmware`.

Child device types can be freely named, however, here are some examples as orientation:.

Fragment	Content	Required for optional capability
`c8y_SupportedChildDeviceTypes`	List contains supported child device types	Yes

Example structure in device inventory:

```
"c8y_SupportedChildDeviceTypes": [
    "Analog",
    "Canbus",
    "CanOpen",
    "Modbus",
    "OPCUA",
    "Profibus",
    "Sigfox",
    "SNMP"
]
```

Log File Retrieval

Device capability to upload (filtered) log files to C8Y. For details and examples, compare chapter `c8y_LogfileRequest` of section miscellaneous in the documentation.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
com_cumulocity_model_Agent	Enables a device to receive operations; Must be present in the device manged object in the inventory;	Yes
c8y_SupportedOperations	List contains element c8y_LogfileRequest	Yes
c8y_SupportedLogs	List of supported log file types; Must be present in the device manged object in the inventory;	Yes (at least 1 type)

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_LogfileRequest"
],
"c8y_SupportedLogs": [
    "syslog",
    "dmesg"
]
```

Example operation sent to the device:

```
"c8y_LogfileRequest": {
    "logFile": "syslog",
    "dateFrom": "2016-01-27T13:45:24+0100",
    "dateTo": "2016-01-28T13:45:24+0100",
    "searchText": "sms",
    "maximumLines": 1000
}
```

When the device receives the operation c8y_LogfileRequest, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with "status" : "PENDING"	Real-time notifications
1.	Update operation "status" : "EXECUTING"	Update operation
2.	Internally retrieve log file and filter w.r.t. criteria found in operation	
3.	Create an event with "type": "c8y_LogfileRequest"	Create event

Step	Action	Documentation
4.	Upload the log file as attachment to the event	Attach file to event
5.	Update operation accordingly `"status": "SUCCESSFUL",` `"c8y_LogfileRequest": {"file":` `"https://<TENANT_DOMAIN>/event/events/{id}/binaries"`	Update operation

Device Configuration

Device capability that enables text- and / or profile-based device configuration. Text based configuration is the more basic approach. It provides a plain text box in the UI to retrieve, edit, and send a configuration text to the device. The text is send as one string, however, it can be structured using json, xml, key-value pairs or any other markup that the device is able to parse. File based configuration allows to have multiple *types* of configurations (e.g. one file for defining polling intervals and another to configure the internal log-levels). For details and examples, compare configuration management section in the documentation.

For a successful certification of the Device Configuration capability, Text Based Configuration or File Based Configuration or both have to be implemented. The certificate will state which configuration methods is supported as information.

Text Based Configuration

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`com_cumulocity_model_Agent`	Must be present in the inventory; Enables a device to receive operations	Yes
`c8y_SupportedOperations`	List contains element `c8y_Configuration`	Yes
`c8y_SupportedOperations`	List contains element `c8y_SendConfiguration`	Yes

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_Configuration",
    "c8y_SendConfiguration"
],
"c8y_SendConfiguration": {}
```

Example operation `c8y_Configuration: {}` as it is sent to the device:

```
    "creationTime": "2021-09-20T13:10:25.933Z",
    "deviceId": "181119",
    "self": "https://t635974191.eu-
latest.cumulocity.com/devicecontrol/operations/440452",
    "id": "440452",
    "status": "PENDING",
    "c8y_Configuration": {
        "config": "test"
    },
    "description": "Configuration update"
```

When the device receives the operation `c8y_Configuration`, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation
2.	Internally interpret transmitted string and execute configuration	
3.	Update operation accordingly `"status": "SUCCESSFUL"`	Update operation

Example operation `c8y_SendConfiguration: {}` as it is sent to the device:

```
creationTime: "2021-09-20T13:53:29.419Z", deviceName: "123456789", deviceId:
"440366",…
c8y_SendConfiguration: {}
creationTime: "2021-09-20T13:53:29.419Z"
description: "Requested current configuration"
deviceId: "440366"
deviceName: "123456789"
id: "440472"
self: "https://t635974191.eu-
latest.cumulocity.com/devicecontrol/operations/440472"
status: "PENDING"
```

When the device receives the operation `c8y_SendConfiguration`, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation
2.	Internally get current configuration and update the fragment `c8y_Configuration` of the device inventory object	Update managed object

Step	Action	Documentation
3.	Update operation accordingly `"status": "SUCCESSFUL"`	Update operation

File Based Configuration

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`com_cumulocity_model_Agent`	Must be present in the inventory; Enables a device to receive operations	Yes
`c8y_SupportedOperations`	List contains element `c8y_DownloadConfigFile`	Yes
`c8y_SupportedOperations`	List contains element `c8y_UploadConfigFile`	Yes
`c8y_SupportedConfigurations`	List of supported configuration file types	Yes (at least 1 type)

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_DownloadConfigFile",
    "c8y_UploadConfigFile"
    ],
"c8y_SupportedConfigurations":[
    "config1",
    "config2"
    ]
```

Example operation sent to the device for `c8y_DownloadConfigFile`:

```
"c8y_DownloadConfigFile": {
    "url": "<download url>"
}
```

When the device receives the operation `c8y_DownloadConfigFile`, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation

Step	Action	Documentation
2.	Download referenced binary and internally apply configuration	
3.	Update operation `"status": "SUCCESSFUL"`	[Update operation]()

Example operation sent to the device for `c8y_UploadConfigFile`:

```
"c8y_UploadConfigFile":{}
```

When the device receives the operation `c8y_UploadConfigFile`, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	[Real-time notifications]()
1.	Update operation `"status" : "EXECUTING"`	[Update operation]()
2.	Internally retrieve the configuration type requested in operation e.g. `"c8y_UploadConfigFile": {"type": "someConfig"}`	
3.	Create an event with the same `type` e.g. `"type": "someConfig"`	[Create event]()
4.	Upload the configuration as attachment to the event	[Attach file to event]()
5.	Update operation accordingly `"status": "SUCCESSFUL"`	[Update operation]()

Managing Device Software

Device capability to manage and deploy software packages to the device. For details and examples, compare [c8y_SoftwareList]() section in the documentation .

Note: *Firmware Management* and *Software Management* are handled separately in Cumulocity IoT and follow different concepts. A device can support one ore both capabilities.

Firmware

- The firmware is the base operating system of a device
- A device can only have 1 firmware installed at a time
- When the firmware is changed, the device usually flashes itself
- Usually used by small / embedded devices

Software

- Software is an optional component executed on the device
- A device can have multiple software installed at a time
- Software can be installed or removed independently of other software and the firmware

- Usually used by larger / more powerfully devices

Info: `"c8y_SupportedOperations": ["c8y_SoftwareList"]` (not to be confused with the *inventory fragment* `c8y_SoftwareList`) and `"c8y_SupportedOperations": ["c8y_Software"]` are deprecated and should not be used for new agent implementations.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`com_cumulocity_model_Agent`	Must be present in the inventory; Enables a device to receive operations	Yes
`c8y_SupportedOperations`	List contains element `c8y_SoftwareUpdate`	Yes
`c8y_SoftwareList`	List of currently installed software on the device	Yes

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_SoftwareUpdate"
],
"c8y_SoftwareList": [
    {
        "name": "Software A",
        "version": "1.0.1",
        "url": "www.some-external-url.com"
    },
    {
        "name": "Software B",
        "version": "2.1.0",
        "url": "mytenant.cumulocity.com/inventory/binaries/12345"
    }
]
```

Example operation sent to the device:

```
"c8y_SoftwareUpdate": [
    {
        "name": "mySoftware1",
        "version": "1.0.0",
        "url": "http://www.example.com",
        "action": "install"
    },
    {
        "name": "mySoftware2",
        "version": "1.1.0",
```

```
            "url": "http://www.example.com",
            "action": "update"
        },
            {
            "name": "mySoftware2",
            "version": "1.1.0",
            "url": "http://www.example.com",
            "action": "uninstall"
        }
    ]
```

When the device receives the operation `c8y_SoftwareUpdate`, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation
2.	Execute the given `"action"` (install, update, uninstall) specified by the operation `c8y_SoftwareUpdate`	
3.	Update `c8y_SoftwareList` fragment in the inventory object of the device	Update managed object
4.	Update operation accordingly `"status": "SUCCESSFUL"`	Update Operation

Managing Device Firmware

Device capability that enables firmware management. For details and examples, compare `device information` section in the documentation.

Note: *Firmware Management* and *Software Management* are handled separately in Cumulocity IoT and follow different concepts. A device can support one ore both capabilities.

Firmware

- The firmware is the base operating system of a device
- A device can only have 1 firmware installed at a time
- When the firmware is changed, the device usually flashes itself
- Usually used by small / embedded devices

Software

- Software is an optional component executed on the device
- A device can have multiple software installed ata time
- Software can be installed or removed independently of other software and the firmware
- Usually used by larger / more powerfully devices

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
com_cumulocity_model_Agent	Must be present in the inventory; Enables a device to receive operations	Yes
c8y_SupportedOperations	List contains element c8y_Firmware	Yes

Firmware tab will be visible on the device page only if c8y_Firmware is listed in the device's supported operations.

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_Firmware"
]

"c8y_Firmware": {
    "name": "raspberrypi-bootloader",
    "version": "1.20140107-1",
    "url": "31aab9856861b1a587e2094690c2f6e272712cb1"
}
```

Example operation sent to the device:

```
"c8y_Firmware": {
    "name": "firmware_a",
    "version": "2.0.24.3",
    "dependency": "2.0.23.0",
    "url": " ",
    "isPatch": true
}
```

When the device receives the operation having c8y_Firmware, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with "status" : "PENDING"	Real-time notifications
1.	Update operation "status" : "EXECUTING"	Update operation

Step	Action	Documentation
2.	Compare name and version stored in the fragment `c8y_Firmware` in the inventory object of the device with the name and version stored in the received operation fragment `c8y_Firmware`. If the name and/or version are differing download from the (device specific) firmware repository referenced in the url and install.	Device information
3.	Update the fragment `c8y_Firmware` of the inventory object of the device.	
4.	Update operation accordingly `"status": "SUCCESSFUL"`	Update operation

Device Profile

Device capability to manage device profiles. Device profiles represent a combination of a firmware version, one or multiple software packages and one or multiple configuration files which can be deployed on a device. Based on device profiles, users can deploy a specific target configuration on devices by using bulk operations. For details and examples, compare `Managing device data` section in the documentation.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`com_cumulocity_model_Agent`	Must be present in the inventory; Enables a device to receive operations	Yes
`c8y_SupportedOperations`	List contains element `c8y_DeviceProfile`	Yes
`c8y_Profile`	List contains element `profileName`, `profileId`, and `profileExecuted`	Yes

Device profile tab will be visible on the device page only if `c8y_DeviceProfile` is listed in the device's supported operations.

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_DeviceProfile"
]
"c8y_Profile": {
    "profileName": "Device_Profile",
    "profileId": "60238",
    "profileExecuted": true
}
```

Example operation sent to the device:

```
"c8y_DeviceProfile": {
    "software": [
        {
            "name": "asd",
            "action": "install",
            "version": "1.0",
            "url": "aURL"
        }
    ],
    "configuration": [],
    "firmware": {
        "name": "aName",
        "version": "aVersion",
        "url": "aURL"
    },
    "description": "Assign device profile Device_Profile to device Device_Name",
    "deviceId": "437604",
    "profileId": "60238",
    "profileName": "Device_Profile"
}
```

When the device receives operation `c8y_DeviceProfile` it will execute the following steps:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation
2.	Use the information stored in the operation `c8y_DeviceProfile` regarding software, firmware and configuration to execute changes as described in respective sections.	
3.	Update the fragment `c8y_Profile` of the inventory object of the device by adding the nested fragments `"profileName":"Device_Profile"`, `"profileId": "60238"` and `"profileExecuted": true/false`.	
3.	Update operation accordingly `"status": "SUCCESSFUL"`	Update operation

Restart

Device capability to restart the device. For details and examples, compare miscellaneous section of the documentation.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`com_cumulocity_model_Agent`	Must be present in the inventory; Enables a device to receive operations	Yes
`c8y_SupportedOperations`	List contains element `c8y_Restart`	Yes

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_Restart"
]
```

When the device receives the operation `c8y_Restart` the following steps are executed:

Step	Action	Documentation
0.	Listen for operations created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `c8y_Restart` to `"status" : "EXECUTING"`	Update operation
2.	Restart the device	
3.	Query all operations in `"status" : "EXECUTING"` to continue processing them	Real-time notifications
4.	Query operations by agent ID and `"status" : "EXECUTING"` to clean up them. This includes the update of `c8y_Restart` to `"status" : "SUCCESSFUL"`	Device Integration, Update operation
5.	Listen to new operations created in Cumulocity IoT	Real-time notifications

Measurement Request

Device capability to send an updated set of measurements on user request. This can be usefully for devices, that send measurements infrequency.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`com_cumulocity_model_Agent`	Must be present in the inventory; Enables a device to receive operations	Yes

Fragment	Content	Required for optional capability
c8y_SupportedOperations	List contains element c8y_MeasurementRequestOperation	Yes

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_MeasurementRequestOperation"
]
```

When the device receives the operation c8y_MeasurementRequestOperation: {"requestName": "LOG"} the following steps are executed:

Note: For legacy reasons, the c8y_MeasurementRequestOperation operation contains the fragment "requestName": "LOG". This is to be ignored by the device. The device vendor can decide if all or a useful subset of measurements are send as response to this operation.

Step	Action	Documentation
0.	Listen for operation created by platform with "status" : "PENDING"	Real-time notifications
1.	Update operation "status" : "EXECUTING"	Update operation
2.	Send all or a useful subset of measurements	Create measurement
3.	Update operation "status": "SUCCESSFUL"	Update operation

Shell

Device capability to send any command to the device. The feature is often used to send shell commands to the device and receive the output as result. For details and examples, compare miscellaneous section of the documentation.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
com_cumulocity_model_Agent	Must be present in the inventory; Enables a device to receive operations	Yes
c8y_SupportedOperations	List contains element c8y_Command	Yes

Example structure in device inventory:

```
  "c8y_SupportedOperations": [
      "c8y_Command"
  ]
```

Example operation sent to the device for `c8y_Command`:

```
  "c8y_Command": {
      "text": "get uboot.sn"
  }
```

When the device receives the operation `c8y_Command`, the following steps are executed:

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation
2.	Locally execute the command and add the result to the operation in the fragment `result`	Update operation
3.	Update operation `"status": "SUCCESSFUL"`	Update operation

Example operation after it has been executed and fragment `result` has been added to `c8y_Command`:

```
  "c8y_Command": {
      "text": "get uboot.sn",
      "result": "123456"
  }
```

Cloud Remote Access

Document Section is still under development

Device capability to initiate a remote connection via VNC or SSH. For details and examples, compare Cloud Remote Access section of the documentation.

Note: The connection type `Telnet` is considered insecure, therefore it is deprecated and should not be used for new implementations. Please use VNC or SSH instead.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability

Fragment	Content	Required for optional capability
com_cumulocity_model_Agent	Must be present in the inventory; Enables a device to receive operations	Yes
c8y_SupportedOperations	List contains element c8y_RemoteAccessConnect	Yes
c8y_RemoteAccessList	List of supported remote access types	Yes (at least 1 type)

Example structure in device inventory:

```
"c8y_SupportedOperations": [
    "c8y_RemoteAccessConnect"
]
"c8y_RemoteAccessList": [
    {
        "Id": 3234,
        "name": "My Connection",
        "hostname": "10.0.0.67",
        "port": 5900,
        "protocol": "VNC",
        "credentials": {
            "username": "someUser",
            "password":  "{cipher
        }SwRUVOR0gKHmPUuKLulIIM3EMGvxFTg22had6b",
    "privateKey": null,
        "publicKey": null,
        "hostKey": null,
        "type": "PASS_ONLY"
    }
}
]
```

Example operation sent to the device for c8y_RemoteAccessConnect:

```
"c8y_RemoteAccessConnect": {
    "hostname": "10.0.0.67", // Endpoint on local network to connect to
    "port": 5900, // Port to be used on local network endpoint
    "connectionKey": "eb5e9d13-1caa-486b-bdda-130ca0d87df8" // Shared secret to
authenticate the connection request from device side
}
```

When the device receives the operation c8y_RemoteAccessConnect, the following steps are executed:

Step	Action	Documentation

Step	Action	Documentation
0.	Listen for operation created by platform with `"status" : "PENDING"`	Real-time notifications
1.	Update operation `"status" : "EXECUTING"`	Update operation
2.	Connect to provided `hostname` and `port` using TCP	
3.	Using the `connectionKey` connect to device websocket endpoint of the Remote Access microservice: `wss:///service/remoteaccess/device/`	
4.	Update operation `"status": "SUCCESSFUL"`	Update operation
5.	Start forwarding binary packets between the TCP connection and the websocket in both directions	
6.	Whenever one of these connections is terminated the device considers the session as ended and will also terminate the second connection	

Location & Tracking

Device capability to display and update location information. For details and examples, compare location capabilities section of the documentation.

The following fragments are related to the optional device capability with a remark if they are required for the capability to work:

Fragment	Content	Required for optional capability
`c8y_Position`	Position information of the device	Yes
`c8y_Position.lat`	Latitude	Yes
`c8y_Position.lng`	Longitude	Yes
`c8y_Position.alt`	Altitude in meters	No
`c8y_Position.trackingProtocol`	Technology used for position acquisition (e.g. GPS, Galileo, TELIC)	No
`c8y_Position.reportReason`	Reason why the position update was send (e.g. triggered by schedule, action)	No

Example structure in device inventory:

```
"c8y_Position": {
    "lat": 51.211977,
    "lng": 6.15173,
```

```
      "alt": 67,
  }
```

Whenever the location shell be updated, the device executes the following steps:

Step	Action	Documentation
1.	Send a position update event	Create event
2.	Update the c8y_Position fragment in device inventory	Update managed object

Example location update event:

```
  {
    "source": {
      "id": 4036, // device ID for which you want to update the position
    },
    "type": "c8y_LocationUpdate", // must be c8y_LocationUpdate
    "text": "Location updated", // field mandatory but content can be changed
    "time": "2021-09-07T14:04:27.845Z'",
    "c8y_Position": {
      "lat": 51.211977,
      "lng": 6.15173,
      "alt": 67,
    },
  }
```

##MD file change Log

Date	Chapter	Severity
30/09/2021	Added MD file change log	minor
01/11/2021	shell: Example added	minor
03/11/2021	com_cumulocity_model_agent added as mandatory for each optional agent capability that relies on receiving operations; Moved supported child device types to optional Capabilities;	major
08/11/2021	Updated broken links	minor
09/11/2021	Updated broken links, added Currently Supported Device Capabilities of Self-Service Certification Microservice	minor
15/11/2021	Changed structure	minor