OCPP: Zephyr RTOS as Electric Vehicle Supply Equipment (EVSE)

Saravanan Sekar

Linumiz

Linumiz

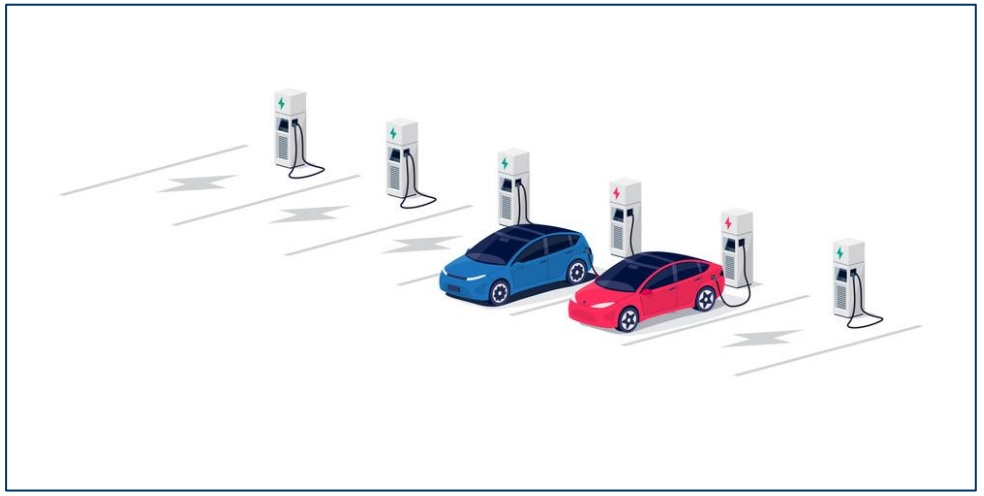
whoami

- Saravanan Sekar, Linumiz
  - Embedded Linux and Zephyr RTOS development, consulting, training
  - Embedded Linux development: BSP, u-boot, Linux Kernel, Yocto Project, Buildroot
  - Zephyr: SoC, Board support, drivers
  - www.linumiz.com
- Living in **Berlin**, Germany

Agenda

- Electric Vehicle (EV) and charging infrastructure
- How Zephyr RTOS helps ?
- OCPP specification internals
- OCPP stack in Zephyr
- EVSE standards (IEC 61851 & ISO 15118)
- OCPP future work in Zephyr

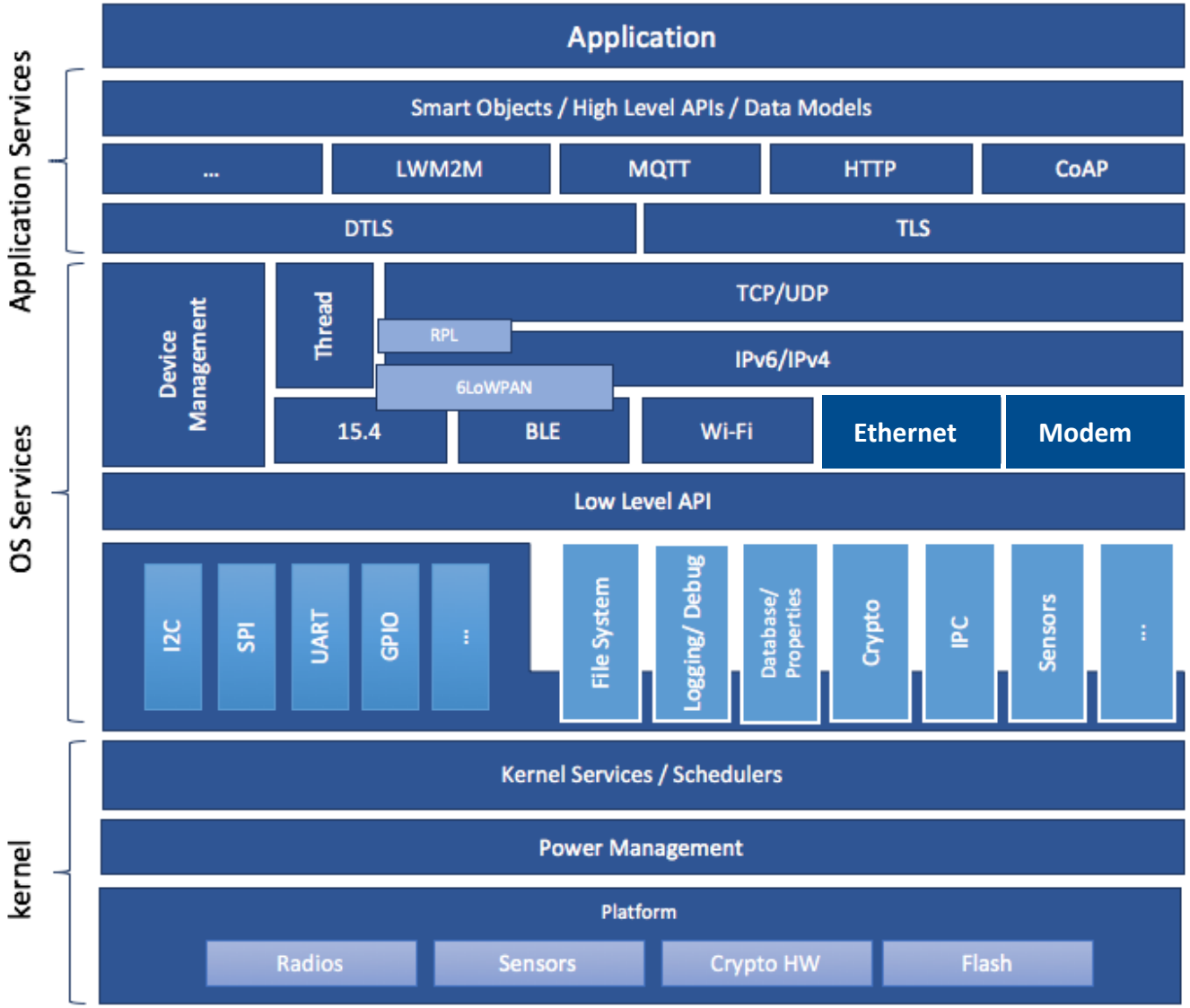
Electric Vehicle (EV) and charging infrastructure



- Electric vehicle supply equipment (EVSE) supplies electricity to an electric vehicle (EV), commonly called charging stations or charging docks
- Lack of charging infrastructure
- EV charging duration is high and occupies a huge parking space
- EVSE is portable and easy to install in most parking space
- More distributed lightweight charging station is needed

How Zephyr RTOS helps ?

- Microcontrollers are more suitable for lightweight applications, which in turn aid in distributed charging infrastructure
- Flexible choice of multiple network interface Ethernet/Wi-Fi/Modem
- Wide range of platform selection
- Industrial energy meter reads data from interfaces like Modbus



Ref: https://docs.zephyrproject.org/latest/_images/security-zephyr-system-architecture.png

Linumiz

OCPP: Introduction

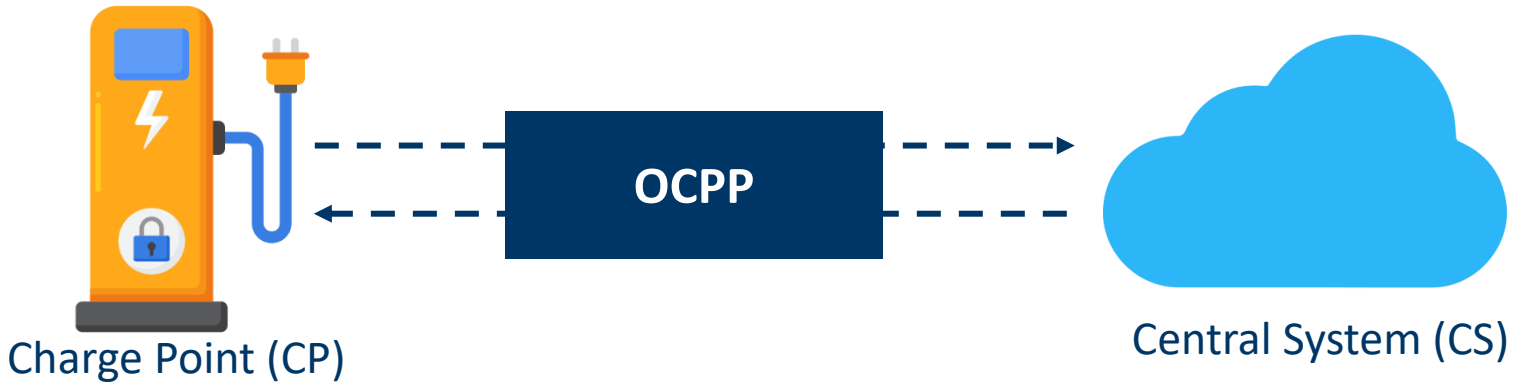
- The Open Charge Point Protocol (OCPP) is an application protocol for communication between Electric Vehicle Supply Equipment (EVSE charging stations) and a central management system
- Established in 2009 by the Open Charge Alliance, OCPP is designed to be a free open-source
- The global benchmark for interoperability throughout the EV charging industry
- OCPP supported charging stations allow easy integration with public infrastructure, and a private charging station
- OCPP versions v1.5, v1.6 and v2.0.1

Abbreviation

Abbreviation	Definition
CP	Charge Point
CPO	Charge Point Operator
CS	Central System
IdTag	RFID Unique identifier
OCPP	Open Charge Point Protocol
PDU	Protocol Defined Unit
RPC	Remote Procedure Call

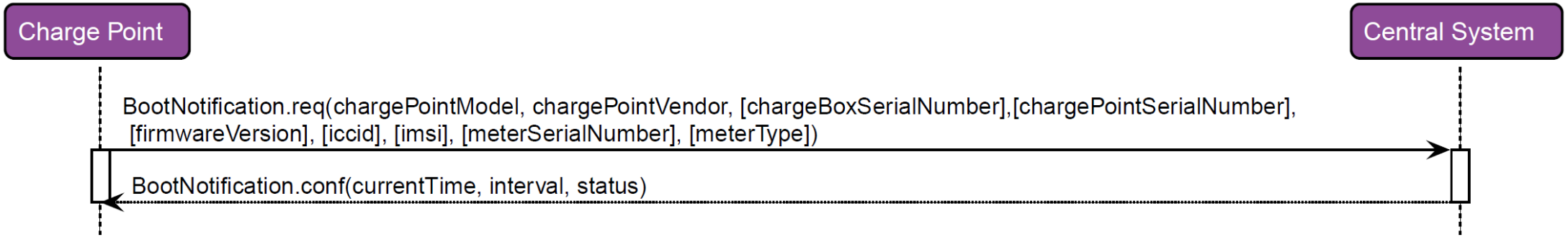
OCPP: Profiles & messages

- The operations between the Central System (CS) and Charge Point (CP) are based on a predefined set of PDU (Protocol Defined Unit) messages
- Messages are grouped as profiles based on features & functionality
  - Core
  - Local auth list management
  - Reservation
  - Smart charging
  - Remote Trigger
  - Firmware Management



OCPP: Boot Notification

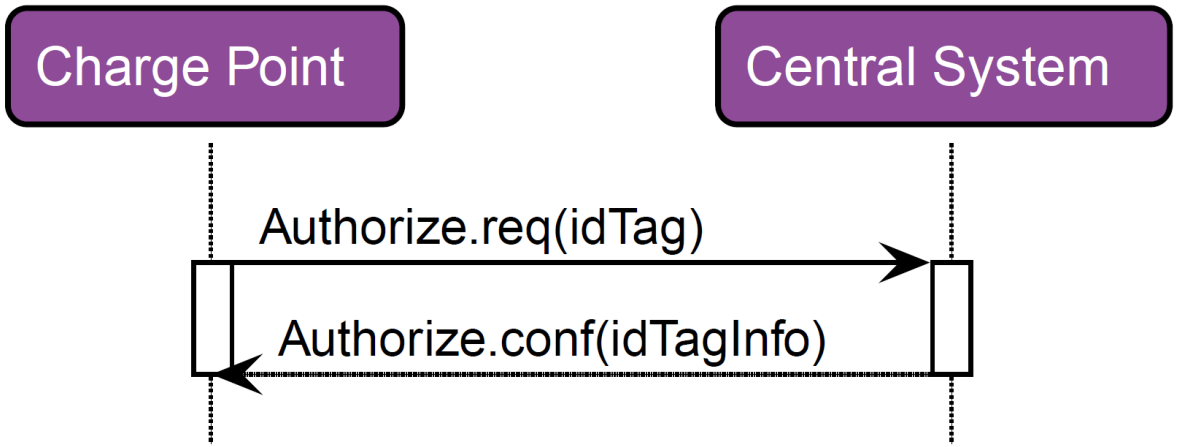
- After start-up or reboot, a Charge Point (CP) initiates communication with the Central Station (CS) by sending a boot notification Protocol Defined Unit (PDU)
- No other PDU should be sent before the boot notification



Linumiz

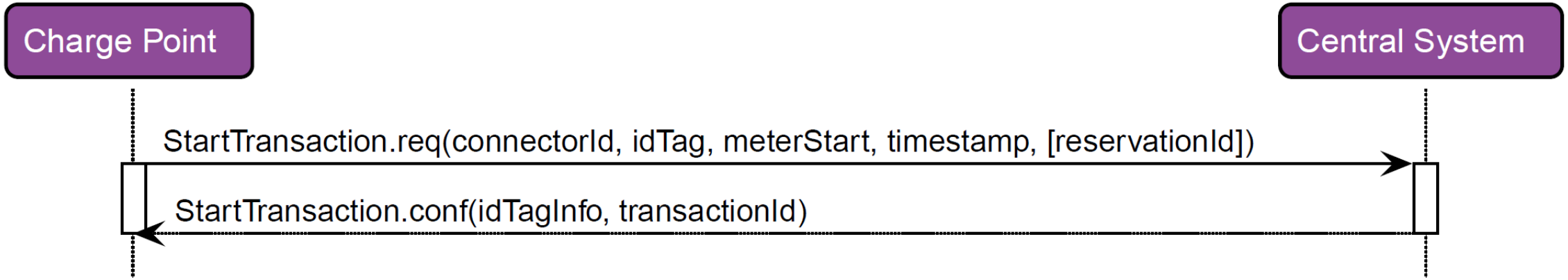
OCPP: Authorize

- The user of a Charge Point (CP) must be authorized before start or stop charging
- A unique identifier (idTag) used for authorization



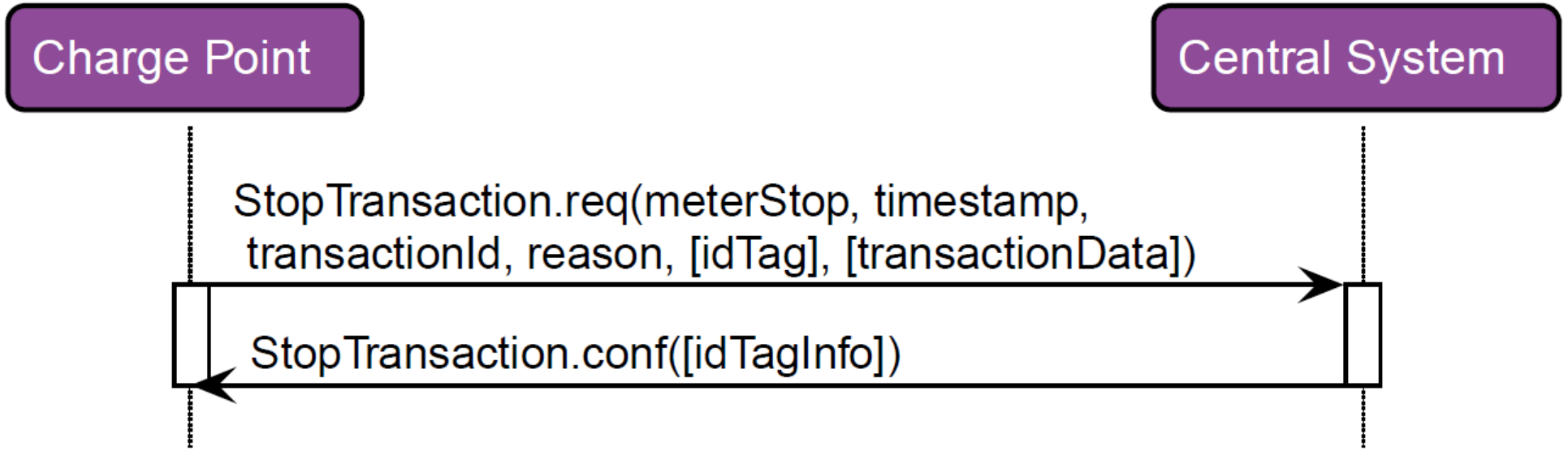
OCPP: Start Transaction

- The Charge Point (CP) sends a *StartTransaction* Protocol Defined Unit (PDU) to the Central System (CS) to notify about a transaction that has been started



OCPP: Stop Transaction

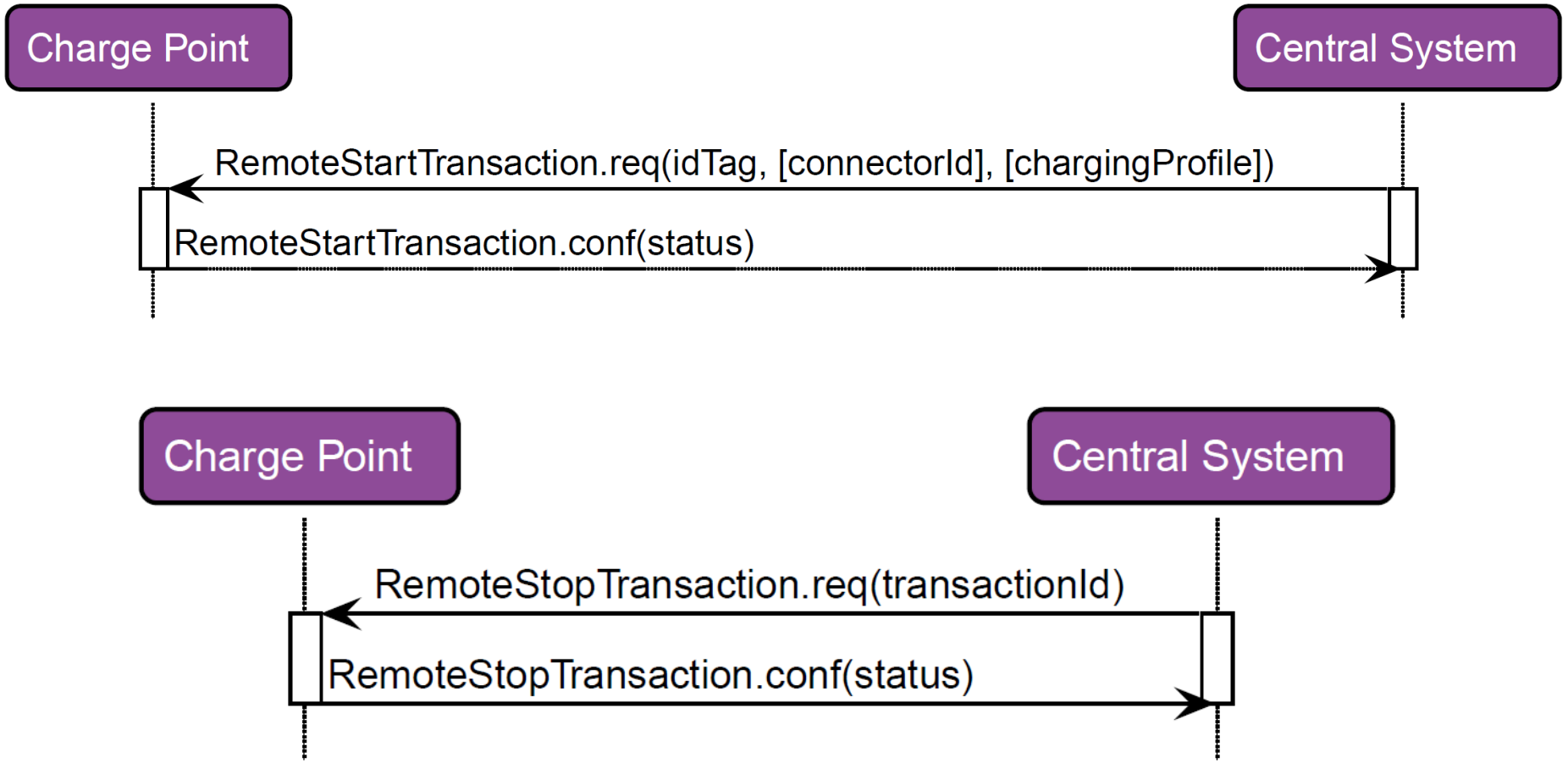
- The Charge Point (CP) sends a *StopTransaction* Protocol Defined Unit (PDU) to the Central System (CS) to notify about a transaction that has stopped



Linumiz

OCPP: Remote start/stop Transaction

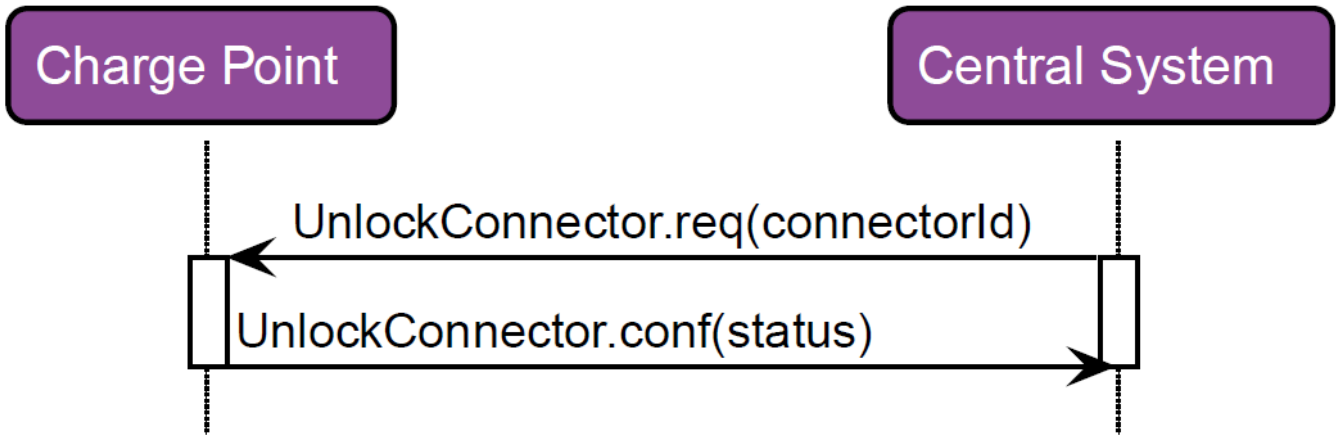
- A Central System (CS) can request a Charge Point (CP) to start/stop charging using *RemoteStartTransaction* / *RemoteStopTranscation* Protocol Defined Unit (PDU) respectively. Upon CP receipt of the PDU, it decides to start/stop



Linumiz

OCPP: Unlock connector

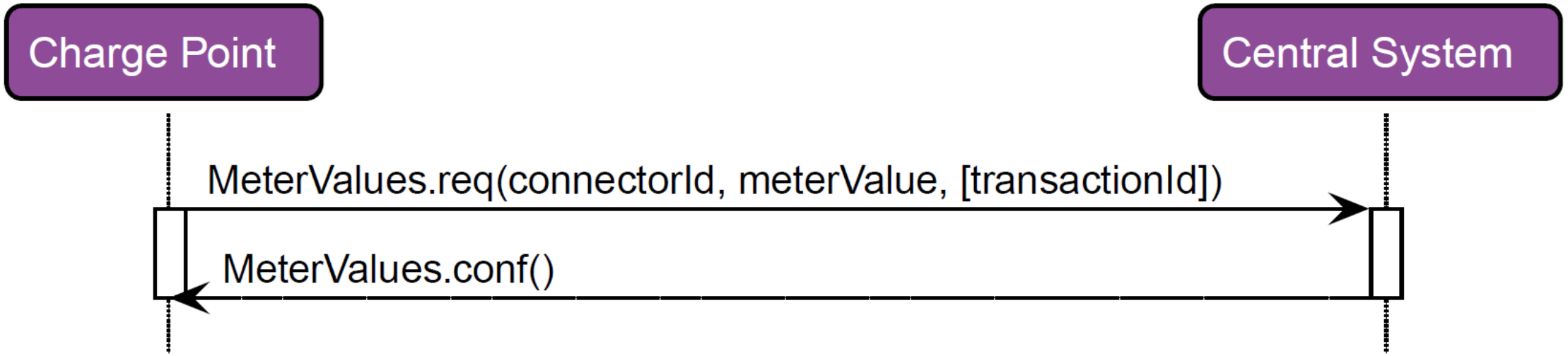
- Central System (CS) can request a Charge Point (CP) to unlock a connector in case of malfunction of the connector cable retention



Linumiz

OCPP: Meter Values

- A Charge Point (CP) may sample the energy meter readings or other sensor to provide extra information about its meter values

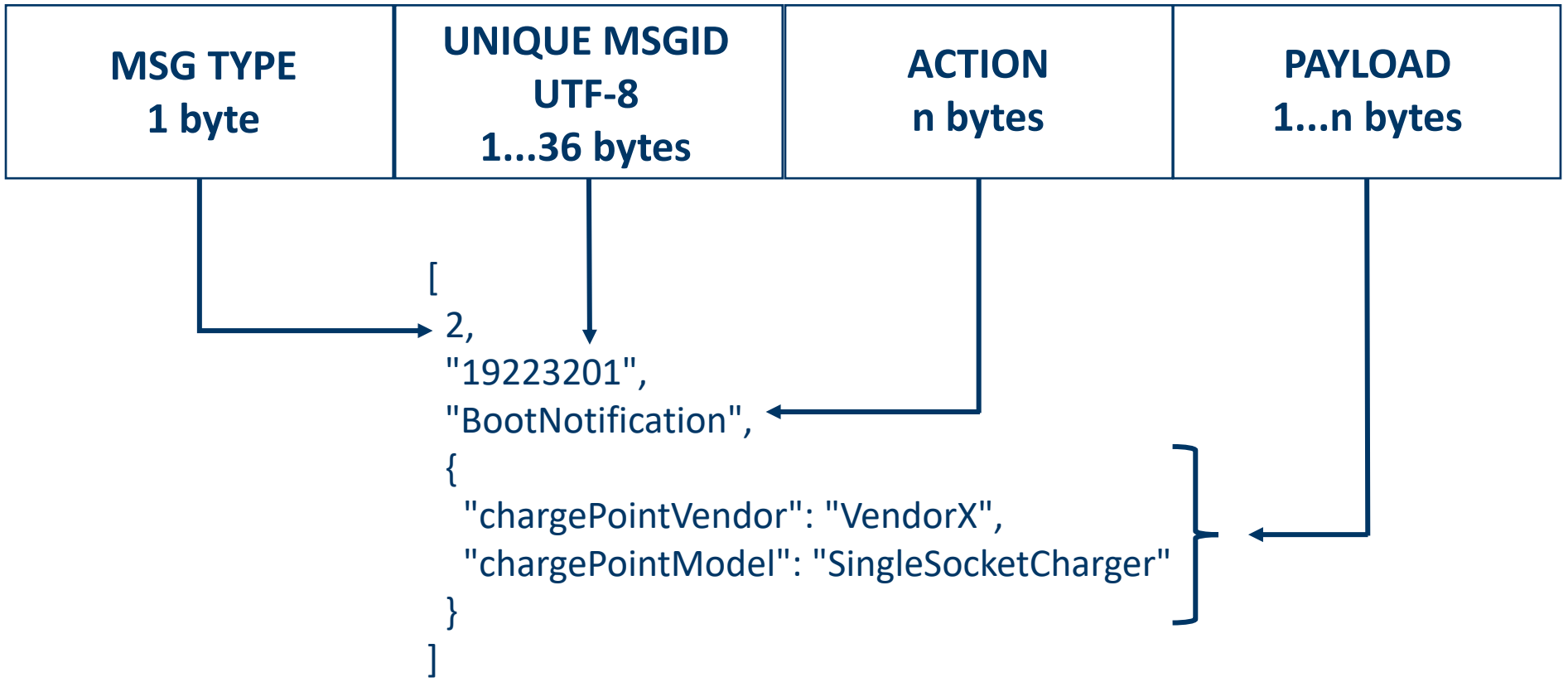


Linumiz

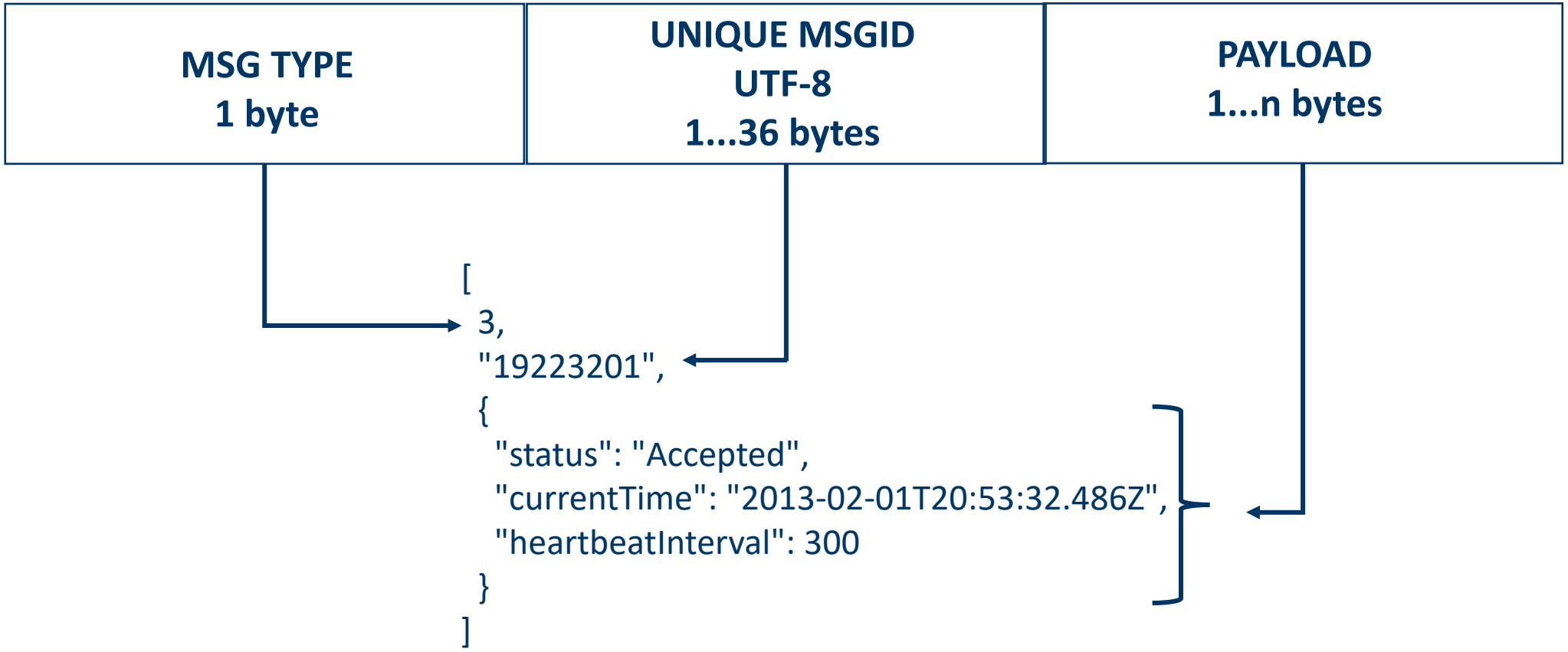
OCPP: RPC (Remote Procedure Call) message frame

- OCPP message is encoded into a wrapper [message type, unique id, payload (OCPP PDU message)]
- message type (CALL) to send, (CALLRESULT) to receive a reply, or (CALLERROR) any failure with reason
- A Charge Point (CP) or Central System (CS) should not send a new CALL message to the other party until previous CALL messages have been responded or timed out

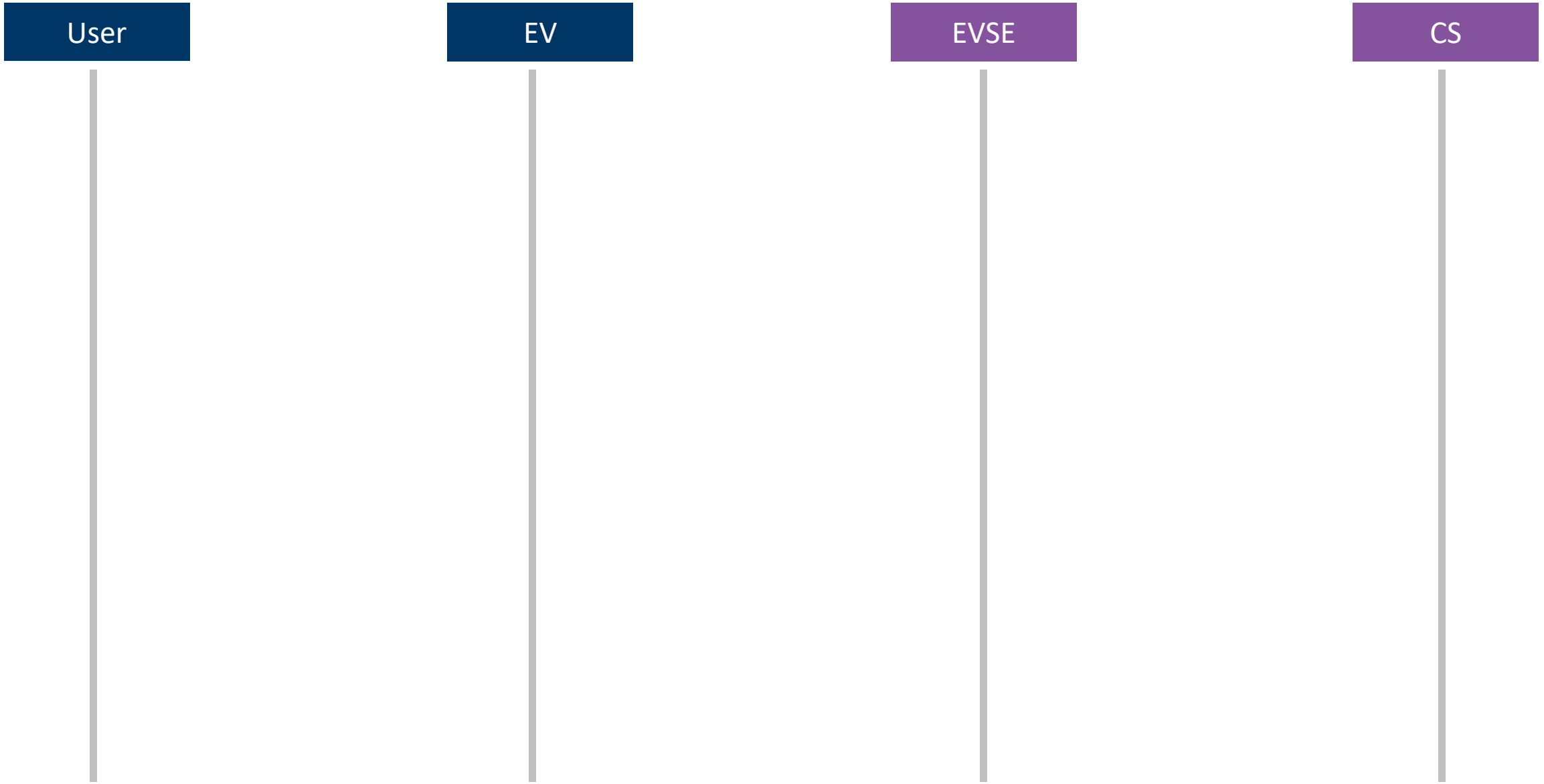
OCPP: CALL REQUEST message frame



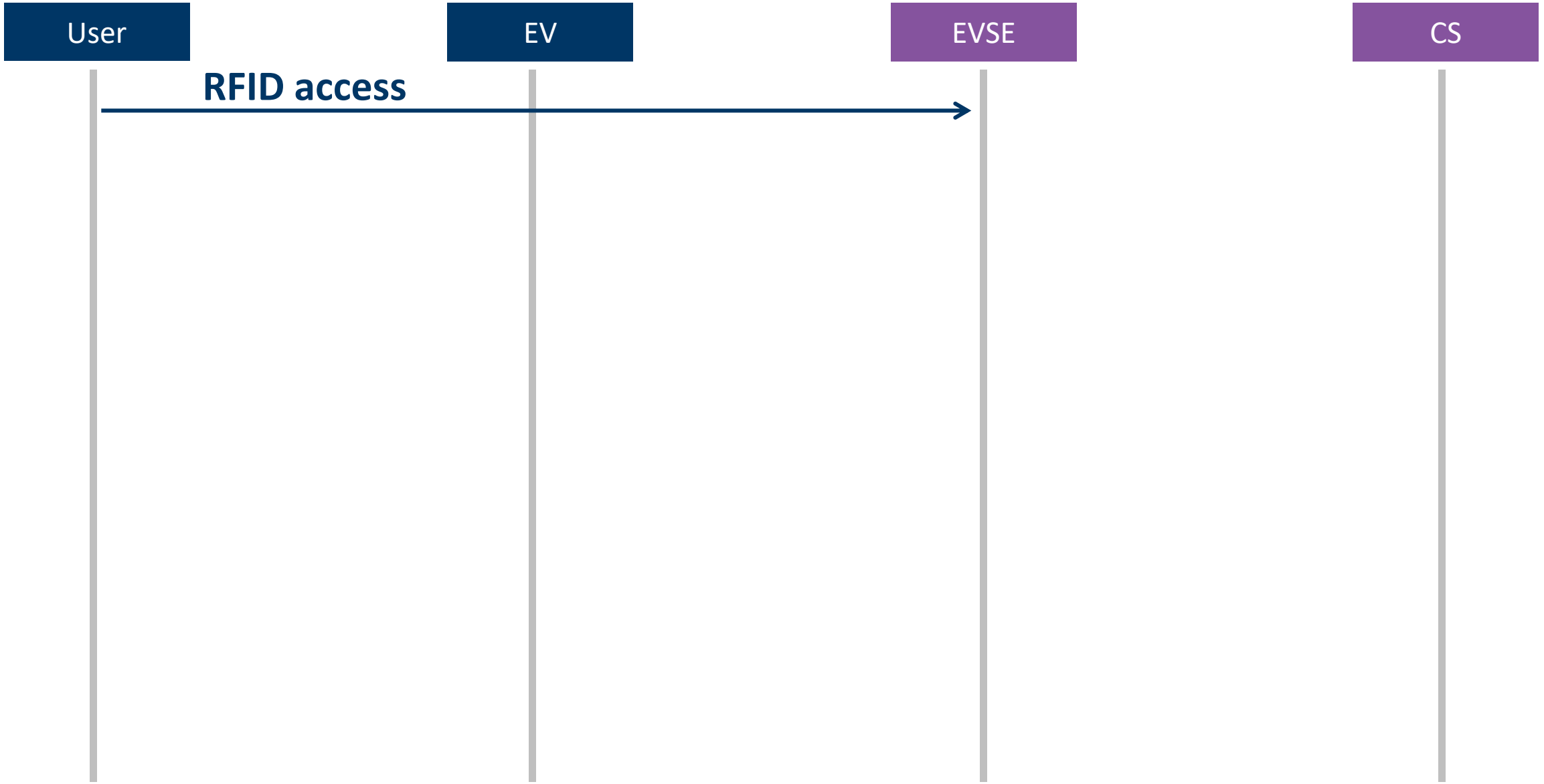
OCPP: CALL RESULT message frame



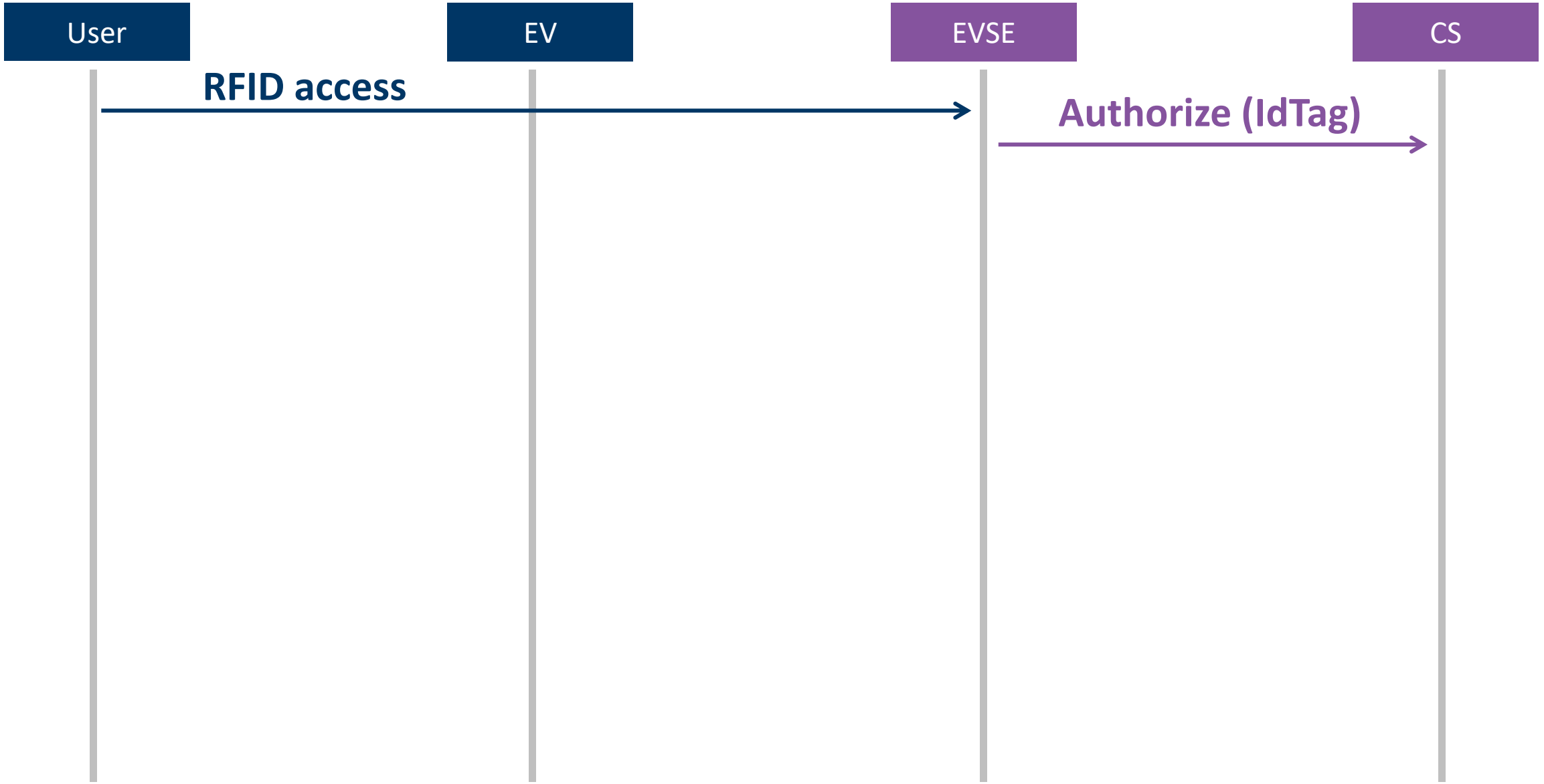
OCPP: Example for a complete charging session



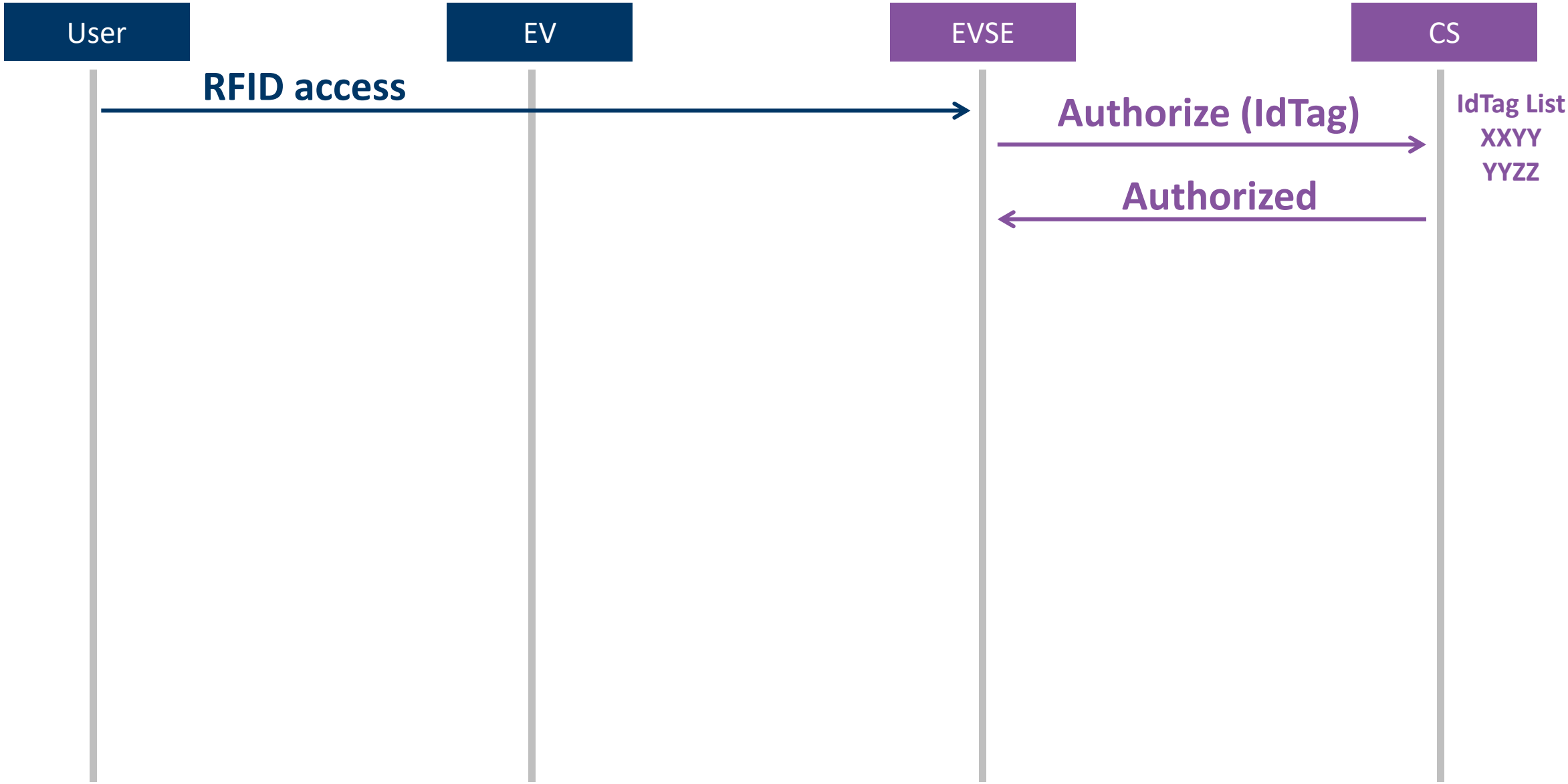
OCPP: Example for a complete charging session



OCPP: Example for a complete charging session

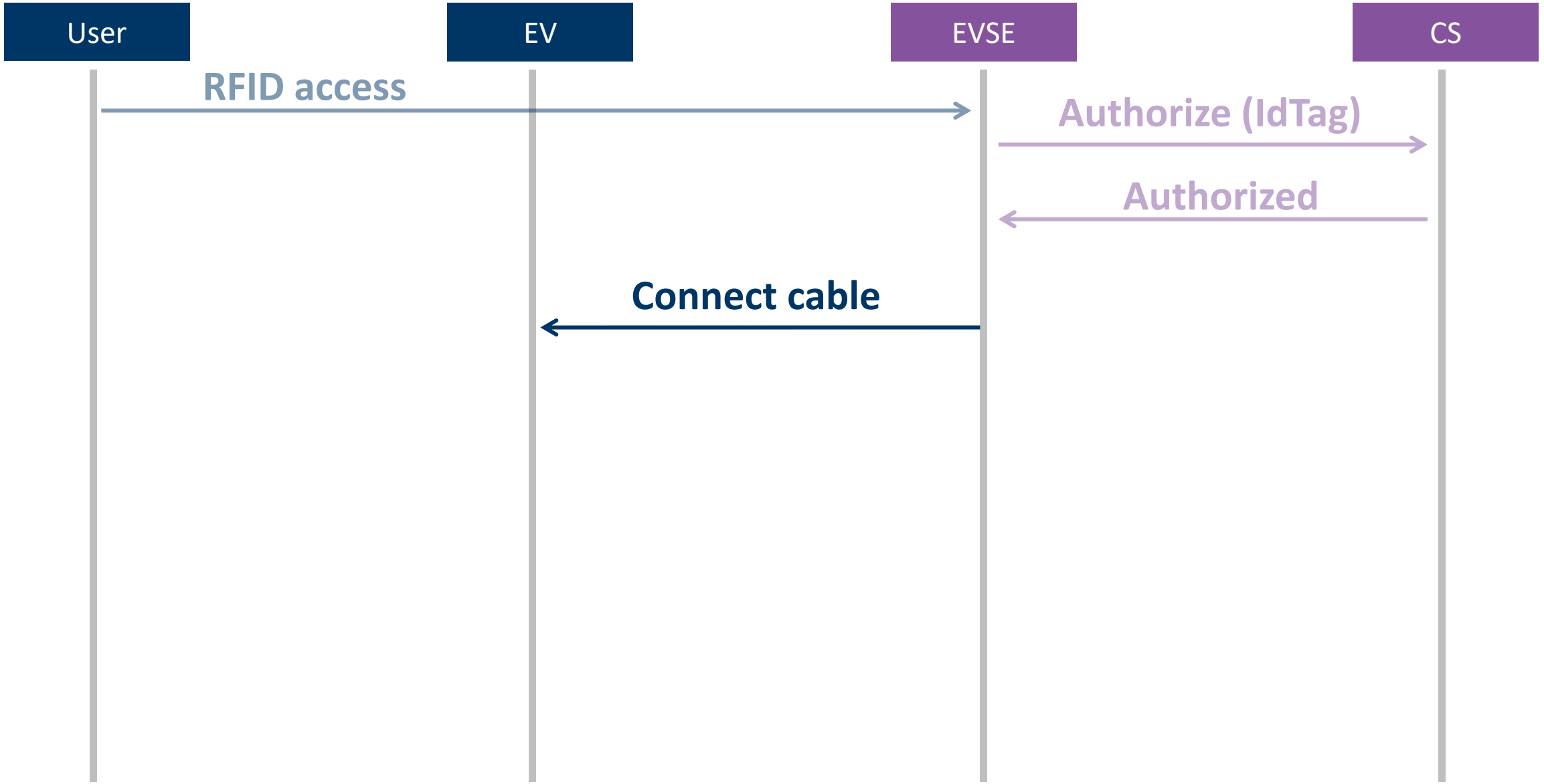


OCPP: Example for a complete charging session

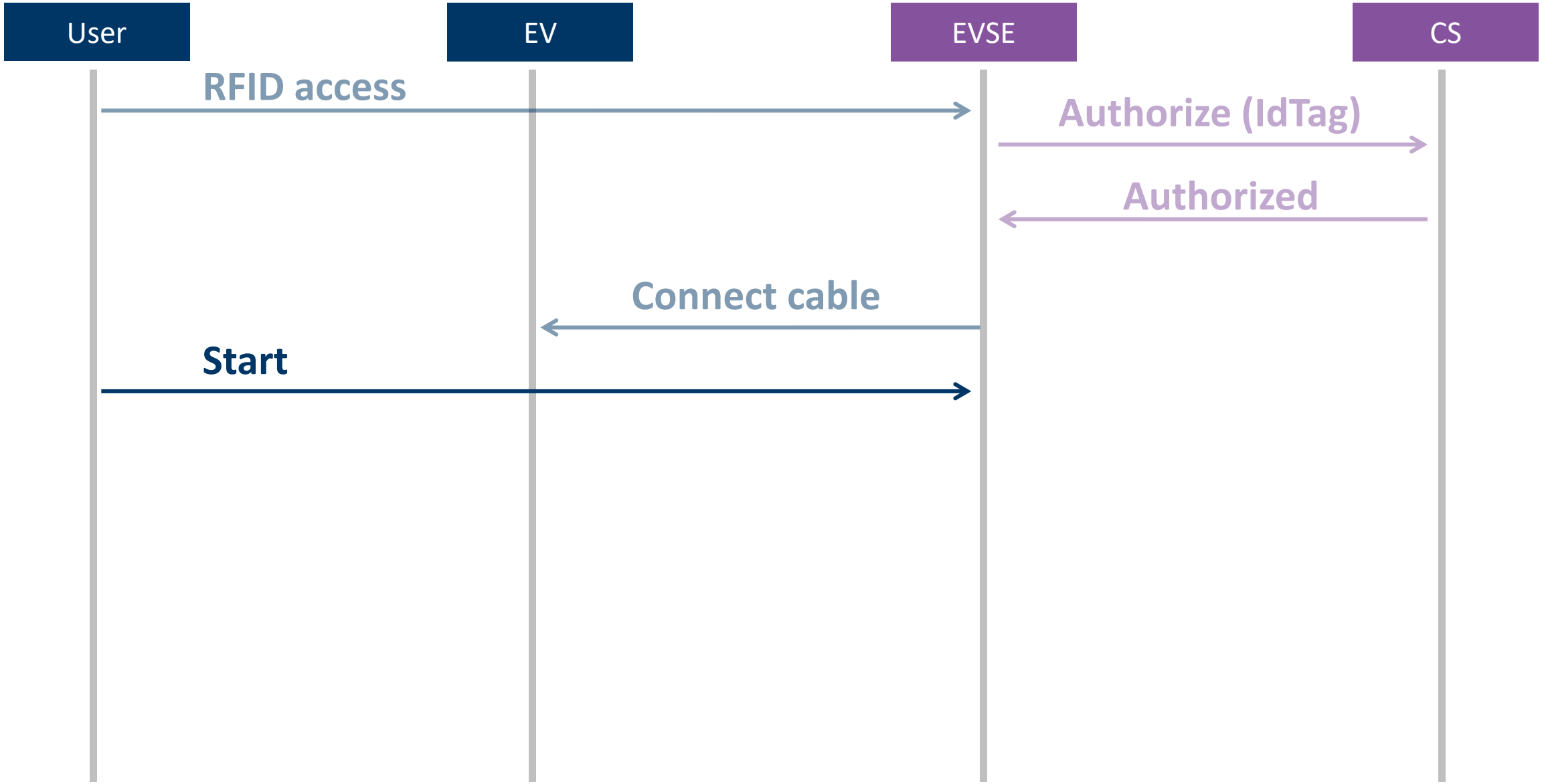


Linumiz

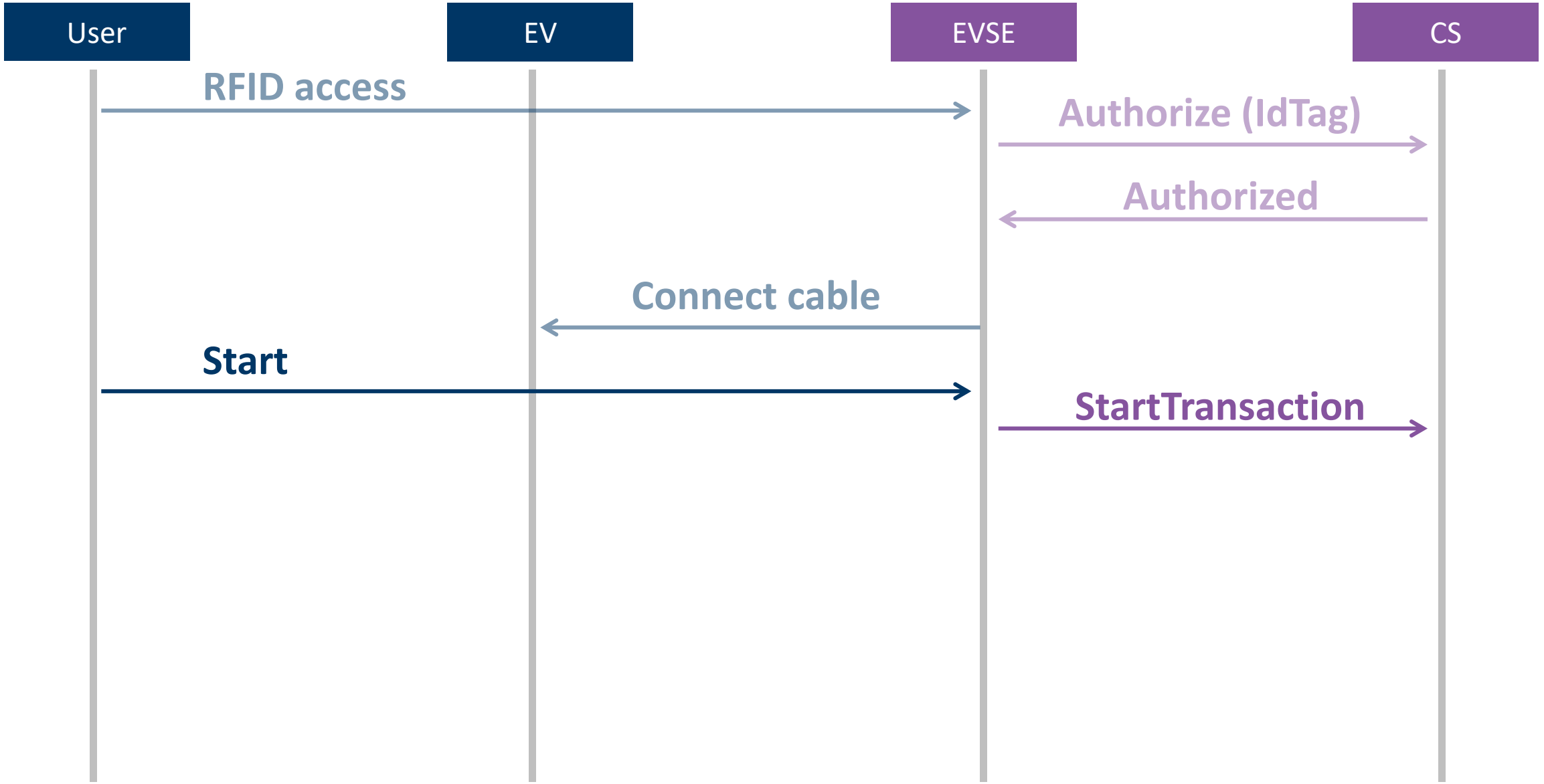
OCPP: Example for a complete charging session



OCPP: Example for a complete charging session

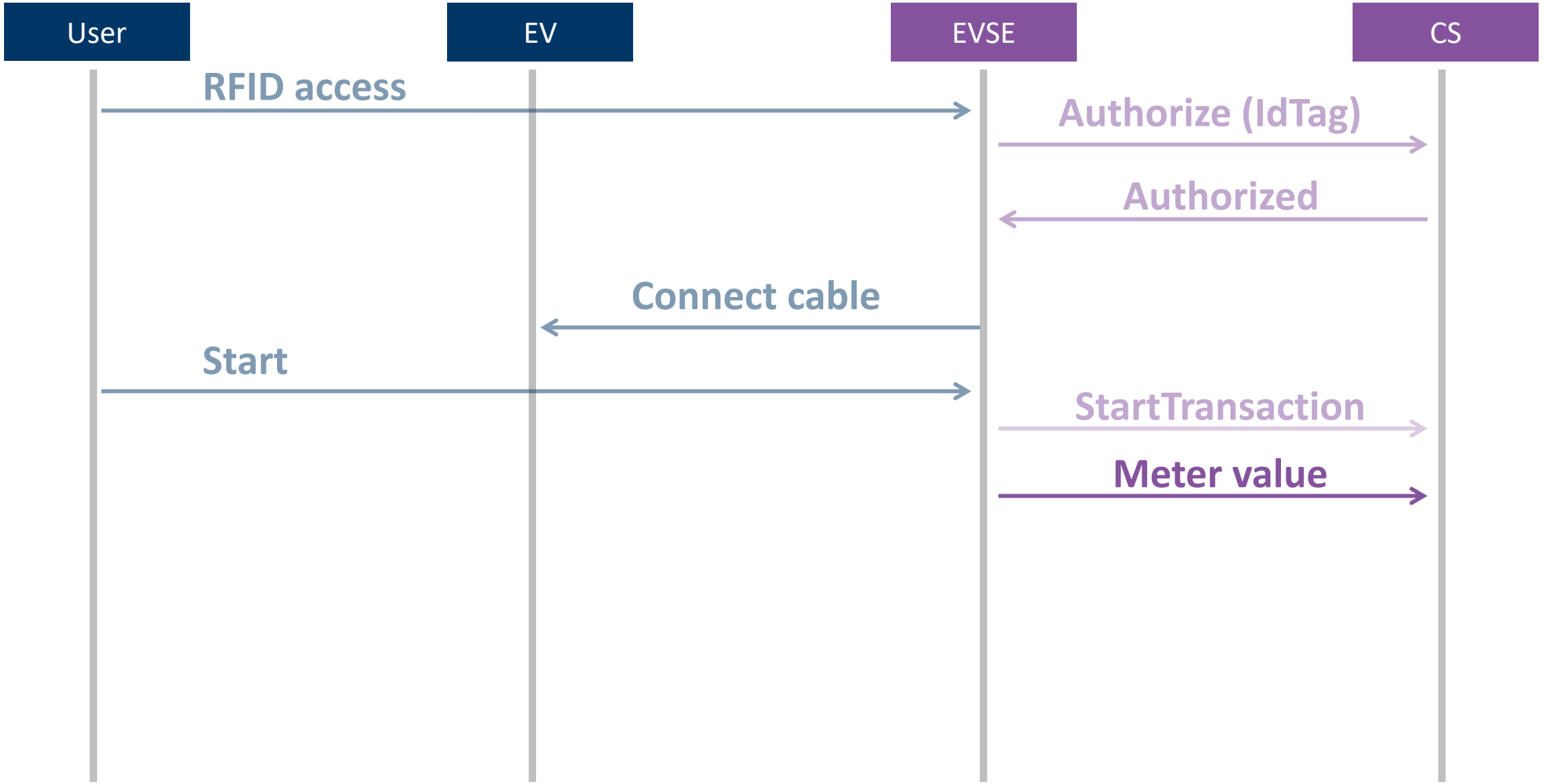


OCPP: Example for a complete charging session

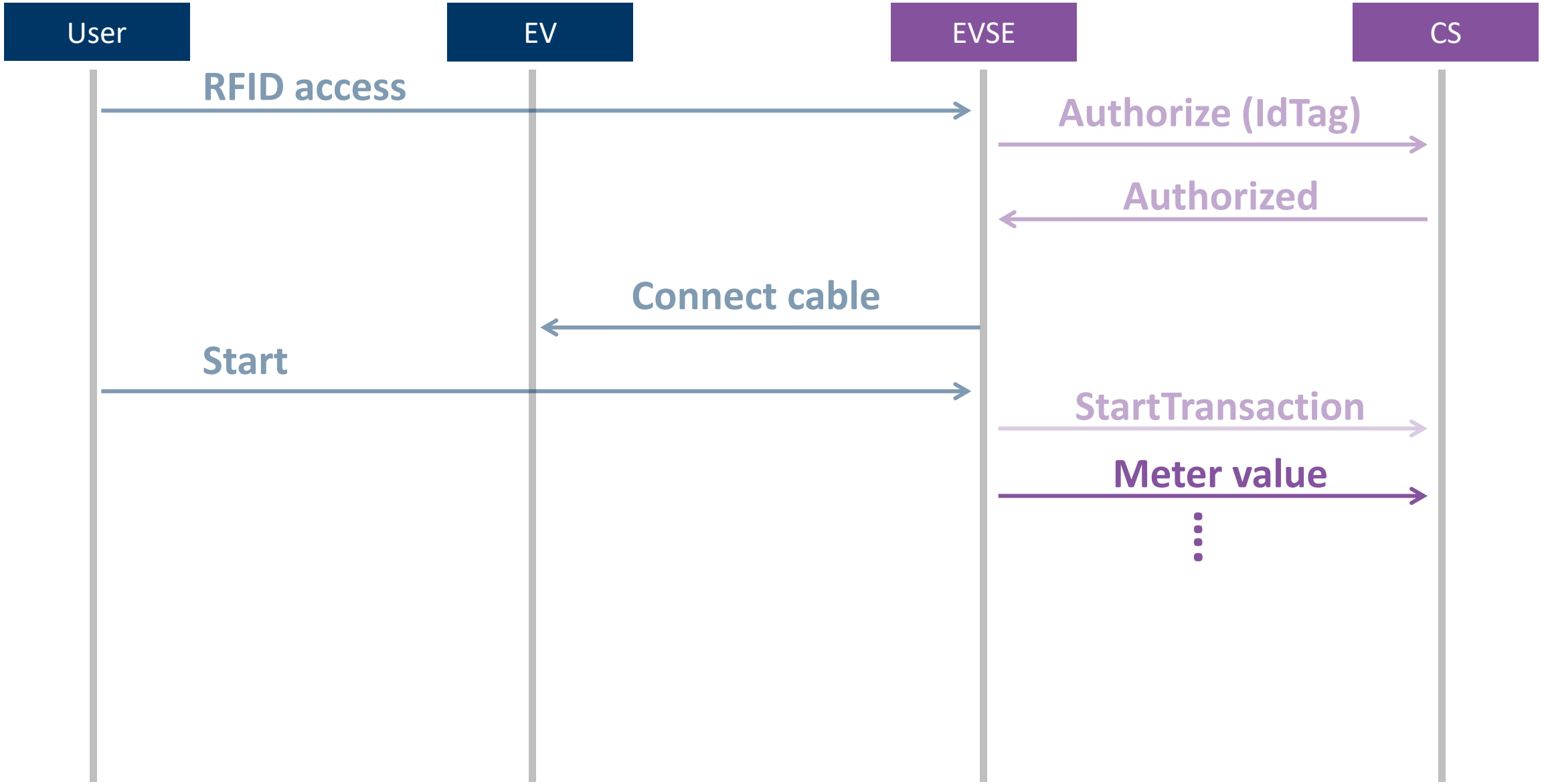


Linumiz

OCPP: Example for a complete charging session

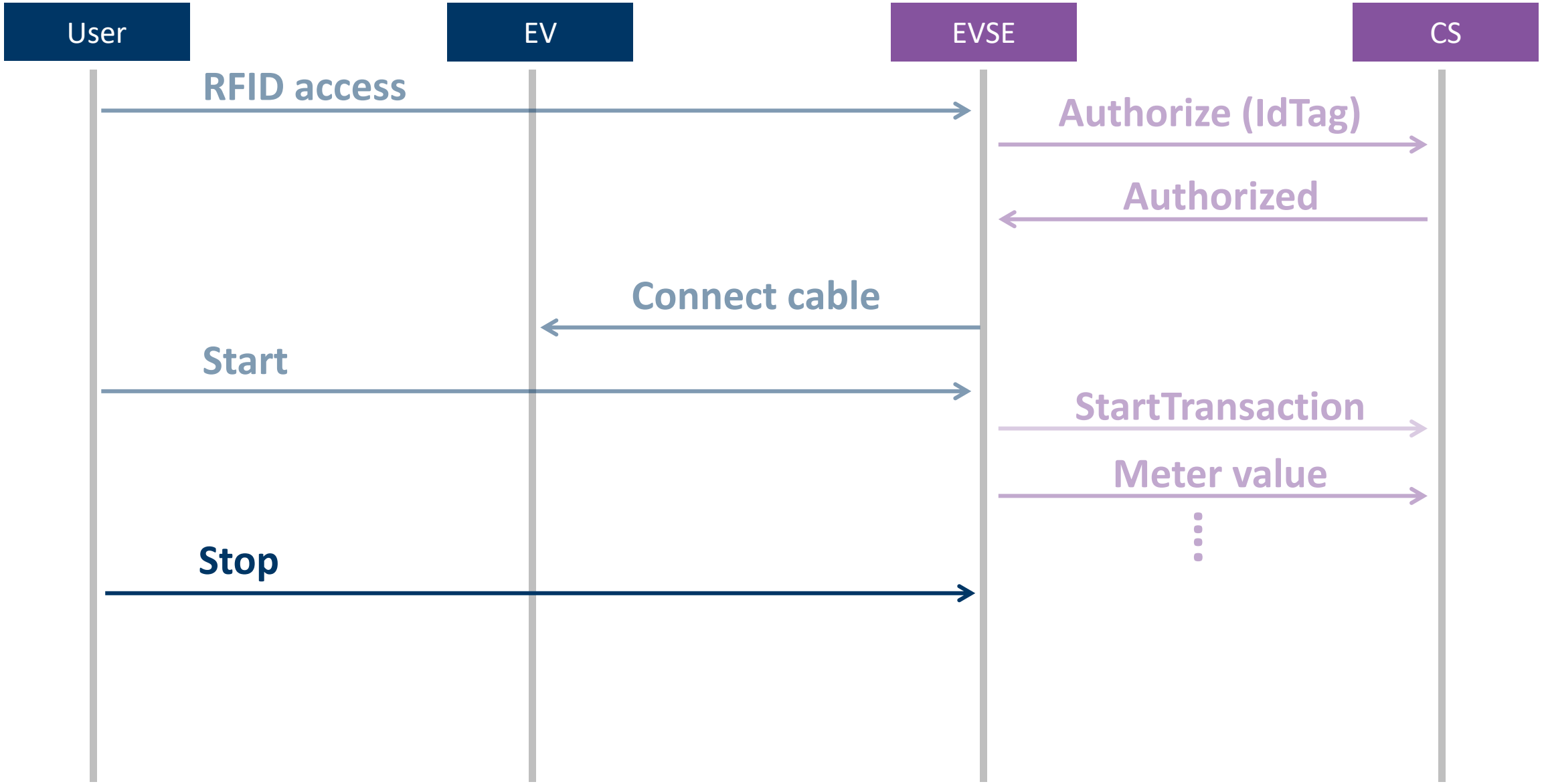


OCPP: Example for a complete charging session

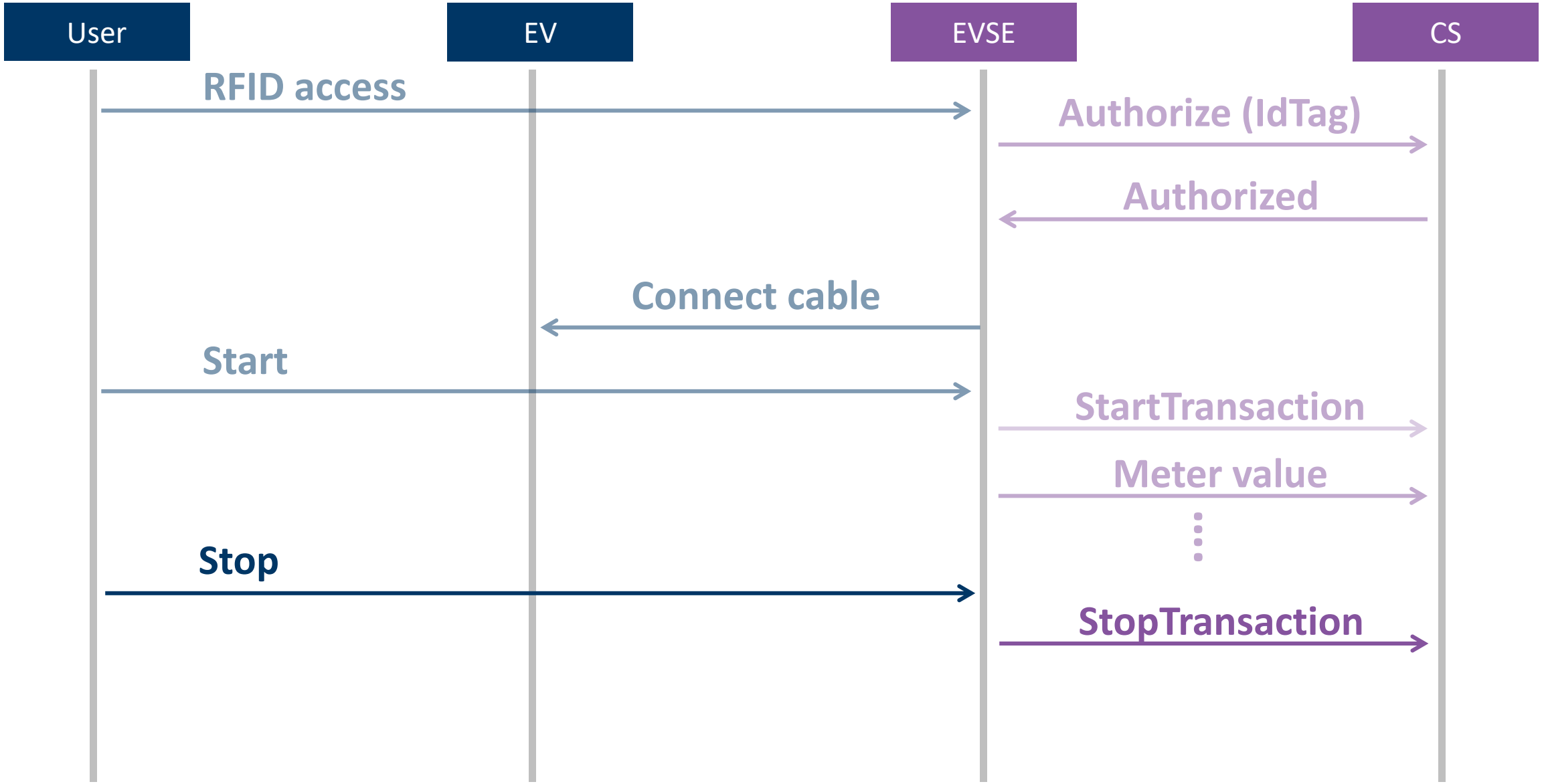


Linumiz

OCPP: Example for a complete charging session

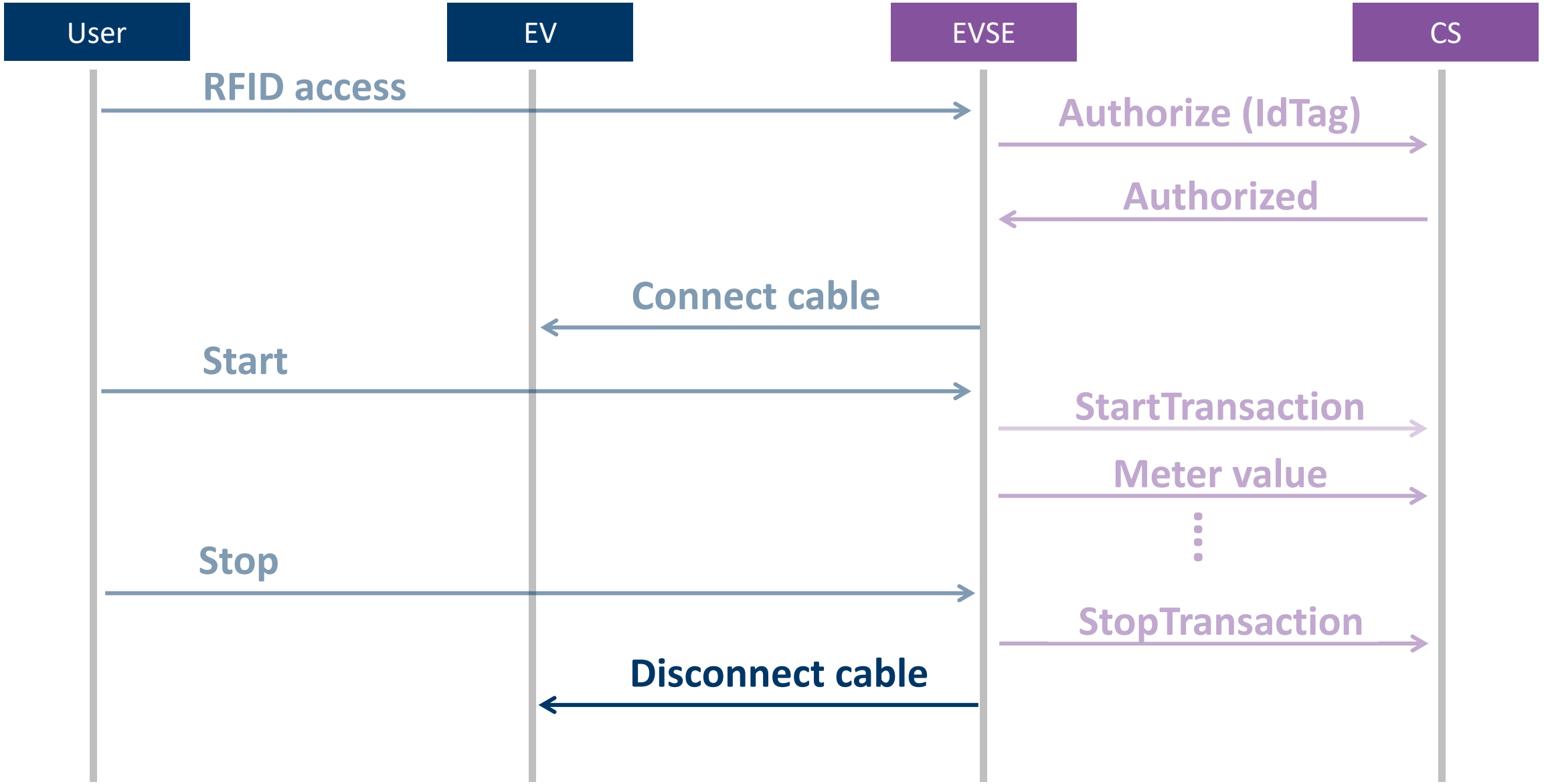


OCPP: Example for a complete charging session

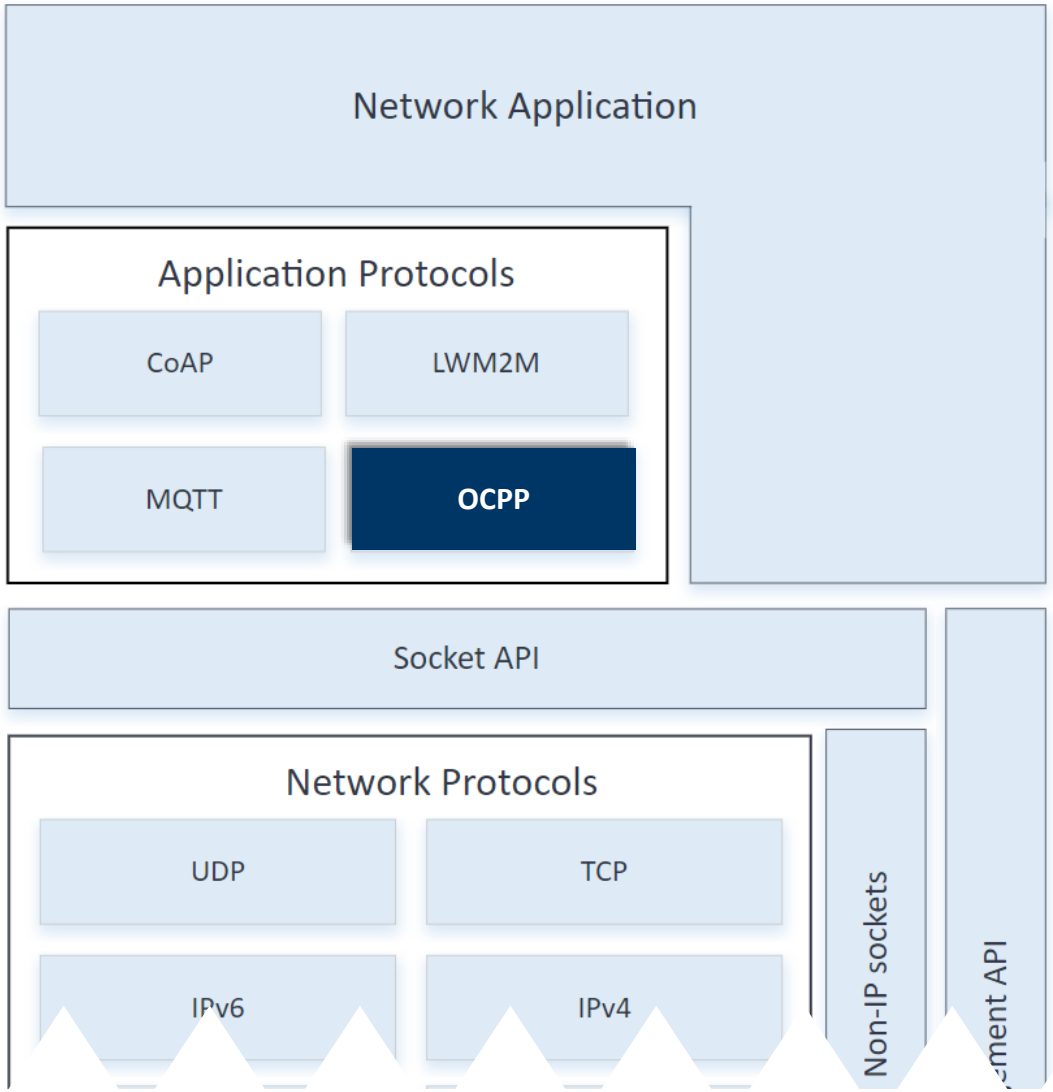


Linumiz

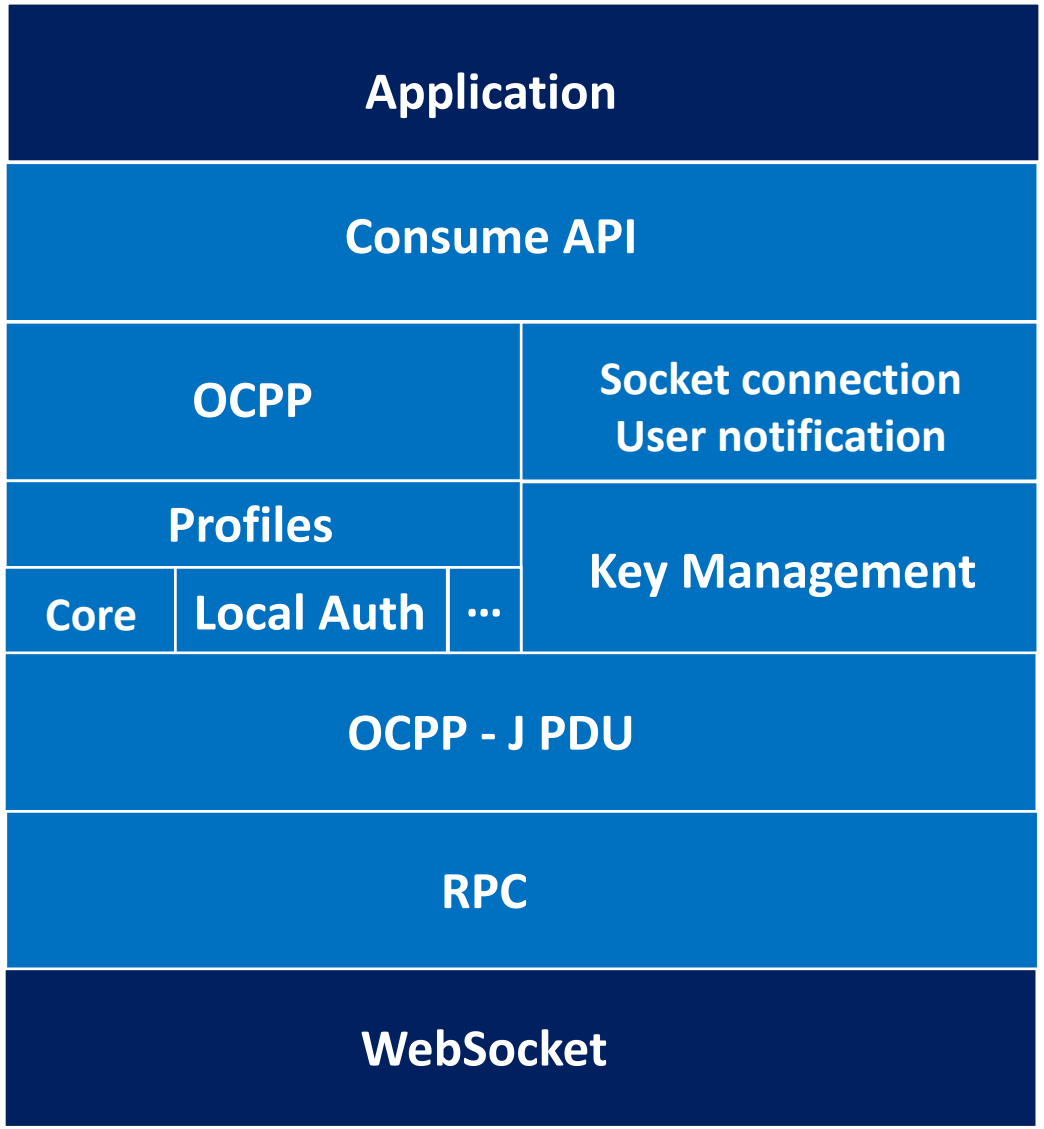
OCPP: Example for a complete charging session



Protocol stack representation in Zephyr



OCPP stack in Zephyr RTOS



OCPP stack API in Zephyr RTOS

```
int ocpp_init(struct ocpp_cp_info *cpi,
              struct ocpp_cs_info *csi,
              ocpp_user_notify_callback_t cb,
              void *user_data);
```

- Initialize the OCPP library with,
  - Charge Point model, vendor, and number of connectors
  - Central System IP address, port, and WebSocket URL
- Before Initialize the OCPP library, the network interface should be ready (ethernet/Wi-Fi/ modem)

OCPP stack API in Zephyr RTOS (Cont.,)

```
int ocpp_session_open(ocpp_session_handle_t *hndl);

void ocpp_session_close(ocpp_session_handle_t hndl);
```

- Each connector should open a unique session after *ocpp_init* for any OCPP transaction
- Session management is internal to the stack, not related to the specification

OCPP stack API in Zephyr RTOS (Cont.,)

```
int ocpp_authorize(ocpp_session_handle_t hndl,
                   char *idtag,
                   enum ocpp_auth_status *status,
                   uint32_t timeout_ms);
```

```
enum ocpp_auth_status {
        OCPP_AUTH_INVALID,
        OCPP_AUTH_ACCEPTED,
        OCPP_AUTH_BLOCKED,
        OCPP_AUTH_EXPIRED,
        OCPP_AUTH_CONCURRENT_TX
};
```

- used for authorization of idtag

OCPP stack API in Zephyr RTOS (Cont.,)

```
int ocpp_start_transaction(ocpp_session_handle_t hndl,
                           int Wh,
                           uint8_t conn_id,
                           uint32_t timeout_ms);
```

- Notify the Central system that a transaction has been started for the connector ID
- Energy meter reading at the time of start

OCPP stack API in Zephyr RTOS (Cont.,)

```
enum ocpp_notify_reason {
        /** User must fill the current reading */
        OCPP_USR_GET_METER_VALUE,

        /** Process the start charging request as like idtag received from local
         *  e.g authorize etc
         */
        OCPP_USR_START_CHARGING,

        /** Process the stop charging sequence */
        OCPP_USR_STOP_CHARGING,

        /** Unlock mechanical connector of CP */
        OCPP_USR_UNLOCK_CONNECTOR,
};
```

```
typedef int (*ocpp_user_notify_callback_t)(enum ocpp_notify_reason reason,
                                           union ocpp_io_value *io,
                                           void *user_data);
```

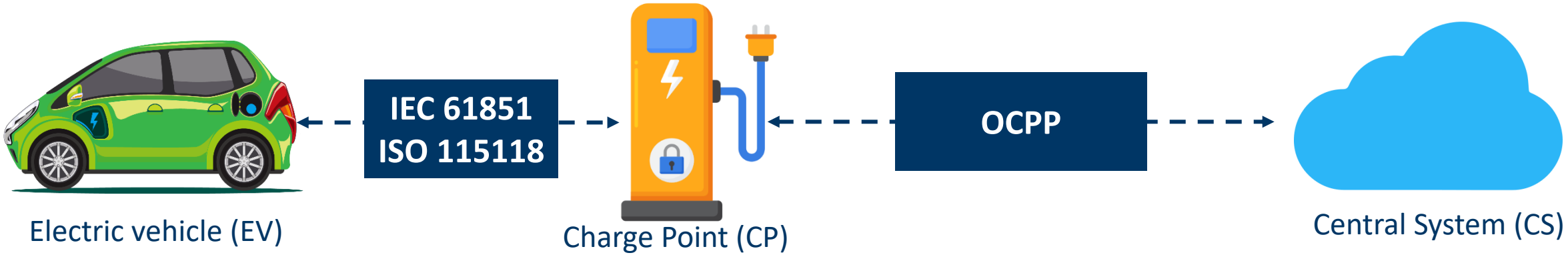
OCPP API stack in Zephyr API (Cont.,)

```
int ocpp_stop_transaction(ocpp_session_handle_t hndl,
                          int Wh,
                          uint32_t timeout_ms);
```

- Notify the Central System that a transaction has stopped
- Energy meter reading at the time of stop

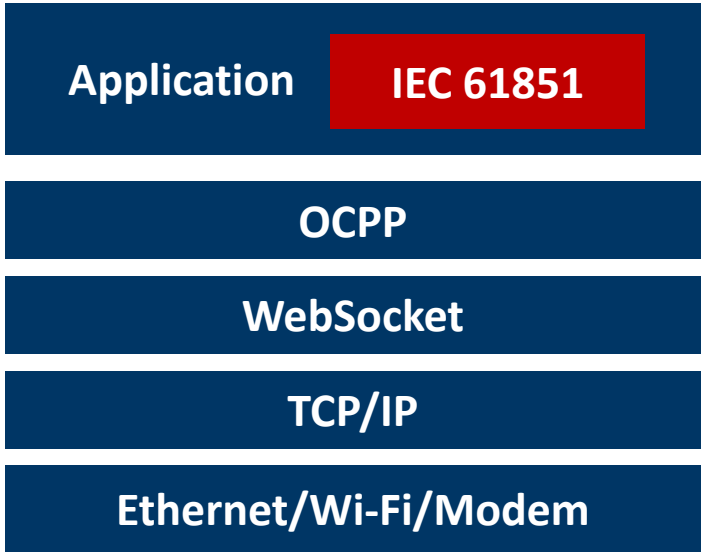
EVSE standards (IEC 61851)

- OCPP covers half of the EVSE, the other part of EVSE handshaking with EV is covered by IEC 61851/ ISO 115118



IEC 61851: Control and Proximity Pilot

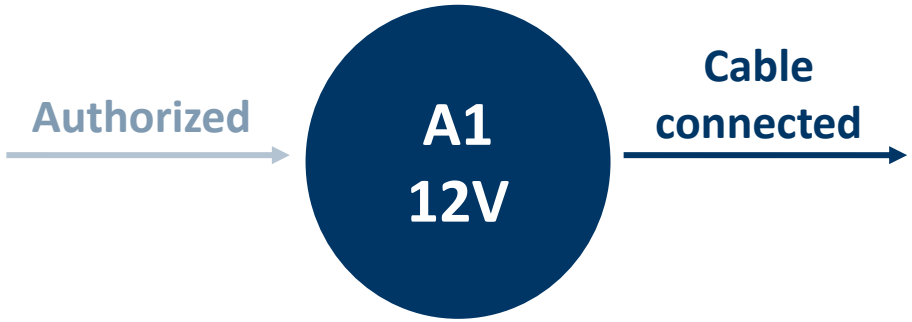
- Control Pilot is a communication line used to negotiate charging level between the car and the EVSE
- Proximity Pilot serves as a charge cable detection and current limitation based on Pulse width modulation (PWM)



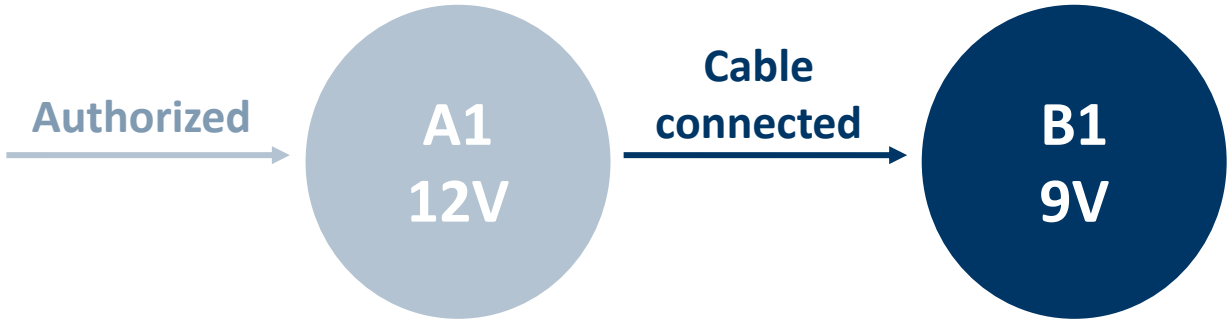
IEC 61851: Control Pilot

IEC 61851: Control Pilot

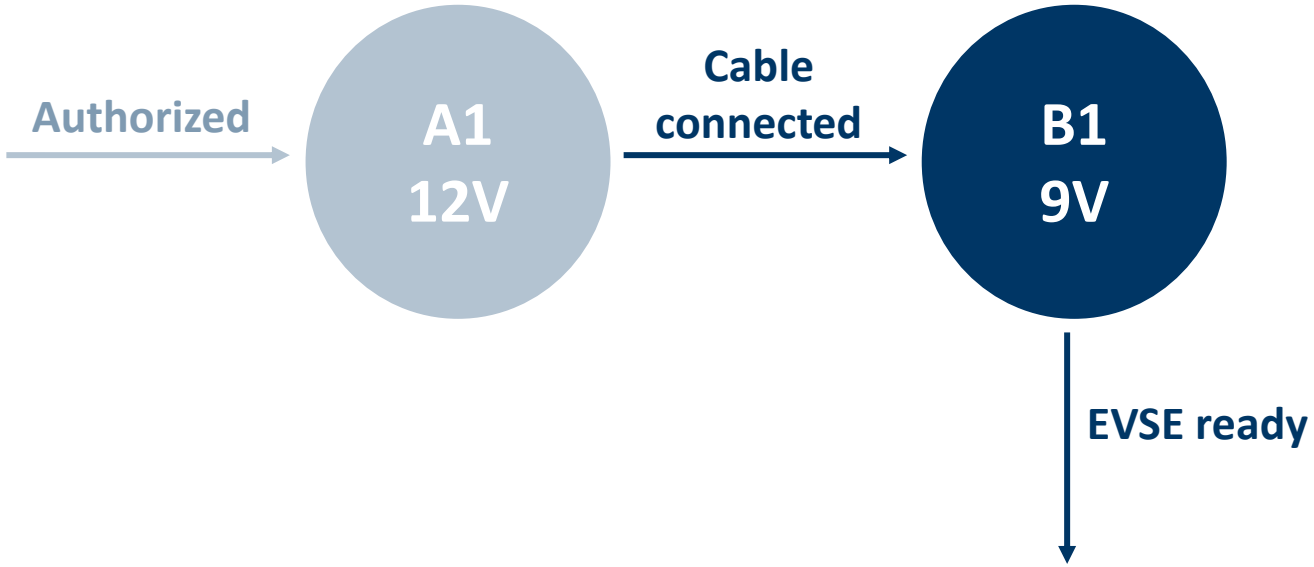
IEC 61851: Control Pilot



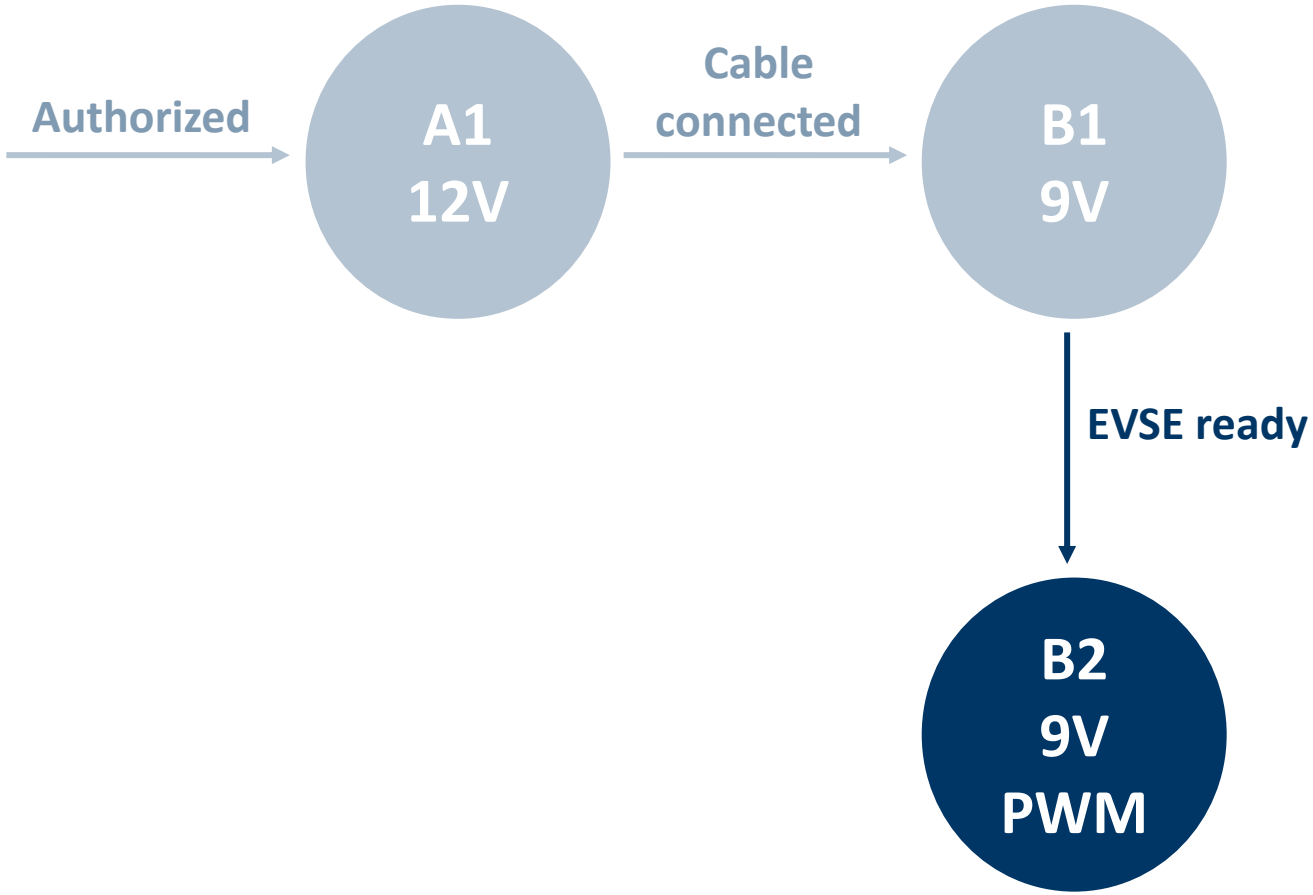
IEC 61851: Control Pilot



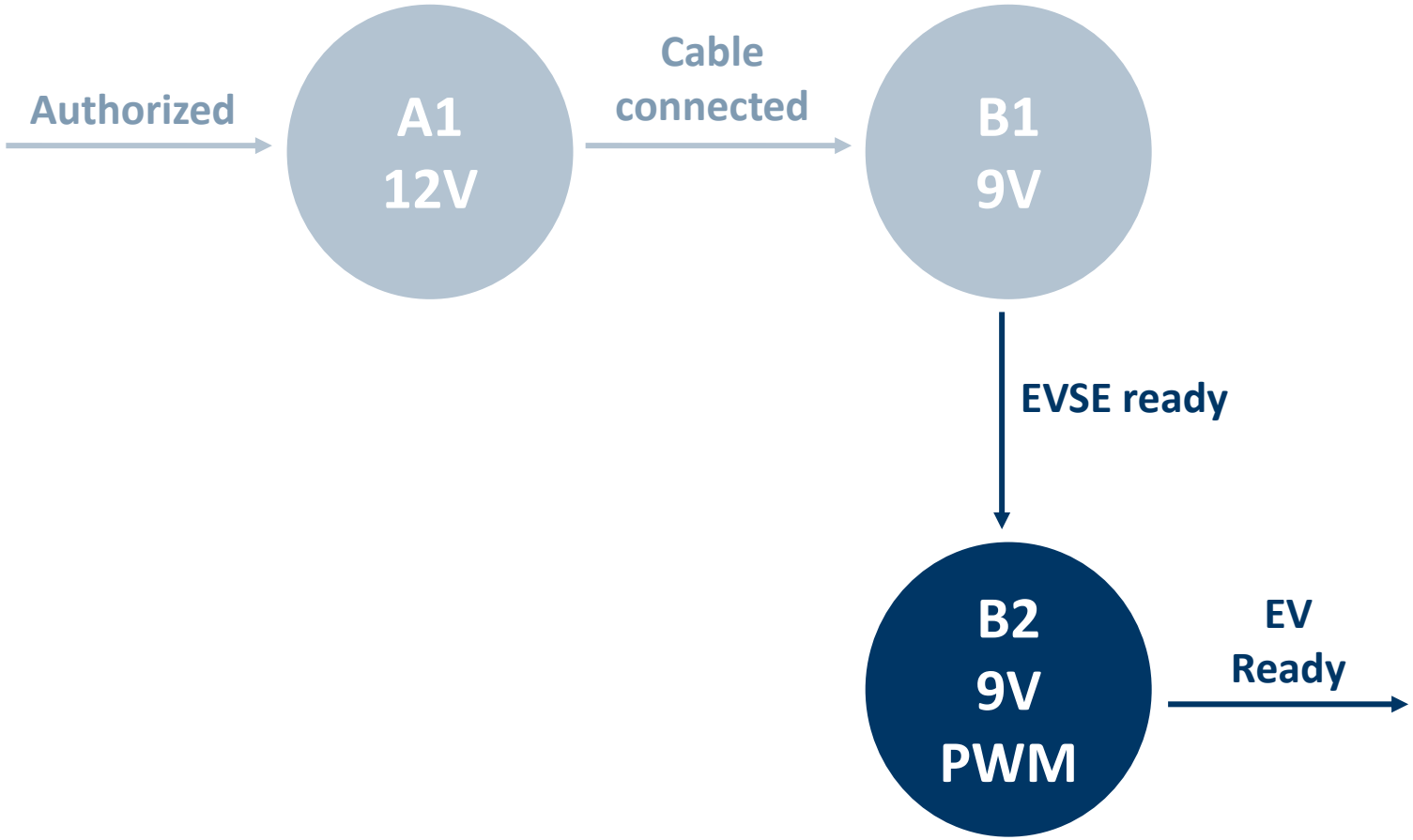
IEC 61851: Control Pilot



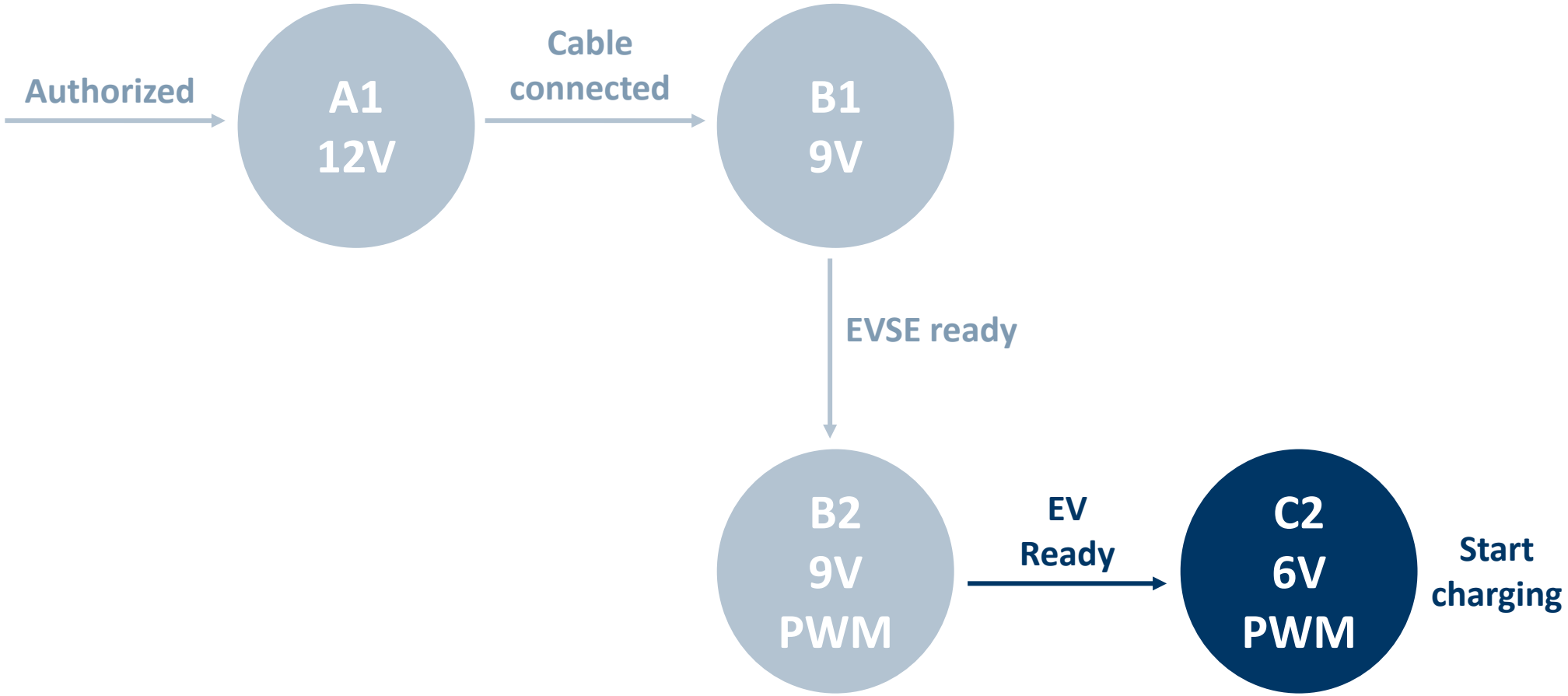
IEC 61851: Control Pilot



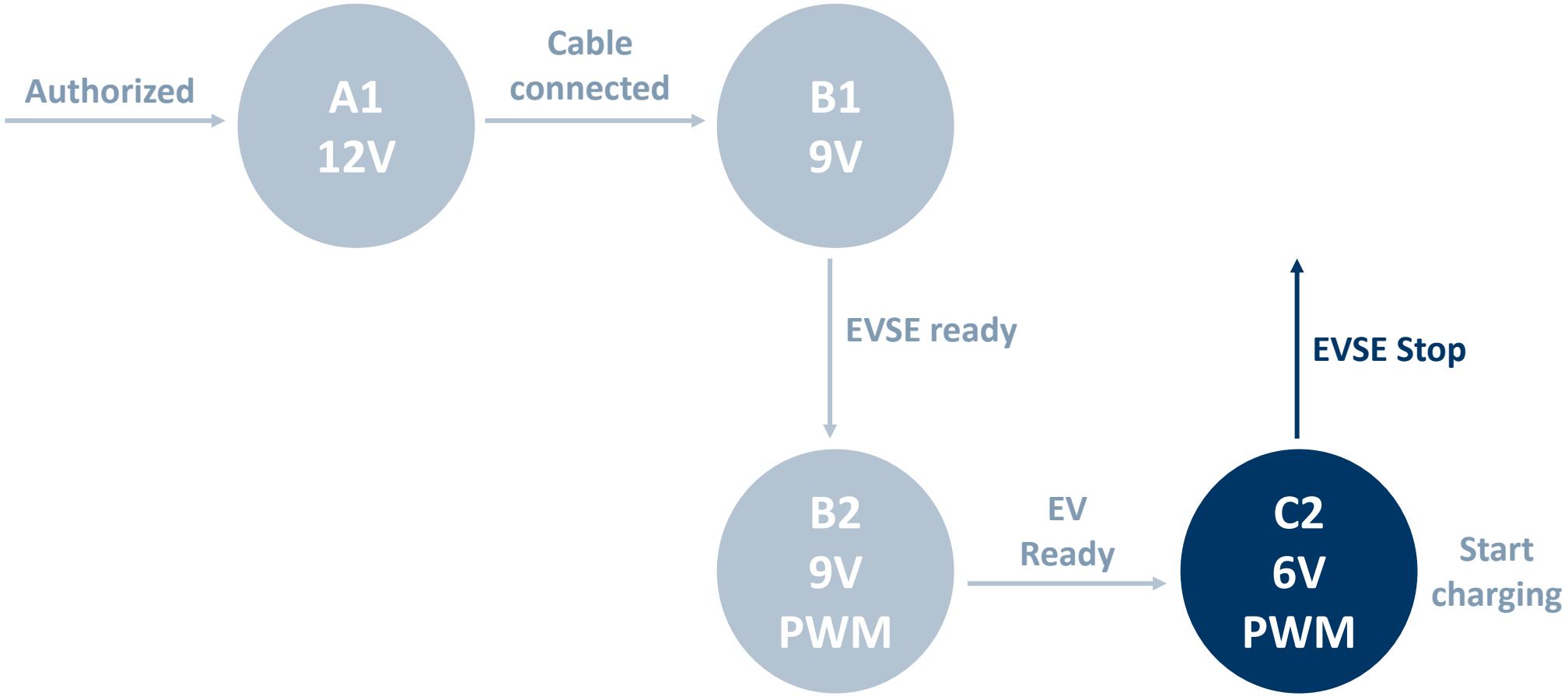
IEC 61851: Control Pilot



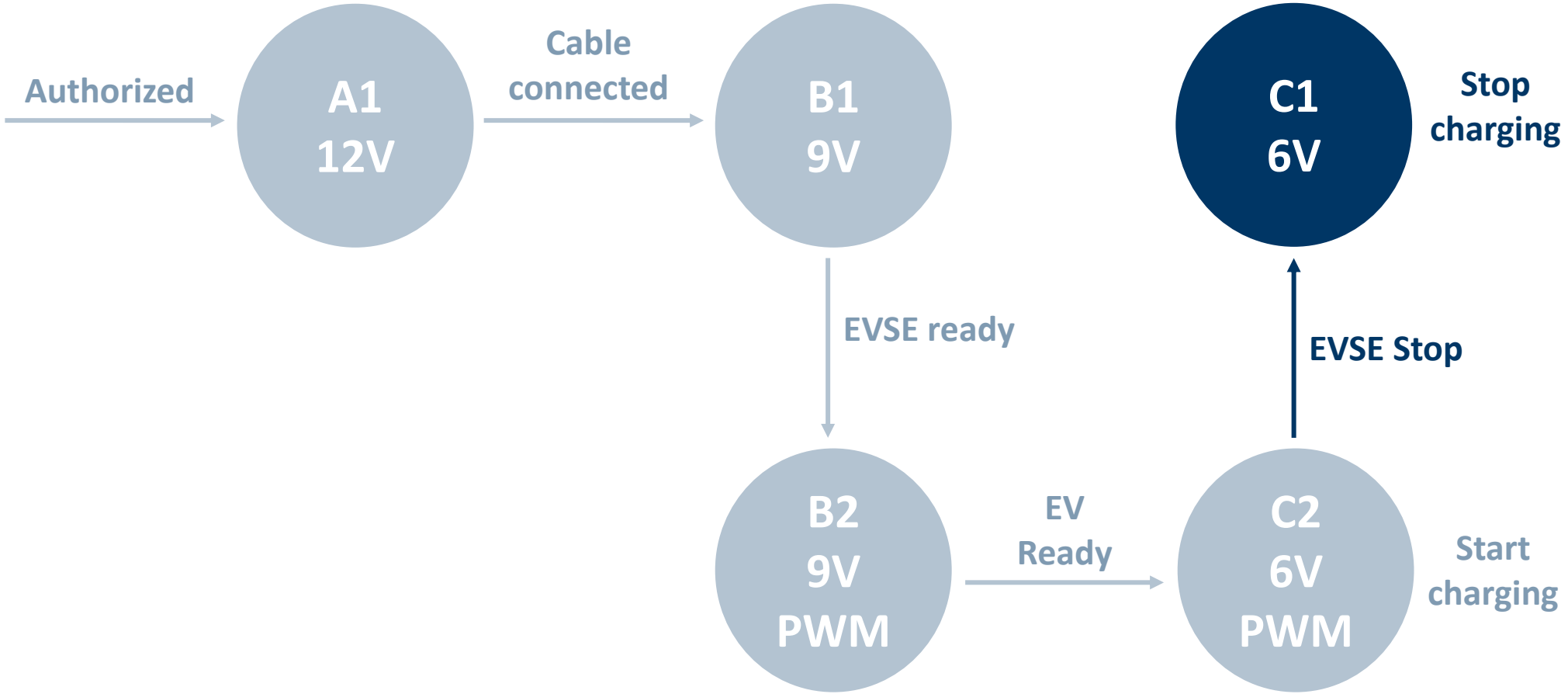
IEC 61851: Control Pilot



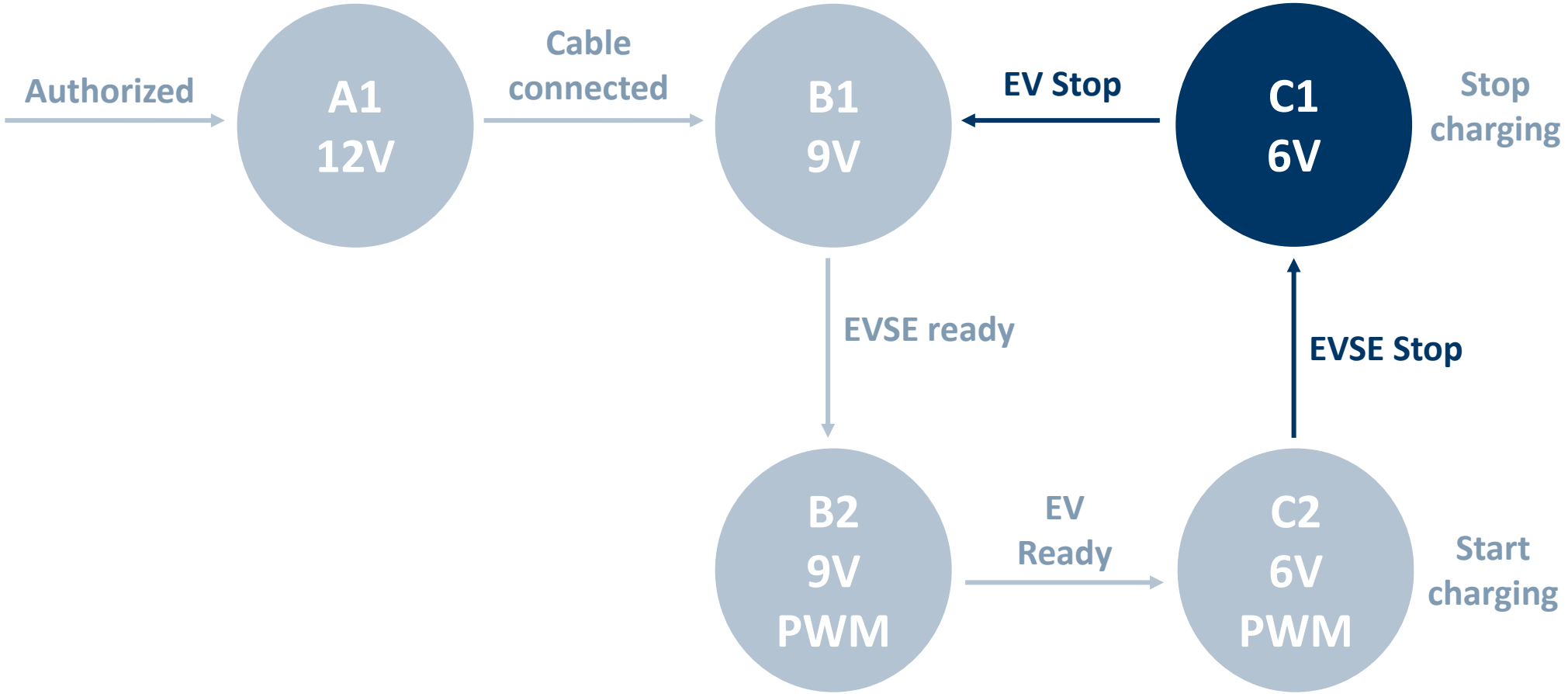
IEC 61851: Control Pilot



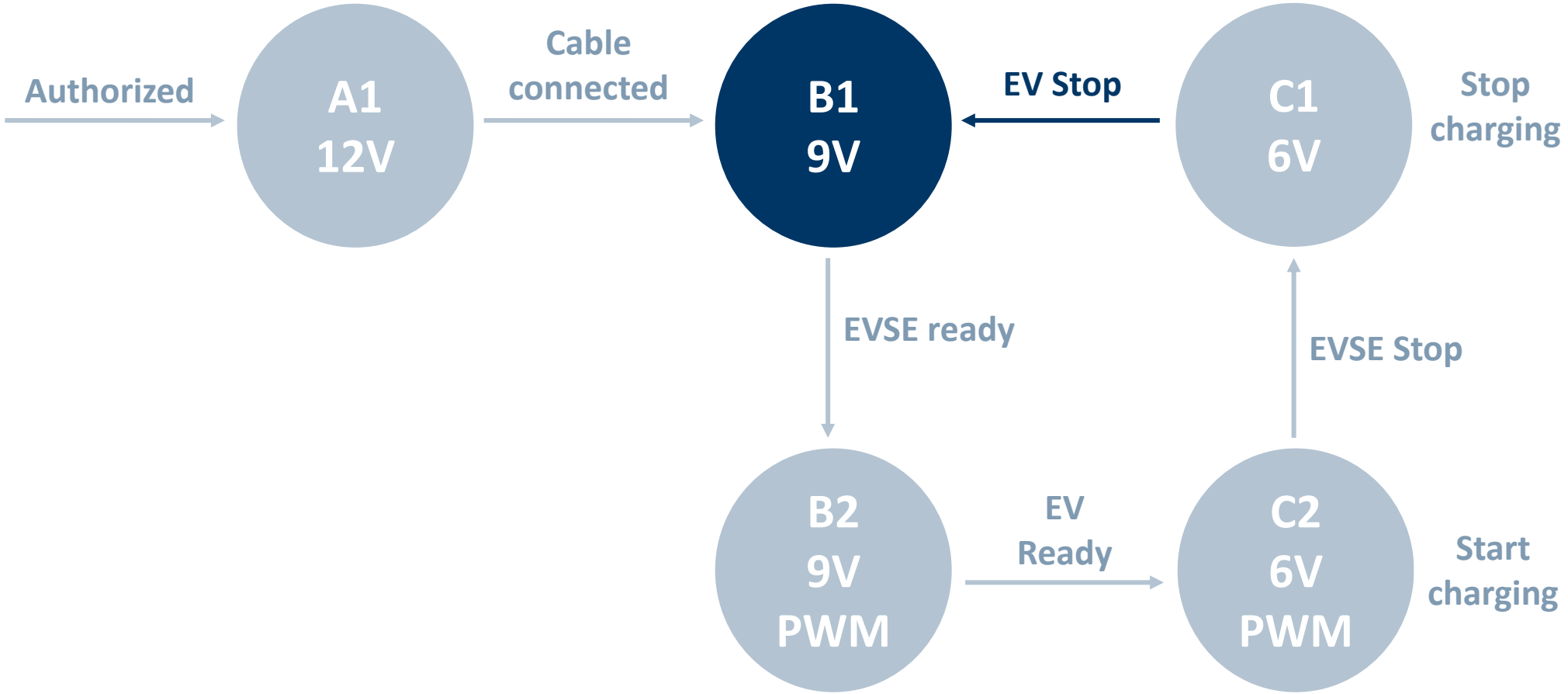
IEC 61851: Control Pilot



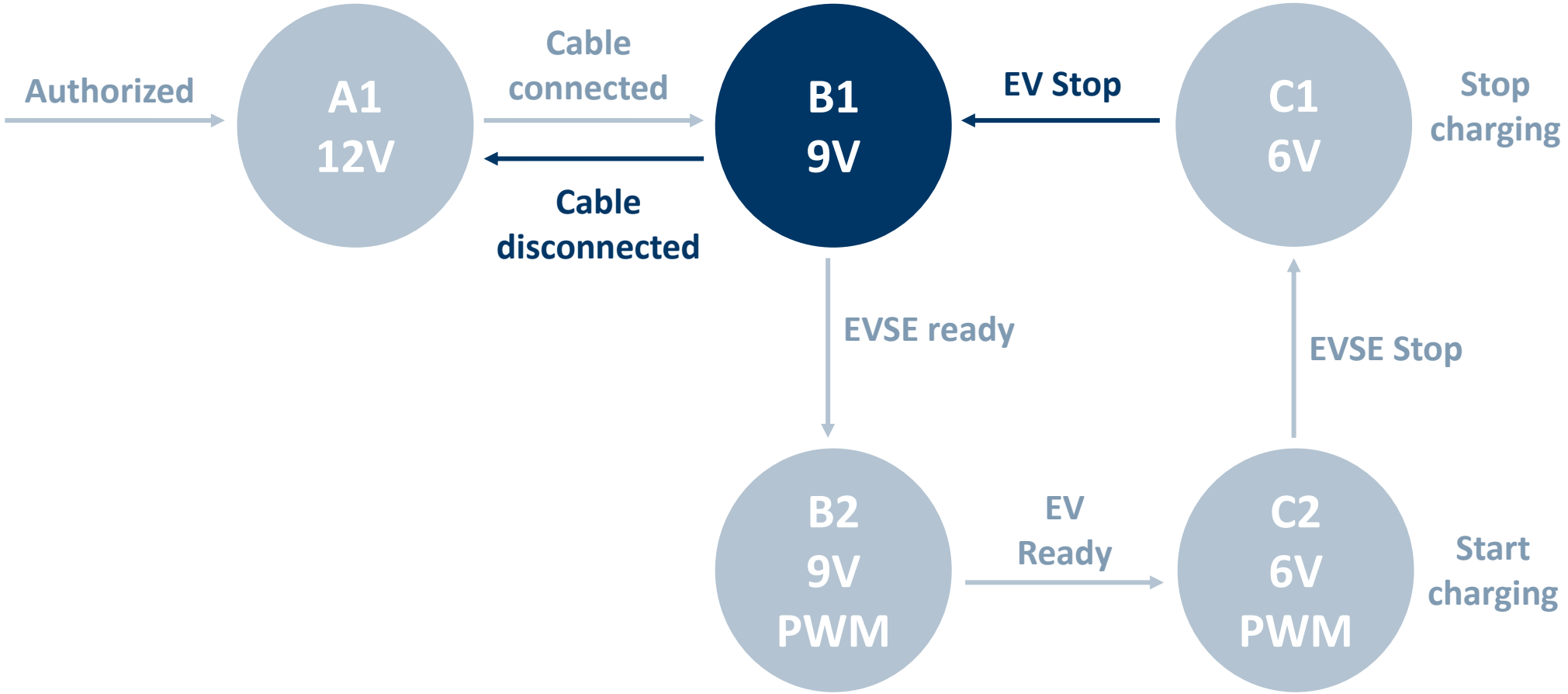
IEC 61851: Control Pilot



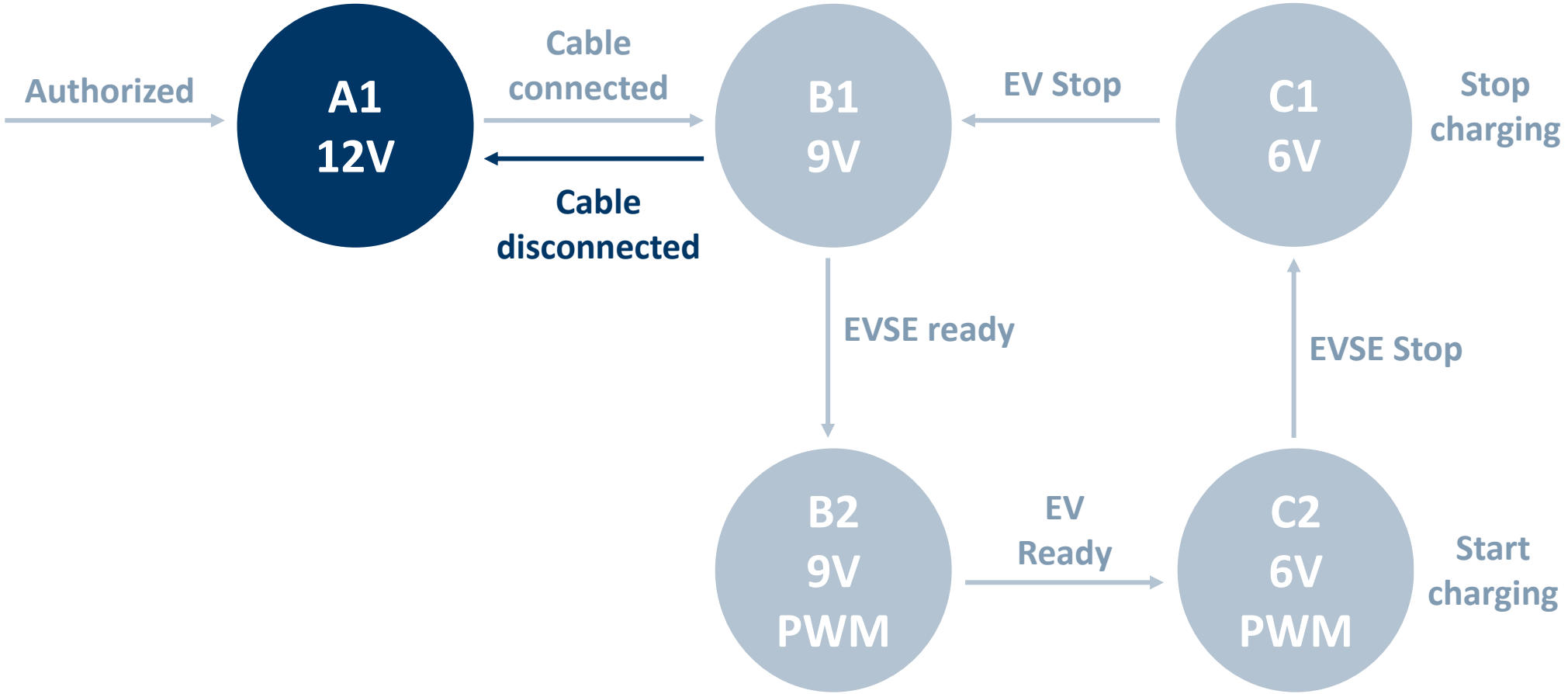
IEC 61851: Control Pilot



IEC 61851: Control Pilot



IEC 61851: Control Pilot



OCPP future work in Zephyr

- PR under review: https://github.com/zephyrproject-rtos/zephyr/pull/68739
- 1.6 core profile supported
- Next step
  - TLS
  - 1.6 optional profiles
  - 2.0.1 support

Questions ?

Saravanan Sekar

saravanan@linumiz.com

Linumiz