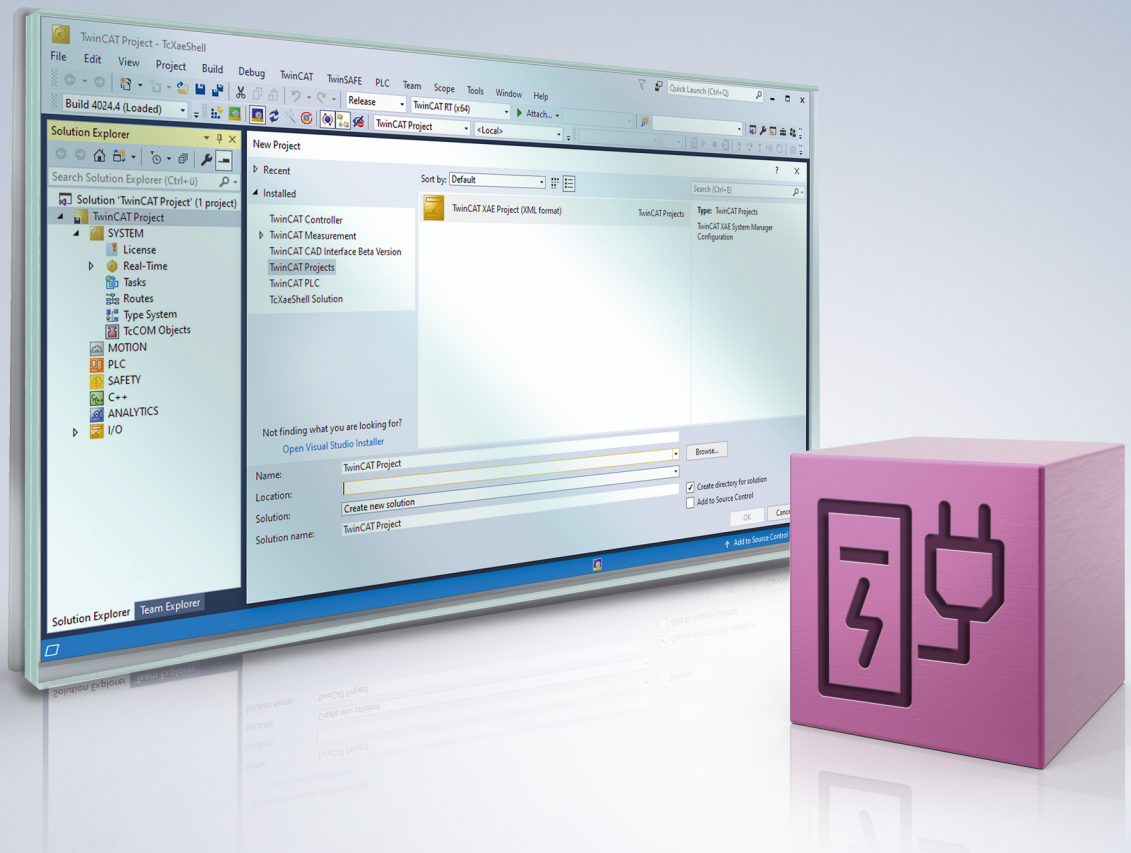
BECKHOFF New Automation Technology

Manual | EN

TF6771

TwinCAT 3 | IoT OCPP



2024-10-31 | Version: 1.1.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all requirements for safety, including all the relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
No claims to modify products that have already been supplied may be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

1.4 Documentation issue status

Version	Change
1.1.0	**New:** Troubleshooting [▶ 212]

2 Overview

With the TwinCAT Function TF6771 IoT OCPP, TwinCAT can act both as an OCPP client and as an OCPP server. The use cases are explained in more detail below.

Product components

The function TF6771 IoT OCPP consists of the following components, which can be used from TwinCAT version 3.1.4026.x:

- **Driver:** TcIotOcpp.tmx (included in the installation of TF6771.IotOCPP.XAE)
- **PLC library:** Tc3_OCPP (included in the installation of TF6771.IotOCPP.XAE)

TF6771 IoT OCPP is only installed on the engineering. As soon as a project is loaded onto the target system, the TMX driver is copied with the project files.

Use cases

Four categories of use cases can be mapped using TF6771. This includes use as an OCPP client, use as an OCPP server, implementation of an OCPP gateway and enabling an OCPP retrofit.

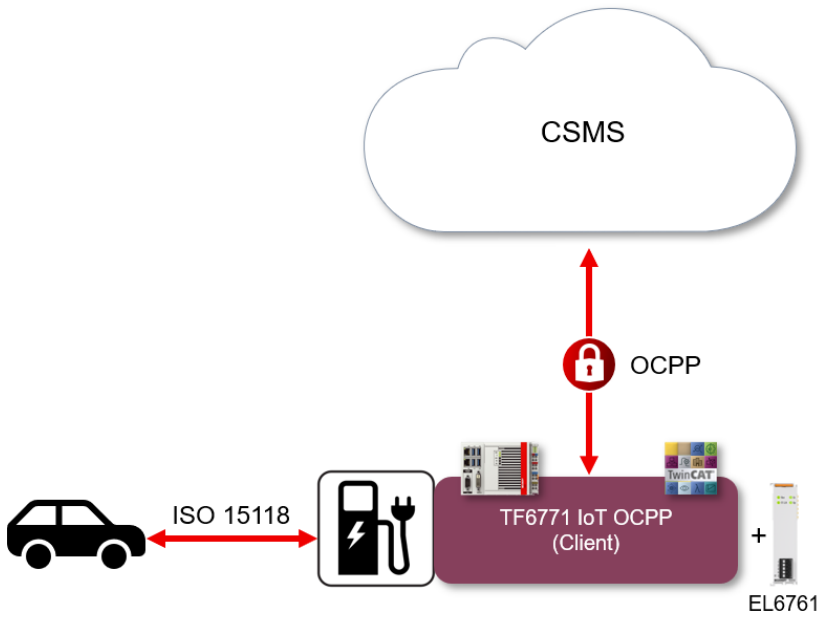
Terminology

The two versions of the OCPP protocol considered in this documentation use different terms.

- In version 1.6, the name for the charging station is **Charge Point**, whereas in version 2.0.1 the name is **Charging Station**.
- In version 1.6, the name for the higher-level management system is **Central System**, while in version 2.0.1 the name is **Charging Station Management System (CSMS)**.
- For ease of understanding, the terms from OCPP version 2.0.1 are used in the following description of the use cases and the associated graphics.

Use case 1: TwinCAT Charging Station

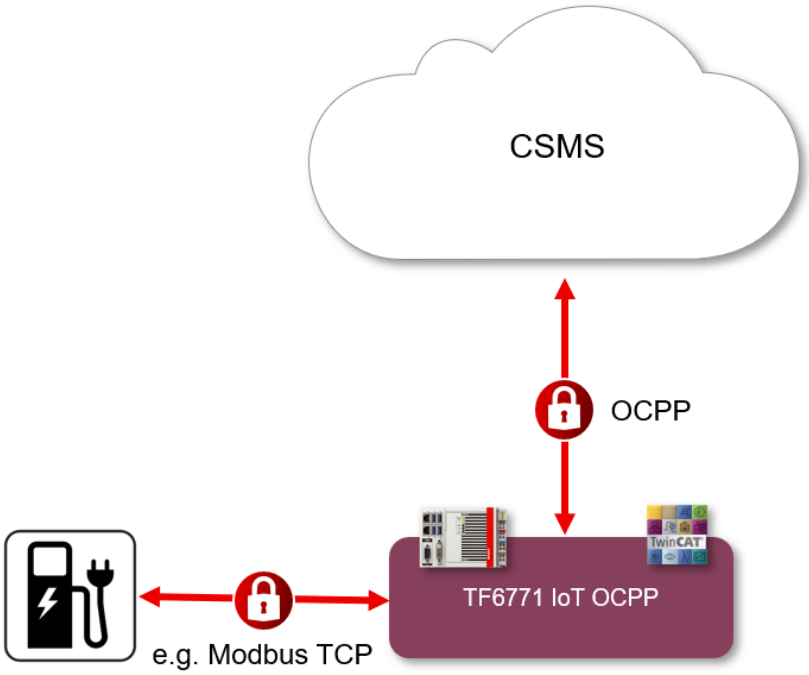
In the first use case, a Charging Station is automated with a Beckhoff control system. In addition to the TF6771 software, the EL6761 EtherCAT Terminal is used for this purpose. The following graphic illustrates use in conjunction with the EL6761.



On the software side, the Charging Station automated with TwinCAT is connected to a CSMS as an OCPP client. Communication in the direction of the vehicle to be charged is realized via the EL6761.

Use case 2: OCPP retrofit

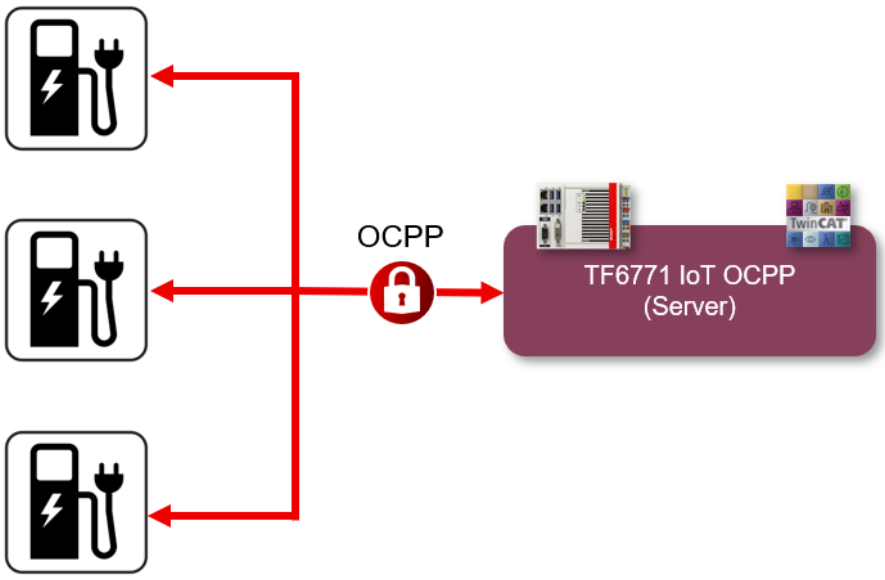
In the second use case, existing Charging Stations without OCPP support are retrofitted for OCPP communication.



In existing charging infrastructure, there may be Charging Stations that do not provide their information via OCPP. For example, there are Charging Stations that distribute information via Modbus TCP. This information can then be collected within TwinCAT and sent to the CSMS on behalf of the Charging Station. To do this, TwinCAT acts as an OCPP client and pretends to be the Charging Station that is to be retrofitted.

Use case 3: Local CSMS

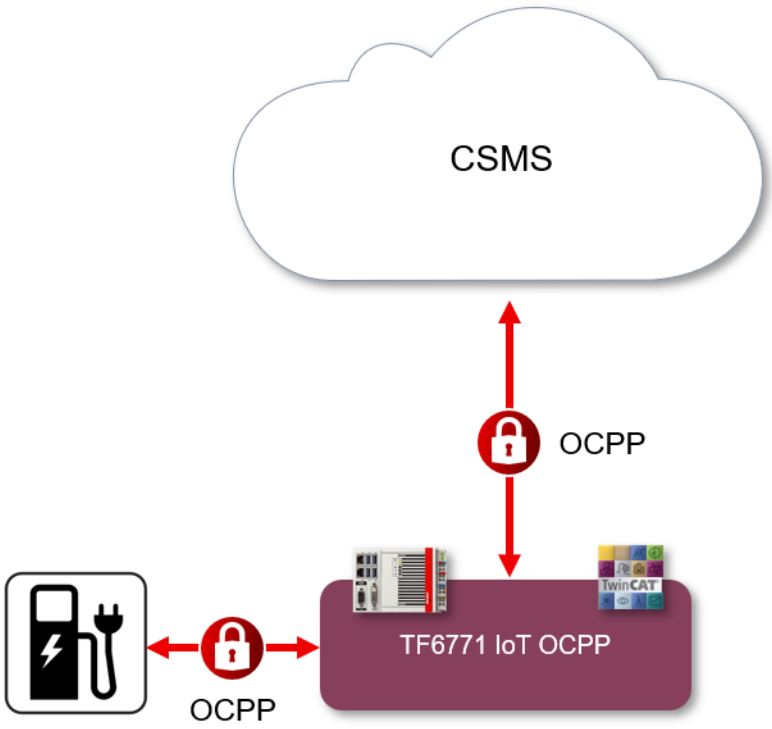
TwinCAT can also be used to implement a local CSMS. Here TwinCAT acts as an OCPP server to which any number of Charging Stations can connect.



At this point, all CSMS functions must be implemented locally. In addition to defining charging profiles, this also includes authentication and billing - if these functions are required.

Use case 4: OCPP gateway

The implementation of an OCPP gateway is also possible. Here TwinCAT acts both as an OCPP client and as an OCPP server.



In such cases, the higher-level CSMS, which is cloud-based in most cases, is often used for authentication and billing. In this case, TwinCAT behaves mainly as a communication switch and forwards messages from the OCPP clients to the higher-level CSMS and vice versa. However, it is also possible to implement local load management via the local OCPP server in the OCPP gateway. Among other things, this can be advantageous due to the information available locally in the control system about the power grid at a location.

3 Installation

3.1 System requirements

Technical data	Description
Operating system	Windows 7/10, Windows Embedded Standard 7, TwinCAT/BSD
Target platform	PC architecture (x86, x64 or ARM)
TwinCAT Version	TwinCAT 3.1 Build 4026.3 or higher
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	TF6771 TC3 IoT OCPP

3.2 Installation

TwinCAT Package Manager

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, you can install this function via the TwinCAT Package Manager, see Installation documentation.

Normally you install the function via the corresponding workload; however, you can also install the packages contained in the workload individually. This documentation briefly describes the installation process via the workload.

Command line program TcPkg

You can use the TcPkg **C**ommand **L**ine **I**nterface (CLI) to display the available workloads on the system:

```
tcpkg list -t workload
```

You can use the following command to install the workload of the TF6771 Iot OCPP Function.

```
tcpkg install TF6771.IotOCPP.XAE
```

TwinCAT Package Manager UI

You can use the **U**ser **I**nterface (UI) to display all available workloads and install them if required.
To do this, follow the corresponding instructions in the interface.

3.3 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

Licensing the full version of a TwinCAT 3 Function

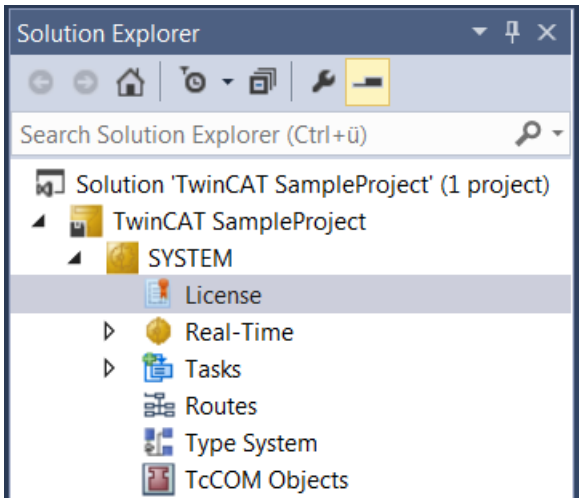
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

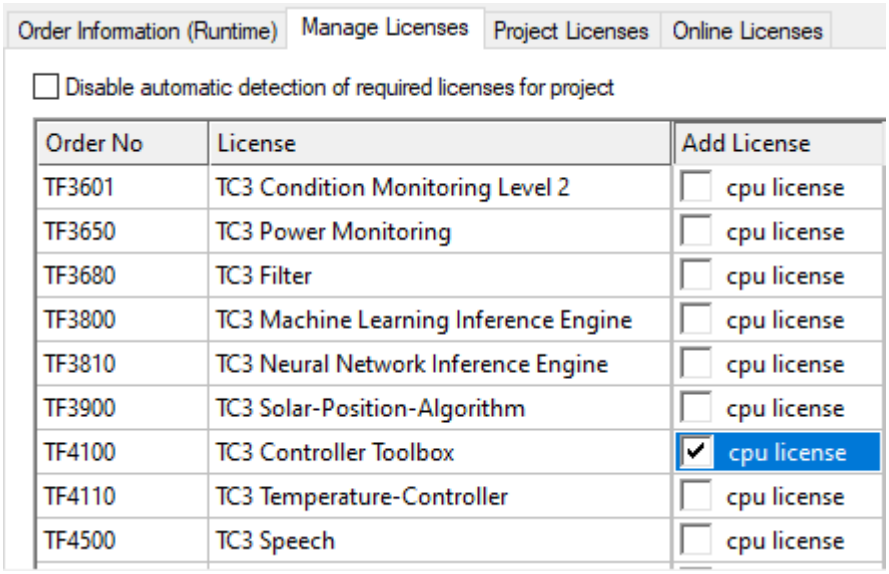
1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



⇨ The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").

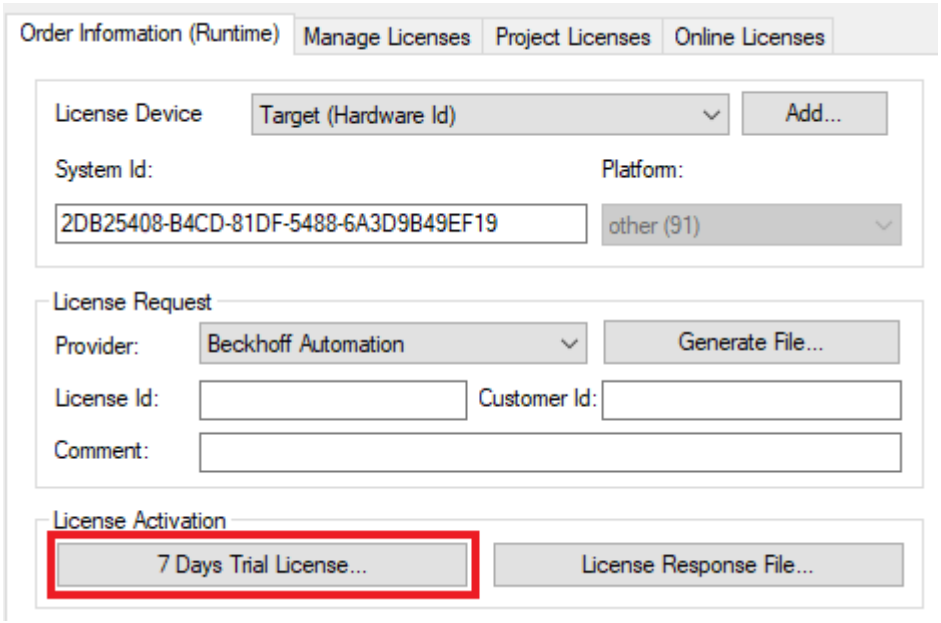


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

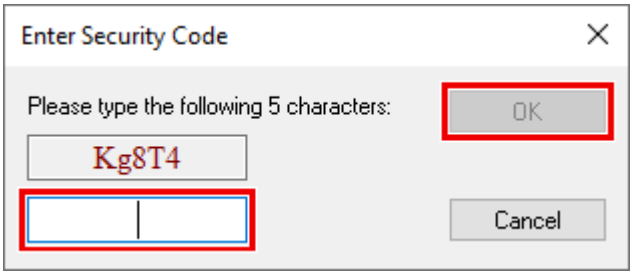
6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 Technical introduction

4.1 OCPP

The Open Charge Point Protocol (OCPP) is a standardized communication protocol that is used between charging stations for electric vehicles (EVSEs) and a central management system (CSMS). It is an open and vendor-independent standard that enables broad interoperability between different vendors of charging stations and the associated management system.

Technically, the OCPP protocol establishes a WebSockets connection between the OCPP client and the OCPP server. The protocol stipulates that an OCPP request can originate from both the client and the server.

4.2 Terminology

The following is a brief explanation of some terms that are important for understanding this documentation. For a more detailed understanding of the OCPP protocol and an explanation of other terms from the OCPP protocol, the OCPP specification must be consulted.

The first table explains the most important terms for a basic understanding of the components:

Term	Explanation
Charging Station (2.0.1)	The Charging Station is the physical system at which an electric vehicle can be charged. A Charging Station has one or more EVSEs. A Charging Station has the name Charge Point in OCPP version 1.6.
Central System (1.6)	Term for the central management system. This term is replaced by CSMS in version 2.0.1 of the OCPP specification.
Charge Point (1.6)	The Charge Point is the physical system at which an electric vehicle can be charged. A Charge Point has one or more Connectors. A Charge Point has the name Charging Station in OCPP version 2.0.1.
Connector	According to the OCPP specification, a Connector is an independently operated and managed socket at a Charge Point.
CSMS (Charging Station Management System) (2.0.1)	Term for the central management system. This term replaces the term Central System from version 1.6 of the OCPP specification.
EVSE (2.0.1)	An EVSE is considered an independently operated and managed part of the Charging Station that can supply energy to one vehicle at a time.

The second table contains further explanations of terms relating to the OCPP protocol:

Term	Explanation
Charging Profile	Generic Charging Profile that is used for different types of profiles. Contains information about the profile and contains the Charging Schedule.
Charging Schedule	Part of a Charging Profile. Defines a block of limit values for the charging power or the charging current. Can contain a start time and a length.
Charging Schedule Period	Specific time period in a Charging Schedule. A Charging Schedule contains one or more Charging Schedule Periods.
Configuration Key	Configurable setting in a Charge Point/Charging Station.
Local Authorization List	The Local Authorization List enables the Charge Point/Charging Station to authorize charging processes even if it is temporarily unable to communicate with the Central System/CSMS.
MeterValue	Container for one or more SampledValues at a specific point in time.
SampledValue	Single sampled value that can contain various data.
Transaction (1.6)	The part of the charging process that begins when all relevant preconditions (e.g. authorization, plug inserted) are met. It ends the moment the charge point leaves this state irrevocably.
Transaction (2.0.1)	A transaction in OCPP is a part of the overall charging process of an electric vehicle that starts and ends based on configurable parameters. These configurable parameters relate to moments in the charging process, such as connecting the electric vehicle or the authorization of the driver.
Unknown Offline Authorization	The Charge Point/Charging Station has the option of admitting unknown users in an offline situation and checking their authorization afterwards.

4.3 Supported functions

Supported versions and data formats

The OCPP protocol in version 1.6 provides for either a SOAP-based data format or a JSON-based data format. In version 2.0.1 of the protocol, only the JSON variant is supported. The following table shows the support in this implementation of the TF6771.

OCPP version	Support?
OCPP1.6S (SOAP)	No
OCPP1.6J (JSON)	Yes
OCPP2.0.1J (JSON)	To follow

4.4 Quick Start

The following chapter provides a quick start to the TwinCAT 3 IoT OCPP function. In these instructions, you will set up an OCPP client and an OCPP server and allow them to communicate with each other in a simplified form. Further functionalities can be found in the other sample projects.

Exemplary implementation

For reasons of simplicity, security is not taken into account in this Quick Start. In real applications, secure implementation should always be a central component of the considerations.

Initialization OCPP client

During initialization, either the function block can be used directly or an existing TcCOM object can be linked. In this Getting Started, initialization is carried out via the function block.

```
    // Client param
    stClientParam      : ST_OCPP1_ClientParam :=(sHost:='127.0.0.1', nPort:=8080, sIdentity:='TestId
ent', sPath:='ocpp', eDebugLevel:=E_OCPP_DebugLevel.MessageLogFile, eTraceLevel:=TcTraceLevel.tlVerb
ose);
    fbClient           : FB_OCPP1_Client(stClientParam);
    eAction            : E_OCPP1_Action;
    hMessageId         : T_OCPP_MessageId;
```

In this case, the OCPP client connects to the OCPP server also running on the same device at port 8080. For demonstration purposes, both the logfile is created and the Trace Level is increased. This is not absolutely necessary in normal operation. Details on logging can be found in the chapter Logging [▶ 19].

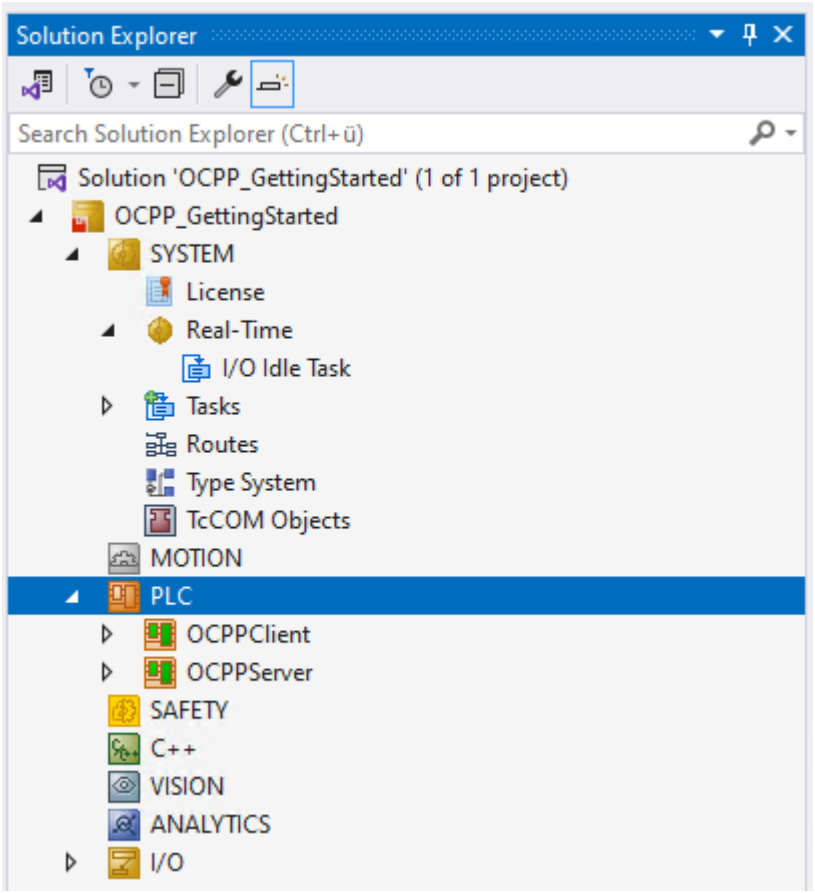
Initialization OCPP server

```
    // Server param
    stServerParam      : ST_OCPP1_ServerParam :=
                          (nPort:=8080,
                           eAuthMode:=E_OCPP1_AuthenticationMode.None,
                           eEncryptionMode := E_OCPP_EncryptionMode.None,
                           eDebugLevel:=E_OCPP_DebugLevel.MessageLogFile,
                           eTraceLevel:=TcTraceLevel.tlVerbose);
    fbOcppServer       : FB_OCPP1_Server(stServerParam);
    eAction            : E_OCPP1_Action;
    hMessageId         : T_OCPP_MessageId;
    hStationId         : UDINT;
```

This OCPP server also runs on the local device, on port 8080. The logfile and Trace Level are also set here for demonstration purposes.

Basic structure sample project

In this Getting Started, a TwinCAT project is used in which both the OCPP client and the OCPP server are operated in a separate PLC project. The actions are initiated by the OCPP client. In a real scenario, actions are of course also triggered on the server side.



Implementation OCPP client

The sample includes two different processes on the client side. On the one hand, a simple StatusNotification can be sent:

```
// Send StatusNotification
IF bSendStatus THEN
    bSendStatus:=FALSE;
    eError:=E_OCPP1_ChargePointError.HighTemperature;
    eStatus:=E_OCPP1_ChargePointStatus.SuspendedEV;
    eAction:=E_OCPP1_Action.StatusNotification;
END_IF
```

The values sent with the StatusNotification can be adjusted using the eError and eStatus parameters. On the other hand, a streamlined process for simulated charging is implemented:

```
// Charging sample
CASE nState OF
    0:
        IF bStartCharging THEN
            bStartCharging:=FALSE;
            eAction:=E_OCPP1_Action.Authorize;
            nState:=nState+1;
        END_IF
    1:
        IF bAuthorized THEN
            bAuthorized:=FALSE;
            nMeterStart:=nMeterStop;
            eAction:=E_OCPP1_Action.StartTransaction;
            nState:=nState+1;
        END_IF
    2:
        IF bChargingStarted THEN
            //10 seconds charging simulation
            timerCharge(IN:=TRUE);
            IF timerCharge.Q THEN
                bChargingStarted:=FALSE;
                timerCharge(IN:=FALSE);
                nMeterStop:=nMeterStart+10;
                eAction:=E_OCPP1_Action.StopTransaction;
                nState:=nState+1;
            END_IF
        END_IF
    3:
        IF bChargingStopped THEN
            bChargingStopped:=FALSE;
            nState:=0;
        END_IF
END_CASE
```

If the Charging Sample is started via bStartCharging, the client first calls the Authorize method. Following a successful Authorize, the StartTransaction method is then executed. After a successful StartTransaction, the system "loads" for 10 seconds and then executes a StopTransaction.

The specific OCPP communication is implemented via a CASE statement. The individual OCPP commands are implemented in this CASE statement:

```
// Different OCPP commands implemented for OCPP client CASE eAction OF
    E_OCPP1_Action.None:
        IF NOT fbClient.bError AND fbClient.bValid THEN
            fbClient.PollRequest(hMessageId=> hMessageId, eAction=> eAction );
        ELSE
            // Implement error handling here
        END_IF
    E_OCPP1_Action.Authorize:
        IF fbClient.SendAuthorize(sIdTag:=sIdTag, eStatus => stIdTagInfo.eStatus) THEN
            IF stIdTagInfo.eStatus = E_OCPP1_AuthorizationStatus.Accepted THEN
                bAuthorized := TRUE;
                eAction    := E_OCPP1_Action.None;
            END_IF
        END_IF
    E_OCPP1_Action.StartTransaction:
        IF fbClient.SendStartTransaction(sIdTag:=sIdTag, nConnectorId:=nConnectorId, nMeterStart:=nM
eterStart, eStatus => stIdTagInfo.eStatus, nTransactionId => nTransactionId) THEN
            IF stIdTagInfo.eStatus = E_OCPP1_AuthorizationStatus.Accepted THEN
                bChargingStarted := TRUE;
                eAction    := E_OCPP1_Action.None;
            END_IF
        END_IF
```

```
    E_OCPP1_Action.StopTransaction:
        IF fbClient.SendStopTransaction(nConnectorId:=nConnectorId, nMeterStop:=nMeterStop, eStatus
=> stIdTagInfo.eStatus) THEN
            bChargingStopped := TRUE;
            eAction    := E_OCPP1_Action.None;
        END_IF
    E_OCPP1_Action.StatusNotification:
        IF fbClient.SendStatusNotification(nConnectorId:=nConnectorId, eError, eStatus) THEN
            eAction    := E_OCPP1_Action.None;
        END_IF
    // Implement here more functionalities like:
    // E_OCPP1_Action.TriggerMessage:
ELSE
    eAction := E_OCPP1_Action.None;
    // not implemented...
END_CASE
```

Only the functions used for the sample are implemented here. Further functions would mean an extension of the CASE statement.

Implementation OCPP server

On the OCPP server side, the commands sent by the client are received and the appropriate response is sent:

```
CASE eAction OF
    E_OCPP1_Action.None:
        IF NOT fbOcppServer.bError AND fbOcppServer.bValid THEN
            fbOcppServer.PollRequest(eAction => eAction, hMessageId => hMessageId, hStationId => hSt
ationId);
        ELSE
            // Implement error handling here
        END_IF
    E_OCPP1_Action.BootNotification:
        IF fbOcppServer.RecvBootNotification(hStationId, hMessageId=>hMessageId,
            sModel                => stBootNotification.sModel,
            sVendor                 => stBootNotification.sVendor,
            sChargeBoxSerial    => stBootNotification.sChargeBoxSerial,
            sChargePointSerial    => stBootNotification.sChargePointSerial,
            sFirmwareVersion    => stBootNotification.sFirmwareVersion,
            sMeterSerial        => stBootNotification.sMeterSerial,
            sMeterType            => stBootNotification.sMeterType)
        THEN
            fbOcppServer.RespBootNotification(hStationId, hMessageId, E_OCPP1_RegistrationStatus.Acc
epted);
            eAction := E_OCPP1_Action.None;
        END_IF
    E_OCPP1_Action.Heartbeat:
        IF fbOcppServer.RecvHeartbeat(hStationId) THEN
            fbOcppServer.RespHeartbeat(hStationId:=hStationId,hMessageId:=hMessageId);
            eAction := E_OCPP1_Action.None;
        END_IF
    E_OCPP1_Action.Authorize:
        IF fbOcppServer.RecvAuthorize(hStationId, sIdTag => sIdTag) THEN
            IF sIdTag='Test123' THEN
                fbOcppServer.RespAuthorize(hStationId:=hStationId, hMessageId:=hMessageId, E_OCPP1_A
uthorizationStatus.Accepted);
                eAction := E_OCPP1_Action.None;
            END_IF
        END_IF
    E_OCPP1_Action.StartTransaction:
        IF fbOcppServer.RecvStartTransaction(hStationId, hMessageId => hMessageId, sIdTag => sIdTag,
 nConnectorId => nConnectorId, nMeterStart => nMeterStart, nTimestamp => nTimestamp) THEN
            nTransactionId:=nTransactionId+1;
            fbOcppServer.RespStartTransaction(hStationId, hMessageId, E_OCPP1_AuthorizationStatus.Ac
cepted, nTransactionId:=nTransactionId);
            eAction := E_OCPP1_Action.None;
        END_IF
    E_OCPP1_Action.StopTransaction:
        IF fbOcppServer.RecvStopTransaction(hStationId, hMessageId => hMessageId, sIdTag => sIdTag,
nTransactionId => nTransactionIdRecv, nConnectorId => nConnectorId, nMeterStop => nMeterStop, nTimes
tamp => nTimestamp, eReason => eReason) THEN
            fbOcppServer.RespStopTransaction(hStationId, hMessageId, E_OCPP1_AuthorizationStatus.Acc
epted);
            eAction := E_OCPP1_Action.None;
        END_IF
    E_OCPP1_Action.StatusNotification:
        IF fbOcppServer.RecvStatusNotification(hStationId, hMessageId => hMessageId, nConnectorId =>
```

```
 nConnectorIdStatus, eError => eError, eStatus => eStatus) THEN
            fbOcppServer.RespStatusNotification(hStationId, hMessageId);
            eAction := E_OCPP1_Action.None;
        END_IF
END_CASE
```

Compared to the client sample, two more functions are implemented here. The Heartbeat is sent internally by the client and can be configured via a Property on the function block. The BootNotification is sent automatically when the client starts up and must therefore also be processed in the server.

OCPP messages in server and client can also be implemented in the other direction. The server can send messages to the client. Conversely, the client can then receive these messages from the server.

The logfile looks as follows after the client sample has been executed once (sending a StatusNotification and executing the charging sample):

TcIotOcppClient.log | TcIotOcppServer.log

```
TcIotOcppClient:Send: [2,"00000001","BootNotification",{"chargePointModel":"TcIotOcpp","chargePointVendor":"Beckhoff Automation"}]
TcIotOcppClient:Recv: [3,"00000001",{"status":"Accepted","currentTime":"2024-07-31T14:57:41.67Z","interval":300}]
TcIotOcppClient:Send: [2,"00000002","Authorize",{"idTag":"Test123"}]
TcIotOcppClient:Recv: [3,"00000002",{"idTagInfo":{"status":"Accepted"}}]
TcIotOcppClient:Send: [2,"00000003","StartTransaction",{"connectorId":0,"idTag":"Test123","meterStart":0,"timestamp":"2024-07-31T14:57:46.948Z"}]
TcIotOcppClient:Recv: [3,"00000003",{"idTagInfo":{"status":"Accepted"},"transactionId":1}]
TcIotOcppClient:Send: [2,"00000004","StopTransaction",{"meterStop":10,"timestamp":"2024-07-31T14:57:56.988Z","transactionId":1}]
TcIotOcppClient:Recv: [3,"00000004",{"idTagInfo":{"status":"Accepted"}}]
TcIotOcppClient:Send: [2,"00000005","StatusNotification",{"connectorId":0,"errorCode":"HighTemperature","status":"SuspendedEV","timestamp":"2024-07-31T14:57:59.438Z"}]
TcIotOcppClient:Recv: [3,"00000005",{}]
TcIotOcppClient:Send: [2,"00000006","Heartbeat",{}]
TcIotOcppClient:Recv: [3,"00000006",{"currentTime":"2024-07-31T14:58:34.61Z"}]
```

TcIotOcppClient.log | TcIotOcppServer.log

```
TcIotOcppServer:Recv:1: [2,"00000001","BootNotification",{"chargePointModel":"TcIotOcpp","chargePointVendor":"Beckhoff Automation"}]
TcIotOcppServer:Send:1: [3,"00000001",{"status":"Accepted","currentTime":"2024-07-31T14:57:41.67Z","interval":300}]
TcIotOcppServer:Recv:1: [2,"00000002","Authorize",{"idTag":"Test123"}]
TcIotOcppServer:Send:1: [3,"00000002",{"idTagInfo":{"status":"Accepted"}}]
TcIotOcppServer:Recv:1: [2,"00000003","StartTransaction",{"connectorId":0,"idTag":"Test123","meterStart":0,"timestamp":"2024-07-31T14:57:46.948Z"}]
TcIotOcppServer:Send:1: [3,"00000003",{"idTagInfo":{"status":"Accepted"},"transactionId":1}]
TcIotOcppServer:Recv:1: [2,"00000004","StopTransaction",{"meterStop":10,"timestamp":"2024-07-31T14:57:56.988Z","transactionId":1}]
TcIotOcppServer:Send:1: [3,"00000004",{"idTagInfo":{"status":"Accepted"}}]
TcIotOcppServer:Recv:1: [2,"00000005","StatusNotification",{"connectorId":0,"errorCode":"HighTemperature","status":"SuspendedEV","timestamp":"2024-07-31T14:57:59.438Z"}]
TcIotOcppServer:Send:1: [3,"00000005",{}]
TcIotOcppServer:Recv:1: [2,"00000006","Heartbeat",{}]
TcIotOcppServer:Send:1: [3,"00000006",{"currentTime":"2024-07-31T14:58:34.61Z"}]
```

4.5 Logging

Various logging functions are available to enable extended diagnostics.

TwinCAT Output

The first point to consider is the TwinCAT Output. Information is written here depending on the Trace Level. In the event of an error, you can first look there for a message corresponding to the error:

Output

Show output from: TwinCAT

```
MSG | 7/25/2024 11:20:50 PM 248 ms | 'TCOM Server' (10): TcIotOcppClient: Demo License valid.
MSG | 7/25/2024 11:20:50 PM 254 ms | 'TCOM Server' (10): TcIotOcppClient: State: SAFEOP
MSG | 7/25/2024 11:20:50 PM 254 ms | 'TCOM Server' (10): TcIotOcppClient: 0.1.8.2
MSG | 7/25/2024 11:20:50 PM 254 ms | 'TCOM Server' (10): TcIotOcppClient: State: OP
MSG | 7/25/2024 11:20:50 PM 267 ms | 'TCOM Server' (10): TcIotOcppClient: Connecting... HR=0x00000203
MSG | 7/25/2024 11:20:50 PM 388 ms | 'TCOM Server' (10): TcIotOcppServer: Demo License valid.
MSG | 7/25/2024 11:20:50 PM 396 ms | 'TCOM Server' (10): TcIotOcppServer: State: SAFEOP
MSG | 7/25/2024 11:20:50 PM 396 ms | 'TCOM Server' (10): TcIotOcppServer: 0.1.8.2
MSG | 7/25/2024 11:20:50 PM 396 ms | 'TCOM Server' (10): TcIotOcppServer: State: OP
MSG | 7/25/2024 11:20:50 PM 408 ms | 'TCOM Server' (10): TcIotOcppServer: Listening... HR=0x00000203
MSG | 7/25/2024 11:20:50 PM 411 ms | 'TCOM Server' (10): TcIotOcppServer: Server: Open
MSG | 7/25/2024 11:20:50 PM 775 ms | 'TCOM Server' (10): TcIotOcppServer: Server: Socket accepted. Addr="127.0.0.1":38336
MSG | 7/25/2024 11:20:50 PM 782 ms | 'TCOM Server' (10): TcIotOcppServer: Station(1): Opened
MSG | 7/25/2024 11:20:50 PM 783 ms | 'TCOM Server' (10): TcIotOcppServer: Station(1): Connected
MSG | 7/25/2024 11:20:50 PM 784 ms | 'TCOM Server' (10): TcIotOcppClient: Station(0): Opened
MSG | 7/25/2024 11:20:50 PM 784 ms | 'TCOM Server' (10): TcIotOcppClient: WebSocket: Connected
MSG | 7/25/2024 11:20:50 PM 784 ms | 'TCOM Server' (10): TcIotOcppClient: Connected. HR=0x00000000
MSG | 7/25/2024 11:20:50 PM 785 ms | 'TCOM Server' (10): TcIotOcppClient: Booting...
MSG | 7/25/2024 11:20:50 PM 810 ms | 'TCOM Server' (10): TcIotOcppClient: Boot notification accepted.
MSG | 7/25/2024 11:20:50 PM 810 ms | 'TCOM Server' (10): TcIotOcppClient: Booted. HR=0x00000000
MSG | 7/25/2024 11:21:39 PM 588 ms | 'TCOM Server' (10): TcIotOcppClient: Received heartbeat result. OffsetError=0.001s
MSG | 7/25/2024 11:22:09 PM 589 ms | 'TCOM Server' (10): TcIotOcppClient: Received heartbeat result. OffsetError=0.000s
MSG | 7/25/2024 11:22:39 PM 579 ms | 'TCOM Server' (10): TcIotOcppClient: Received heartbeat result. OffsetError=0.000s
```

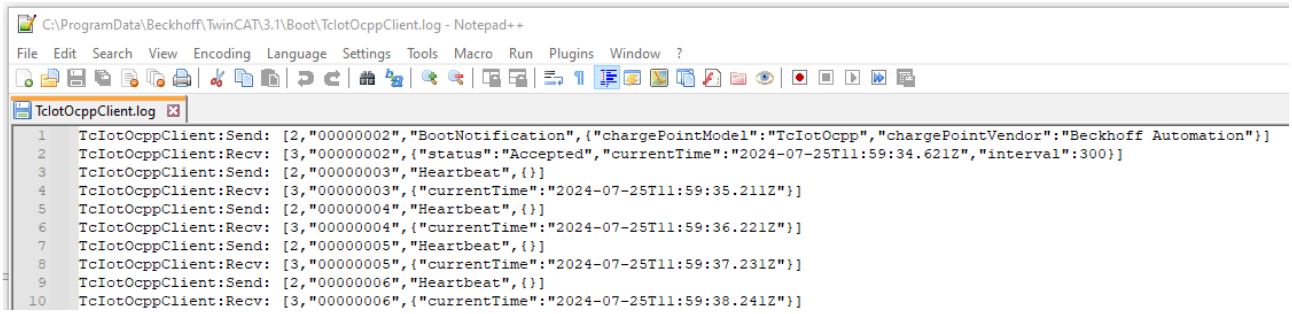
The Trace Level can be adjusted via the structures ST_OCPP1_Client_Param [▶ 199] and ST_OCPP1_Server_Param [▶ 201]. This is not only possible at the start of the application, but also at runtime. To do this, the value must be changed and the respective Init method must be called by the client or server. The following table shows the different TcTraceLevels and their effects:

TcTraceLevel	Effect
tlAlways	No messages are output via the TwinCAT Output.
tlError	Only error messages are output.
tlWarning	Warnings and error messages are output.
tlInfo	One-time status information, warnings, and error messages are output.
tlVerbose	All messages are output. This includes cyclic information, one-time status information, warnings and error messages.

Logfile

The second diagnostic option is to write a logfile. There are only two different options here, writing the logfile is either enabled or disabled. The Trace Level can be adjusted via the structures ST_OCPP1_Client_Param [▶ 199] and ST_OCPP1_Server_Param [▶ 201].

The OCPP messages are logged in the logfile. The following shows an example of an activated logfile in the OCPP client.



The logfiles are stored in the TwinCAT boot directory:

C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot

Use of the logfile for debugging purposes only

The logfile should only be used for debugging purposes. All OCPP messages are logged in the file so that large logfile sizes can be created over long periods of time.

5 PLC API

5.1 POUs

5.1.1 FB_OCPP1_Client

This function block represents an OCPP client that can be connected to an OCPP server. There are two directions of communication.

The send methods are used when sending requests to the server. In these methods, the response from the server is processed directly and stored in the output parameters of the methods.

If, on the other hand, a request is received from the server using one of the Receive methods, the response must be sent using the appropriate Response method.

The client sends an internal Heartbeat, the time interval of which can be adjusted using the HeartbeatInterval Property. If the client does not receive a response from the server to a Heartbeat, a Reconnect to the server is then attempted. For all other message types, a timeout error is output on the client function block if there is no response.

Syntax

```
FUNCTION BLOCK FB_OCPP1_Client
VAR_OUTPUT
    bValid              : BOOL;
    bBusy               : BOOL;
    bError              : BOOL;
    eErrorResult        : HRESULT;
    eErrorAction        : E_OCPP1_Action;
END_VAR
```

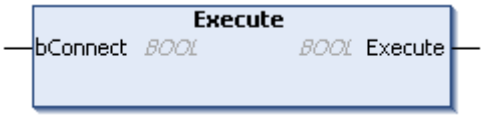
Outputs

Name	Type	Description
bValid	BOOL	The interface to the driver in the background exists.
bBusy	BOOL	Is TRUE as long as the function block is busy with processing.
bError	BOOL	Becomes TRUE when an error situation occurs.
eErrorResult	HRESULT	Last error present on the function block.
eErrorAction	E_OCPP1_Action [▶ 189]	OCPP command for which the error occurred.

Properties

Name	Type	Access	Description
HeartbeatInterval	TIME	Get, Set	Internal HeartbeatInterval from the client. Another Heartbeat can also be sent manually using the SendHeartbeat [▶ 69] method. With the internal HeartbeatInterval, it should be noted that the server responds in its BootNotification.conf with a specified HeartbeatInterval for the client. This specified HeartbeatInterval is then set internally. However, the HeartbeatInterval can then be changed as required during runtime.
IsConnected	BOOL	Get	Status of the connection to the OCPP server.
IsPending	BOOL	Get	Waiting for completion of the request.

5.1.1.1 Execute



This method can be used to establish a connection or to disconnect a connection. No permanent call is required, only changes are transferred from bConnect to the driver.

If a connection exists, FALSE is used to disconnect the connection; if a connection does not exist, TRUE is used to establish it.

Syntax

```
METHOD Execute : BOOL
VAR_INPUT
    bConnect : BOOL;
END_VAR
```

Return value

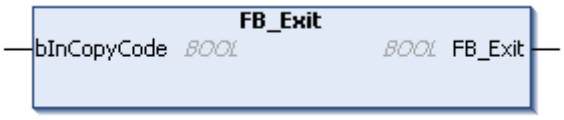
Name	Type	Description
Execute	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bConnect	BOOL	If a connection exists, FALSE is used to disconnect the connection; if a connection does not exist, TRUE is used to establish it.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.2 FB_Exit



Syntax

```
METHOD FB_Exit : BOOL
VAR_INPUT
    bInCopyCode : BOOL;
END_VAR
```

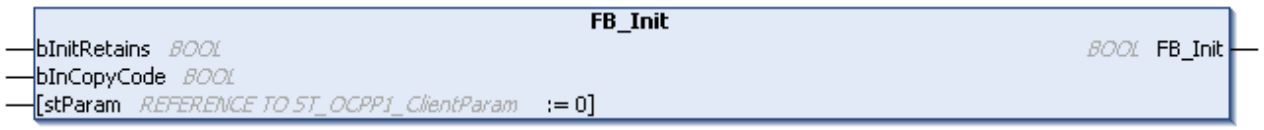
Return value

Name	Type	Description
FB_Exit	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bInCopyCode	BOOL	If TRUE, the Exit method is called to exit an instance, which is then copied (online change).

5.1.1.3 FB_Init



Syntax

```
METHOD FB_Init : BOOL
VAR_INPUT
    bInitRetains : BOOL;
    bInCopyCode  : BOOL;
    stParam      : REFERENCE TO ST_OCPP1_Client_Param REF=0;
END_VAR
```

Return value

Name	Type	Description
FB_Init	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bInitRetains	BOOL	If TRUE, the Retain variables are initialized (warm start/cold start).
bInCopyCode	BOOL	If TRUE, the Exit method is called to exit an instance, which is then copied (online change).
stParam	REFERENCE TO ST_OCPP1_Client_Param [▶ 199]	Parameters for the WebSockets connection of the OCPP client.

5.1.1.4 Init

This method is called when the function block is initialized in the FB_Init [▶ 23] method and specifies the parameters of the WebSockets connection. If the connection parameters are to be changed afterwards, this method can be called again while the application is running.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    stParam : REFERENCE TO ST_OCPP1_Client_Param;
END_VAR
```

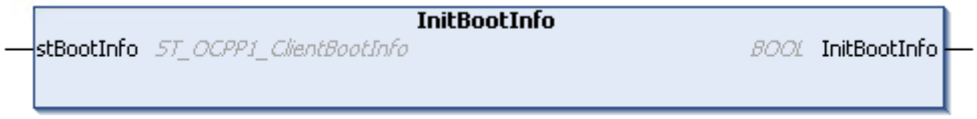
Return value

Name	Type	Description
Init	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
stParam	REFERENCE TO ST_OCPP1_Client_Param [▶ 199]	Parameters for the WebSockets connection of the OCPP client.

5.1.1.5 InitBootInfo



This method can be used to set the Boot Notification sent by the OCPP client when an OCPP connection is established. If this method is not called after starting the application, default values are used. More detailed information can be found in ST_OCPP1_Client_BootInfo [▶ 198].

Syntax

```
METHOD InitBootInfo : BOOL
VAR_IN_OUT CONSTANT
    stBootInfo : ST_OCPP1_Client_BootInfo;
END_VAR
```

Return value

Name	Type	Description
InitBootInfo	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs/outputs

Name	Type	Description
stBootInfo	ST_OCPP1_Client_BootInfo [▶ 198]	BootInfo for the OCPP client.

5.1.1.6 InitOptions

This method can be used to set various options for the OCPP client. If this method is not called after starting the application, default values are used. More detailed information can be found in ST_OCPP1_Client_Options [▶ 199].

Syntax

```
METHOD InitOptions : BOOL
VAR_IN_OUT CONSTANT
    stOptions : ST_OCPP1_Client_Options;
END_VAR
```

Return value

Name	Type	Description
InitOptions	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs/outputs

Name	Type	Description
stOptions	ST_OCPP1_Client_Options [▶ 199]	Options for the OCPP client.

5.1.1.7 InitSettings

This method can be used to make various settings for the OCPP client. If this method is not called after starting the application, default values are used. More detailed information can be found in ST_OCPP1_Client_Settings [▶ 200].

Syntax

```
METHOD InitSettings : BOOL
VAR_IN_OUT CONSTANT
    stSettings : ST_OCPP1_Client_Settings;
END_VAR
```

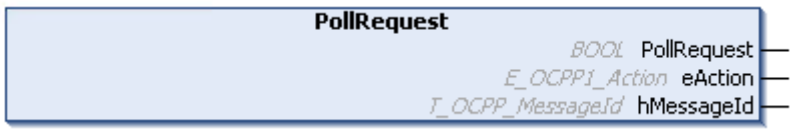
Return value

Name	Type	Description
InitSettings	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs/outputs

Name	Type	Description
stSettings	ST_OCPP1_Client_Settings [▶ 200]	Settings for the OCPP client.

5.1.1.8 PollRequest



This method must be called in order to receive incoming requests from the OCPP server. The outputs of the method return both the message ID of the received message and the information about which OCPP request it is.

Syntax

```
METHOD PollRequest : BOOL
VAR_OUTPUT
    eAction    : E_OCPP1_Action;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
PollRequest	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

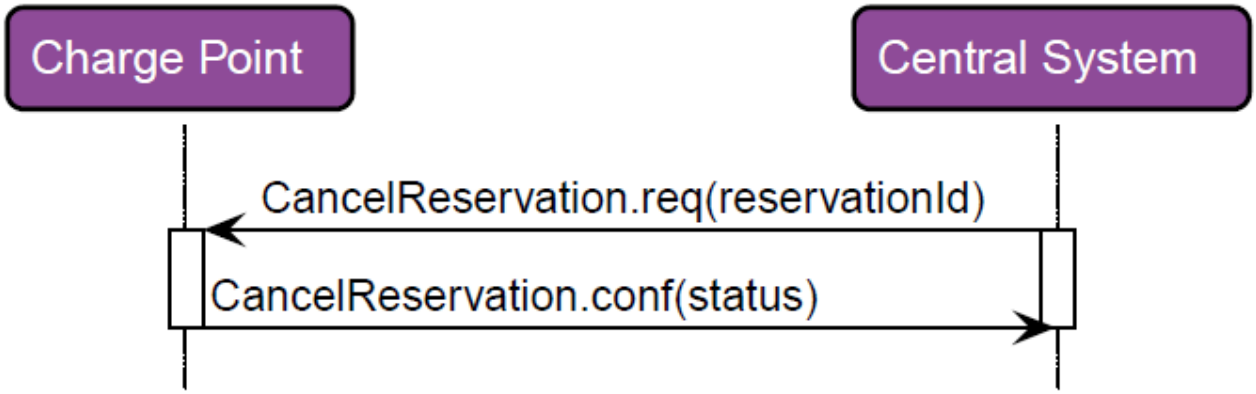
Outputs

Name	Type	Description
eAction	E_OCPP1_Action [▶ 189]	Type of OCPP request received from the server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.9 RecvCancelReservation

With this method, an OCPP client receives a Cancel Reservation request from the corresponding OCPP server. To respond to the request, the method RespCancelReservation [▶ 47] must be called.



Syntax

```
METHOD RecvCancelReservation : BOOL
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    nReservationId : UDINT;
END_VAR
```

Return value

Name	Type	Description
RecvCancelReservation	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

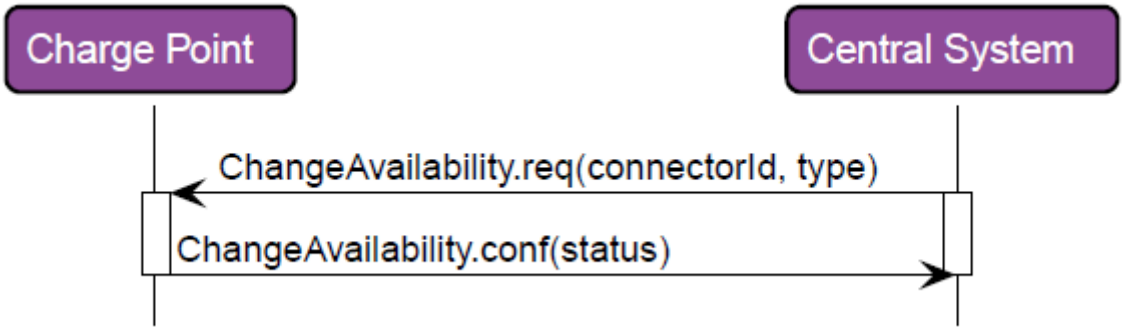
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nReservationId	UDINT	ID of the reservation to be canceled.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.10 RecvChangeAvailability

With this method, an OCPP client receives a Change Availability request from the corresponding OCPP server. To respond to the request, the method RespChangeAvailability [▶ 48] must be called.



Syntax

```
METHOD RecvChangeAvailability : BOOL
VAR_OUTPUT
    hMessageId    : T_OCPP_MessageId;
    nConnectorId : UDINT;
    eType         : E_OCPP1_AvailabilityType;
END_VAR
```

Return value

Name	Type	Description
RecvChangeAvailability	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

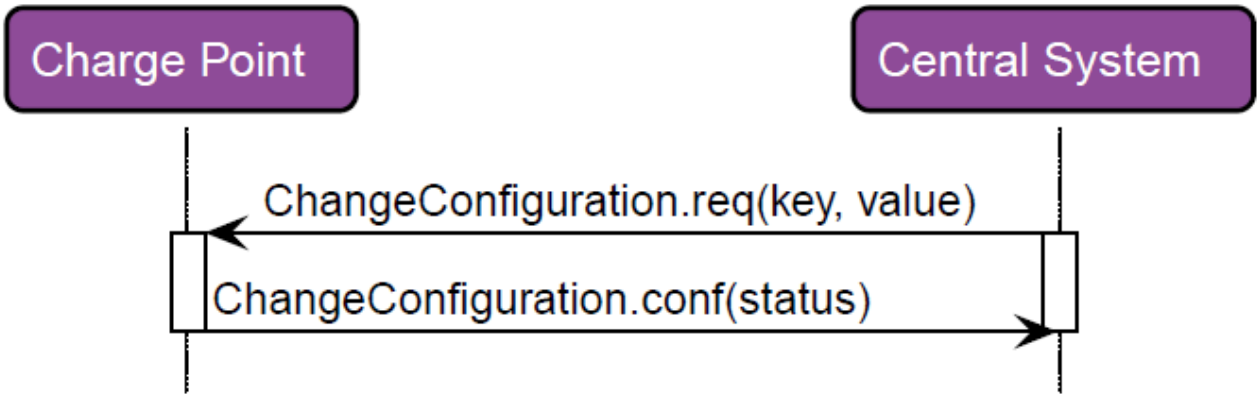
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
eType	E_OCPP1_AvailabilityType [▶ 190]	Type of availability change that the Charge Point is to carry out.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.11 RecvChangeConfiguration

With this method, an OCPP client receives a Change Configuration request from the corresponding OCPP server. To respond to the request, the method RespChangeConfiguration [▶ 49] must be called.



Syntax

```
METHOD RecvChangeConfiguration : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    sKey       : T_OCPP1_ConfigKey;
    sValue     : T_OCPP1_ConfigValue;
END_VAR
```

Return value

Name	Type	Description
RecvChangeConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

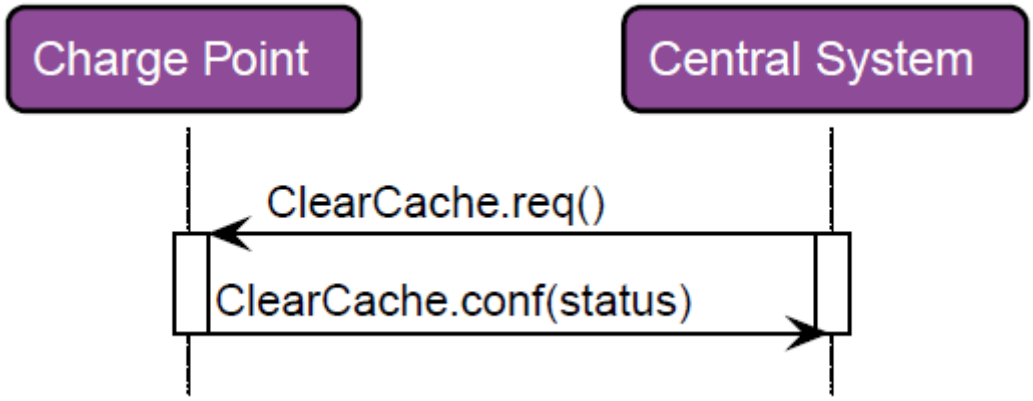
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sKey	T_OCPP1_ConfigKey [▶ 208]	Name of the Configuration Key to be changed.
eType	T_OCPP1_ConfigValue [▶ 208]	New value for the Configuration Key.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.12 RecvClearCache

With this method, an OCPP client receives a Clear Cache request from the corresponding OCPP server. To respond to the request, the method RespClearCache [▶ 50] must be called.



Syntax

```
METHOD RecvClearCache : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

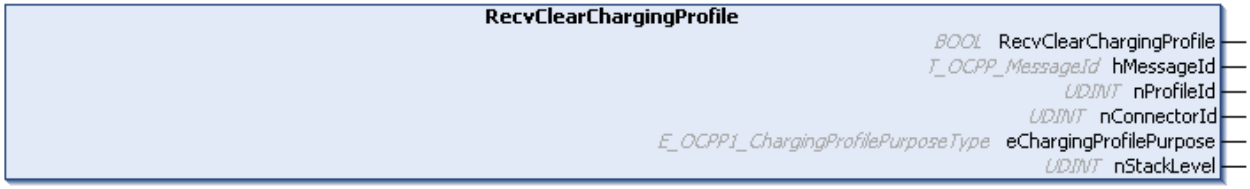
Name	Type	Description
RecvClearCache	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

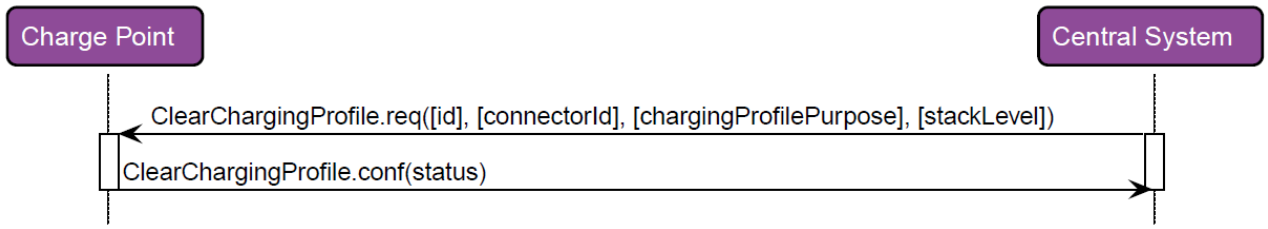
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.13 RecvClearChargingProfile



With this method, an OCPP client receives a Clear Charging Profile request from the corresponding OCPP server. To respond to the request, the method RespClearChargingProfile [▶ 51] must be called.



Syntax

```
METHOD RecvClearChargingProfile : BOOL
VAR_OUTPUT
    hMessageId              : T_OCPP_MessageId;
    nProfileId              : UDINT;
    nConnectorId            : UDINT;
    eChargingProfilePurpose : E_OCPP1_ChargingProfilePurposeType;
    nStackLevel             : UDINT;
END_VAR
```

Return value

Name	Type	Description
RecvClearChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

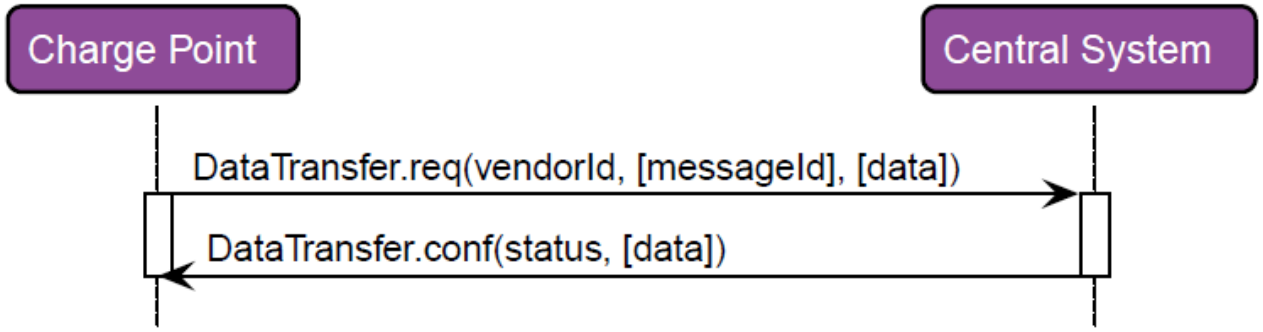
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nProfileId	UDINT	ID of the Charging Profile to be deleted.
nConnectorId	UDINT	ID of the Connector of a Charge Point for which the Charging Profiles are to be deleted.
eChargingProfilePurpose	E_OCPP1_ChargingProfilePurposeType [▶ 191]	Specifies the ChargingProfilePurposeType whose Charging Profiles are to be deleted.
nStackLevel	UDINT	Specifies the Stack Level for which Charging Profiles are to be deleted if all other criteria in the request are met.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.14 RecvDataTransfer

With this method, an OCPP client receives a Data Transfer request from the corresponding OCPP server. To respond to the request, the method RespDataTransfer [▶ 52] must be called.



Syntax

```
METHOD RecvDataTransfer : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    sVendorId  : STRING(255);
    sMessageId : STRING(63);
END_VAR
VAR_IN_OUT
    sData      : T_OCPP1_DataTransferData;
END_VAR
```

Return value

Name	Type	Description
RecvDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sVendorId	STRING(255)	Identifier for the vendor, which identifies the vendor-specific implementation.
sMessageId	STRING(63)	Additional identification field for a single message.

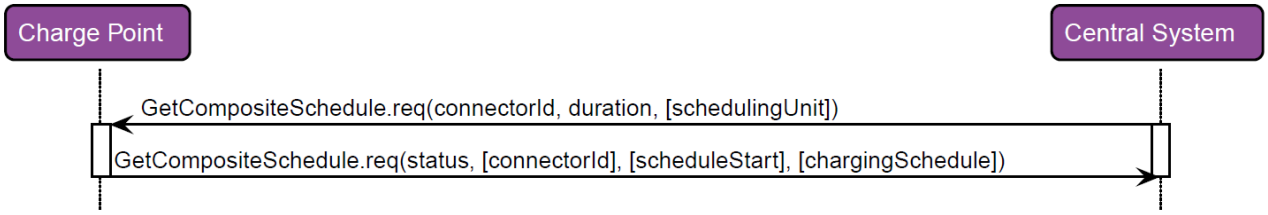
Inputs/outputs

Name	Type	Description
sData	T_OCPP1_DataTransferData [▶ 208]	Text without specified length and format.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.15 RecvGetCompositeSchedule

With this method, an OCPP client receives a Get Composite Schedule request from the corresponding OCPP server. To respond to the request, the method RespGetCompositeSchedule [▶ 53] must be called.



Syntax

```
METHOD RecvGetCompositeSchedule : BOOL
VAR_OUTPUT
    hMessageId        : T_OCPP_MessageId;
    nConnectorId      : UDINT;
    nDuration         : UDINT;
    eChargingRateUnit : E_OCPP1_ChargingRateUnitType;
END_VAR
```

Return value

Name	Type	Description
RecvGetCompositeSchedule	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

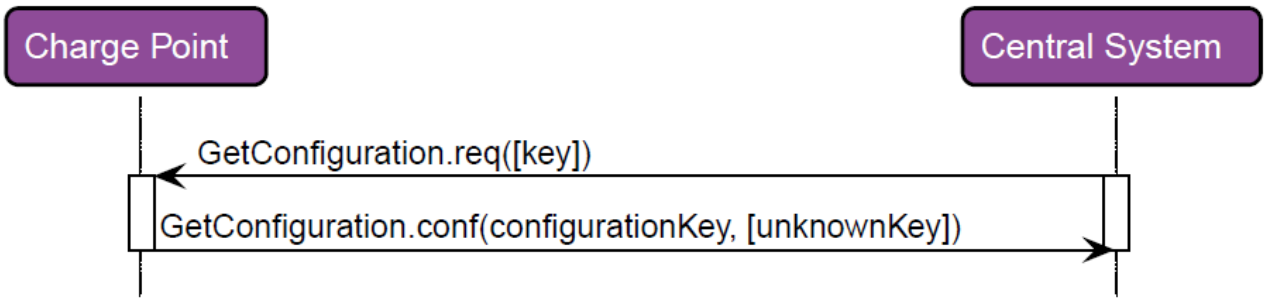
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nDuration	UDINT	Time of the requested schedule in seconds.
eChargingRateUnit	E_OCPP1_ChargingRateUnitType [▶ 192]	Can optionally be used to specify the unit of the requested schedule.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.16 RecvGetConfiguration

With this method, an OCPP client receives a Get Configuration request from the corresponding OCPP server. To respond to the request, the method RespGetConfiguration [▶ 54] must be called.



Syntax

```
METHOD RecvGetConfiguration : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    nKeysCount : UDINT := 0;
    aKeys      : ARRAY[1..Param_OCPP.nConfigKeysMax] OF T_OCPP1_ConfigKey;
END_VAR
```

Return value

Name	Type	Description
RecvGetConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

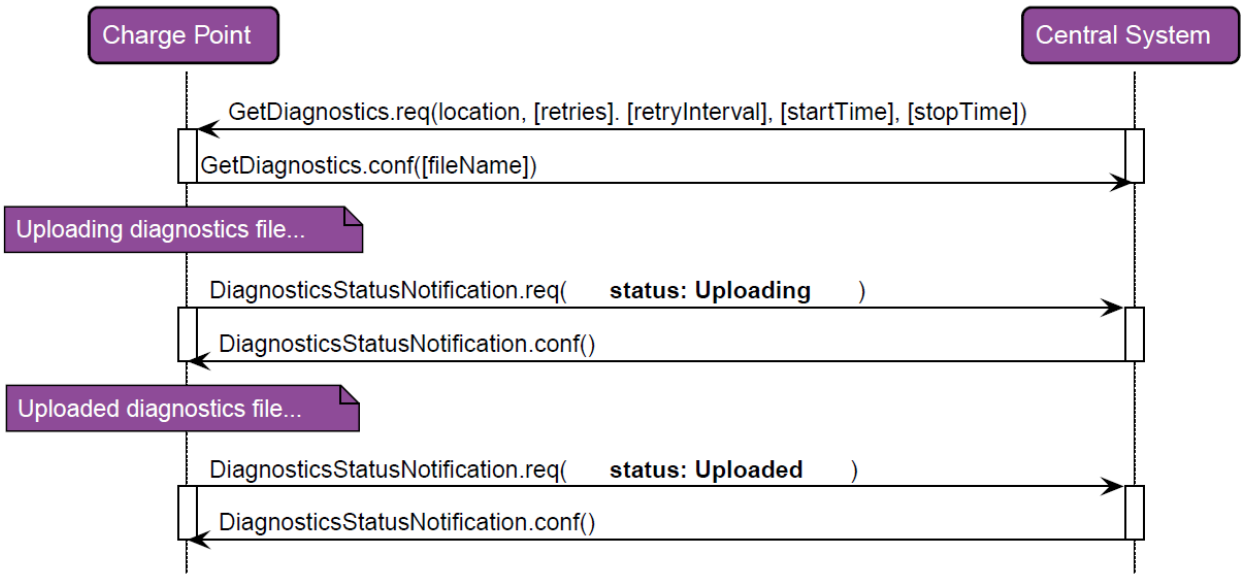
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nKeysCount	UDINT	Number of Configuration Keys following in this message.
aKeys	ARRAY[1..Param_OCPP [▶ 209].nConfigKeysMax] OF T_OCPP1_ConfigKey [▶ 208]	List of Configuration Keys for which the value of the configuration is requested.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.17 RecvGetDiagnostics

With this method, an OCPP client receives a Get Diagnostics request from the corresponding OCPP server. To respond to the request, the method RespGetDiagnostics [▶ 55] must be called.



Syntax

```
METHOD RecvGetDiagnostics : BOOL
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    sLocation      : STRING(255);
    nRetries       : UDINT;
    nRetryInterval : UDINT;
    nStartTime     : ULINT;
    nStopTime      : ULINT;
END_VAR
```

Return value

Name	Type	Description
RecvGetDiagnostics	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sLocation	STRING(255)	Contains the directory to which the diagnostic file is to be uploaded.
nRetries	UDINT	Optionally contains the number of attempts the Charge Point has made to upload the file.
nRetryInterval	UDINT	Optionally contains the number of seconds after which the upload is retried.
nStartTime	ULINT	Optionally marks the oldest point in time to be integrated into the diagnostic file.
nStopTime	ULINT	Optionally marks the most recent point in time to be integrated into the diagnostic file.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.18 RecvGetLocalListVersion

With this method, an OCPP client receives a Get Local List Version request from the corresponding OCPP server. To respond to the request, the method RespGetLocalListVersion [▶ 56] must be called.

Syntax

```
METHOD RecvGetLocalListVersion : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RecvGetLocalListVersion	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

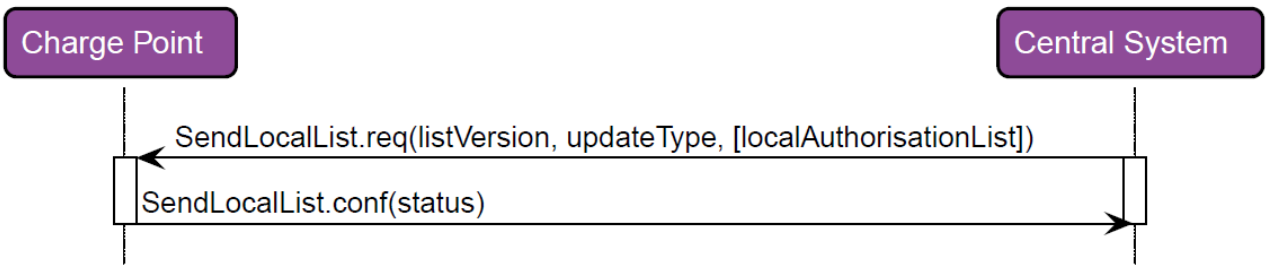
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.19 RecvLocalList

With this method, an OCPP client receives a Send Local List request from the corresponding OCPP server. To respond to the request, the method RespLocalList [▶ 57] must be called.



Syntax

```
METHOD RecvLocalList : BOOL
VAR_IN_OUT
    aAuthList    : ARRAY[*] OF ST_OCPP1_AuthorizationData;
END_VAR
VAR_OUTPUT
    hMessageId   : T_OCPP_MessageId;
    nListVersion : UDINT;
    eUpdateType  : E_OCPP1_UpdateType;
    nAuthCount   : UDINT := 0;
END_VAR
```

Return value

Name	Type	Description
RecvLocalList	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs/outputs

Name	Type	Description
aAuthList	ARRAY[*] OF ST_OCPP1_AuthorizationData [▶ 202]	List of authorized ID tags.

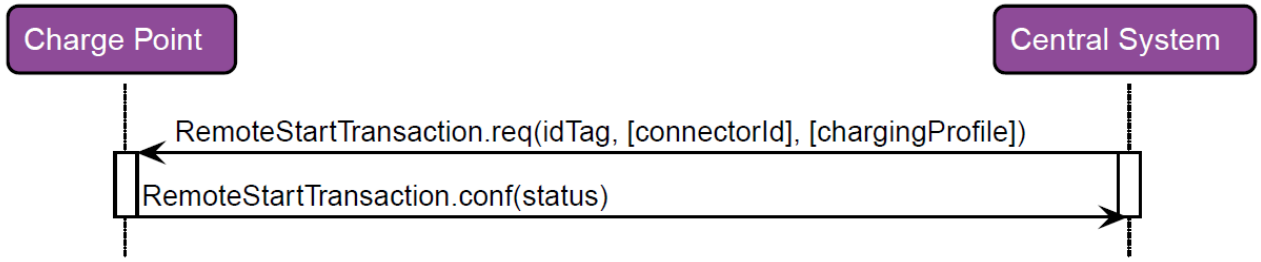
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nListVersion	UDINT	Contains the version number of the new list in the case of a complete update and the version number of the list after the update in the case of a differential update.
eUpdateType	E_OCPP1_UpdateType [▶ 197]	Information on whether a complete or incremental update should be carried out.
nAuthCount	UDINT	Number of following entries in the Local Authorization List.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.20 RecvRemoteStartTransaction

With this method, an OCPP client receives a Remote Start Transaction request from the corresponding OCPP server. To respond to the request, the method RespRemoteStartTransaction [▶ 58] must be called.



Syntax

```
METHOD RecvRemoteStartTransaction : BOOL
VAR_OUTPUT
    hMessageId   : T_OCPP_MessageId;
    nConnectorId : UDINT;
    sIdToken     : T_OCPP1_IdToken;
END_VAR
```

Return value

Name	Type	Description
RecvRemoteStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

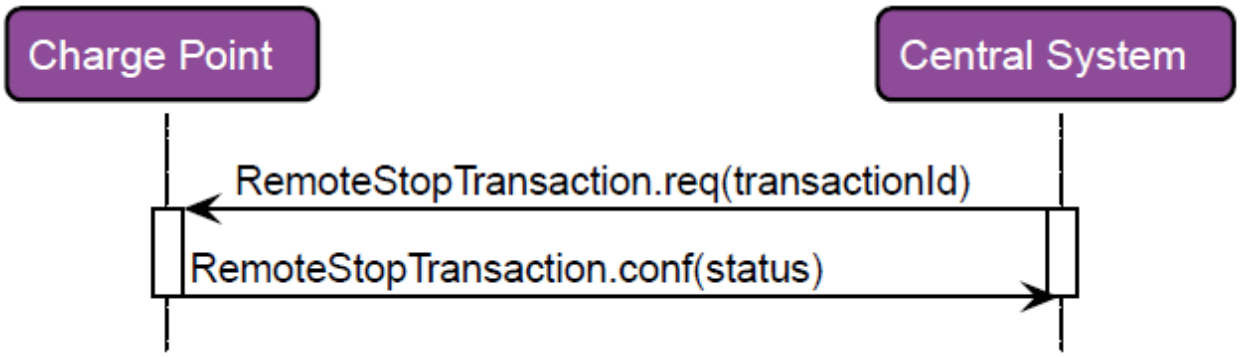
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
sIdToken	T_OCPP1_IdToken [▶ 209]	ID token of the Charge Point in the Central System.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.21 RecvRemoteStopTransaction

With this method, an OCPP client receives a Remote Stop Transaction request from the corresponding OCPP server. To respond to the request, the method RespRemoteStopTransaction [▶ 59] must be called.



Syntax

```
METHOD RecvRemoteStopTransaction : BOOL
VAR_OUTPUT
    hMessageId   : T_OCPP_MessageId;
    nConnectorId : UDINT;
END_VAR
```

Return value

Name	Type	Description
RecvRemoteStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

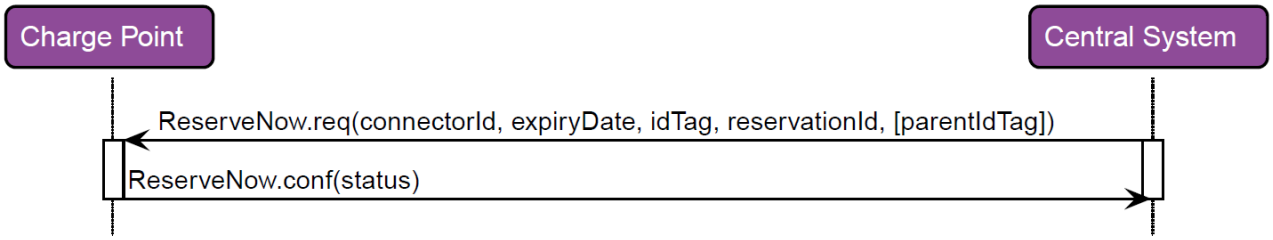
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point. The assignment of the TransactionId sent by the server to the appropriate Connector is carried out internally by the OCPP driver and does not have to be observed separately by the user.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.22 RecvReserveNow

With this method, an OCPP client receives a Reserve Now request from the corresponding OCPP server. To respond to the request, the method RespReserveNow [▶ 60] must be called.



Syntax

```
METHOD RecvReserveNow : BOOL
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    nConnectorId   : UDINT;
    nExpiryDate    : ULINT;
    sIdTag         : T_OCPP1_IdToken;
    sParentIdTag   : T_OCPP1_IdToken;
    nReservationId : UDINT;
END_VAR
```

Return value

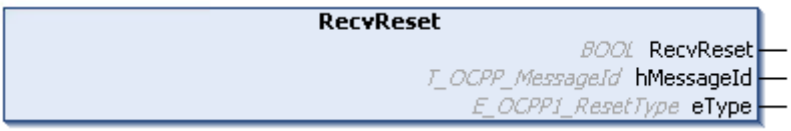
Name	Type	Description
RecvReserveNow	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

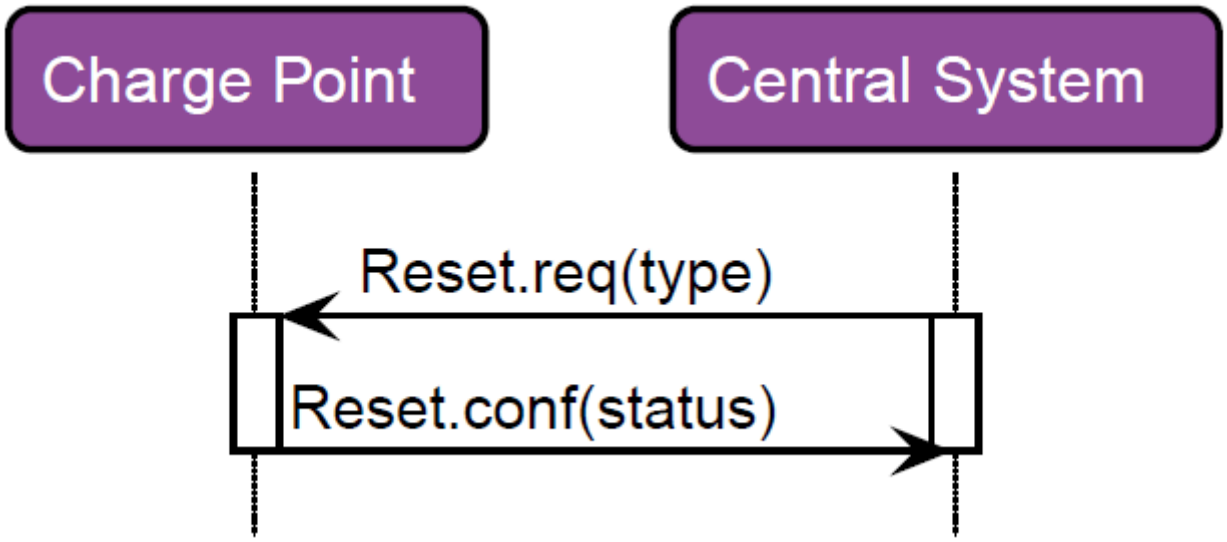
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	Contains the ID of the Connector to be reserved.
nExpiryDate	ULINT	Date and time at which the reservation ends.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID tag for which the Connector is to be reserved.
sParentIdTag	T_OCPP1_IdToken [▶ 209]	Optional parent ID tag.
nReservationId	UDINT	Unique ID for the reservation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.23 RecvReset



With this method, an OCPP client receives a Reset request from the corresponding OCPP server. To respond to the request, the method RespReset [▶ 61] must be called.



Syntax

```
METHOD RecvReset : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    eType : E_OCPP1_ResetType;
END_VAR
```

Return value

Name	Type	Description
RecvReset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

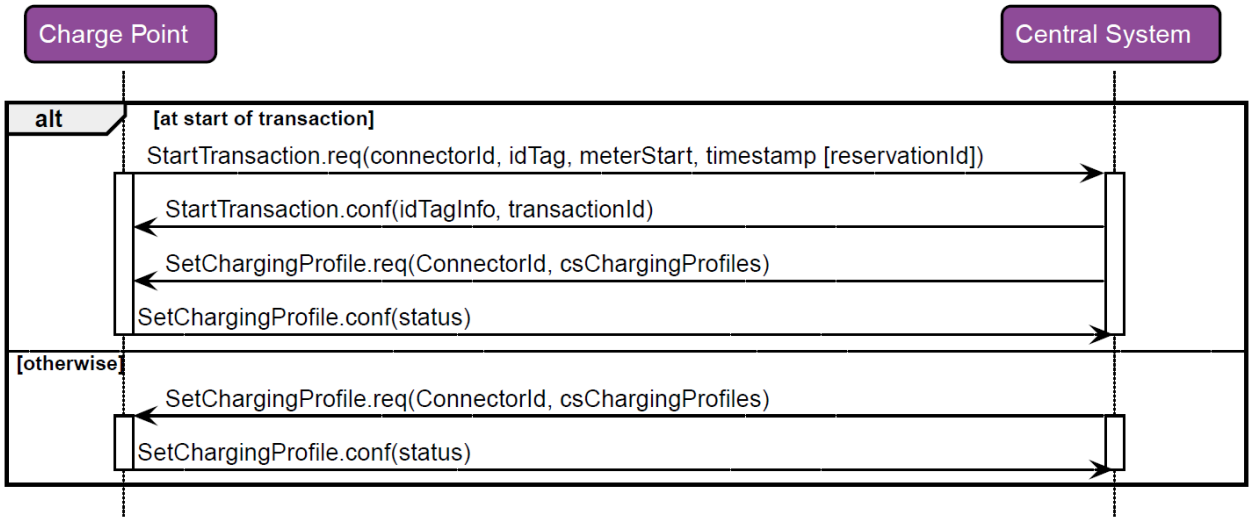
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eType	E_OCPP1_ResetType [▶ 196]	Type of reset that the Charge Point should perform.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.24 RecvSetChargingProfile

With this method, an OCPP client receives a Set Charging Profile request from the corresponding OCPP server. To respond to the request, the method RespSetChargingProfile [▶ 62] must be called.



Syntax

```
METHOD RecvSetChargingProfile : BOOL
VAR_OUTPUT
    hMessageId       : T_OCPP_MessageId;
    nConnectorId     : UDINT;
    mChargingProfile : ST_OCPP1_ChargingProfileMax;
END_VAR
```

Return value

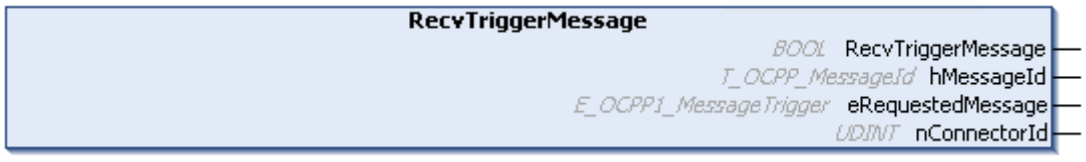
Name	Type	Description
RecvSetChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

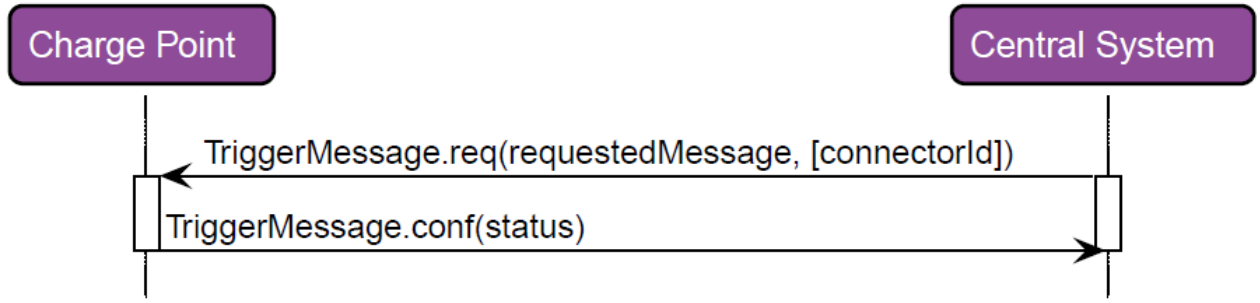
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
mChargingProfile	ST_OCPP1_ChargingProfileMax [▶ 204]	The Charging Profile to be used in the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.25 RecvTriggerMessage



With this method, an OCPP client receives a Trigger Message request from the corresponding OCPP server. To respond to the request, the method RespTriggerMessage [▶ 63] must be called.



Syntax

```
METHOD RecvTriggerMessage : BOOL
VAR_OUTPUT
    hMessageId        : T_OCPP_MessageId;
    eRequestedMessage : E_OCPP1_MessageTrigger;
    nConnectorId      : UDINT;
END_VAR
```

Return value

Name	Type	Description
RecvTriggerMessage	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

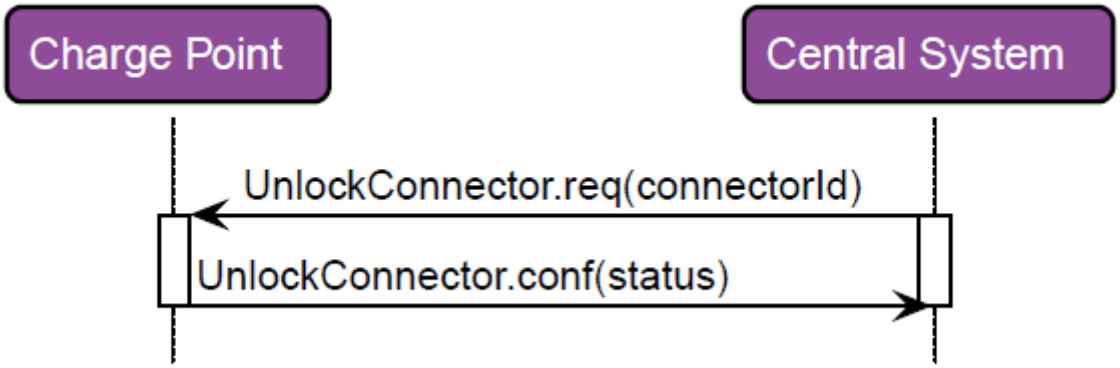
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eRequestedMessage	E_OCPP1_MessageTrigger [▶ 194]	Requested message that the Charge Point should send to the Central System.
nConnectorId	UDINT	ID of the Connector of a Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.26 RecvUnlockConnector

With this method, an OCPP client receives an Unlock Connector request from the corresponding OCPP server. To respond to the request, the method RespUnlockConnector [▶ 64] must be called.



Syntax

```
METHOD RecvUnlockConnector : BOOL
VAR_OUTPUT
    hMessageId   : T_OCPP_MessageId;
    nConnectorId : UDINT;
END_VAR
```

Return value

Name	Type	Description
RecvUnlockConnector	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.27 RecvUpdateFirmware

With this method, an OCPP client receives an Update Firmware request from the corresponding OCPP server. To respond to the request, the method RespUpdateFirmware [▶ 64] must be called.

Syntax

```
METHOD RecvUpdateFirmware : BOOL
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    sLocation      : STRING(255);
    nRetries       : UDINT;
    nRetryInterval : UDINT;
    nRetrieveDate  : ULINT;
END_VAR
```

Return value

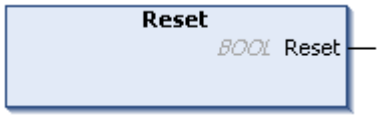
Name	Type	Description
RecvUpdateFirmware	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sLocation	STRING(255)	String with the URI from which the firmware is to be obtained.
nRetries	UDINT	Optionally contains information on how often the Charge Point should try to download the firmware before it gives up.
nRetryInterval	UDINT	Optionally contains the information after which time a new attempt should be made.
nRetrieveDate	ULINT	Contains information on when the Charge Point is permitted to receive the new firmware.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.28 Reset



This method is used to reset the last error present on the function block.

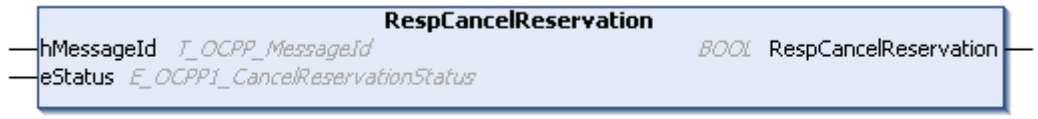
Syntax

```
METHOD Reset : BOOL
```

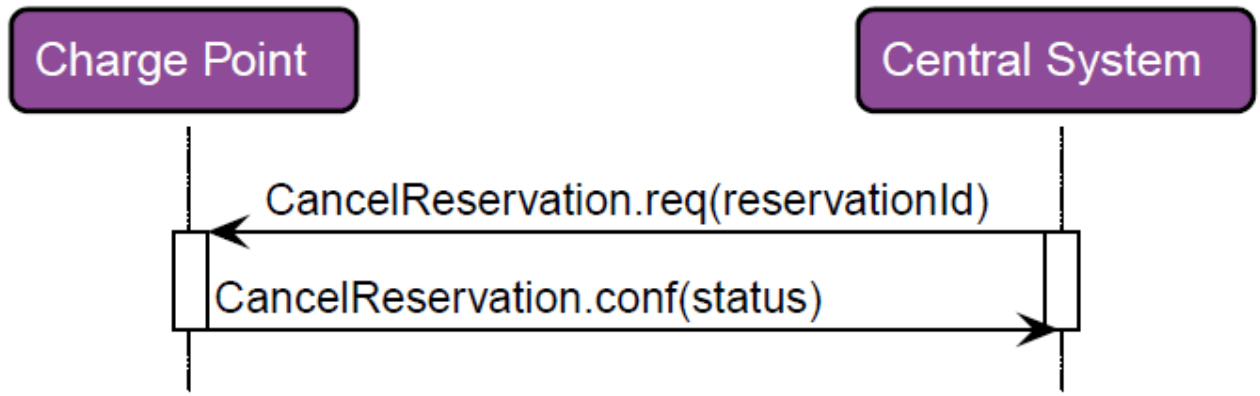
Return value

Name	Type	Description
Reset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

5.1.1.29 RespCancelReservation



With this method, an OCPP client responds to a Cancel Reservation request from the corresponding OCPP server.



Syntax

```
METHOD RespCancelReservation : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_CancelReservationStatus;
END_VAR
```

Return value

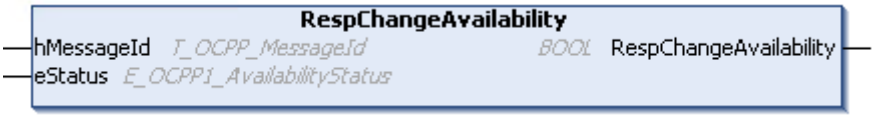
Name	Type	Description
RespCancelReservation	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

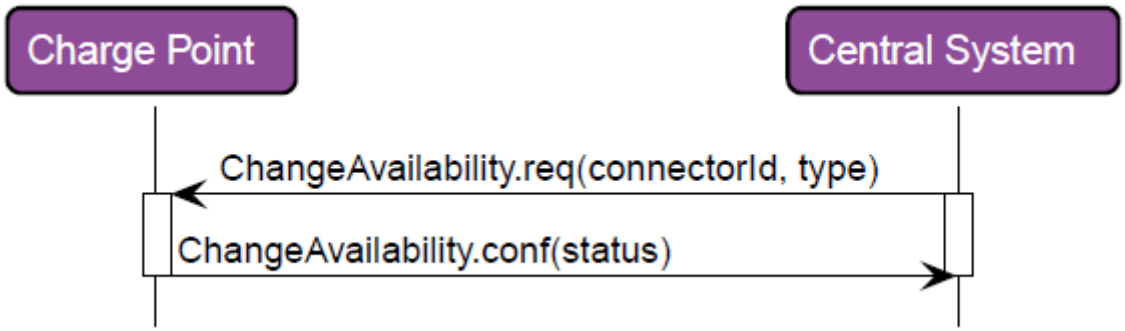
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_CancelReservationStatus [▶ 190]	Response whether the Charge Point was able to perform the requested cancellation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.30 RespChangeAvailability



With this method, an OCPP client responds to a Change Availability request from the corresponding OCPP server.



Syntax

```
METHOD RespChangeAvailability : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_AvailabilityStatus;
END_VAR
```

Return value

Name	Type	Description
RespChangeAvailability	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

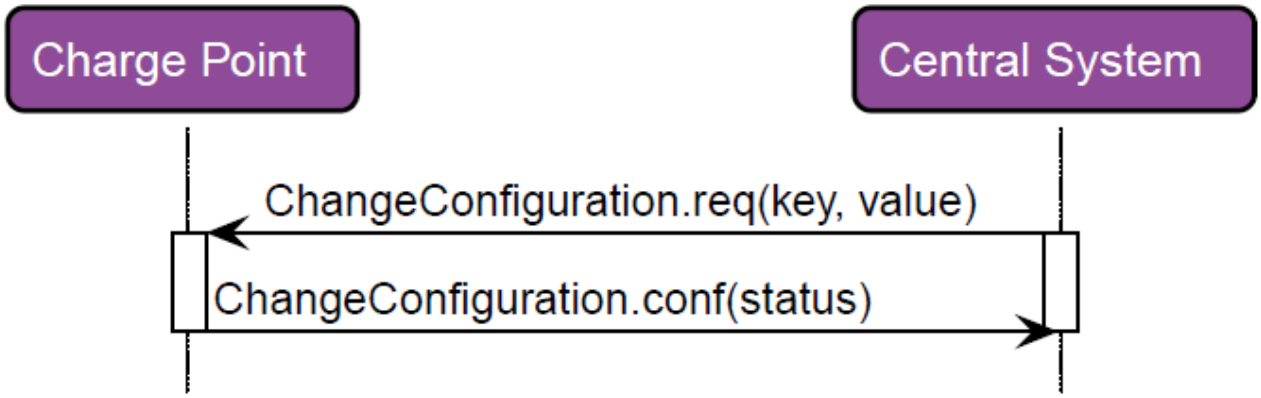
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AvailabilityStatus [▶ 190]	Response whether the Charge Point was able to implement the requested availability change.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.31 RespChangeConfiguration

With this method, an OCPP client responds to a Change Configuration request from the corresponding OCPP server.



Syntax

```
METHOD RespChangeConfiguration : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_ConfigurationStatus;
END_VAR
```

Return value

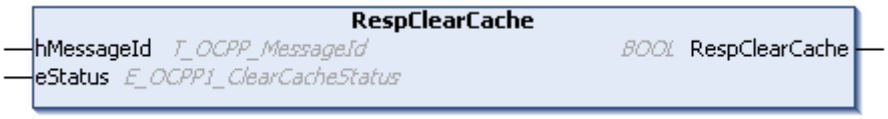
Name	Type	Description
RespChangeAvailability	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

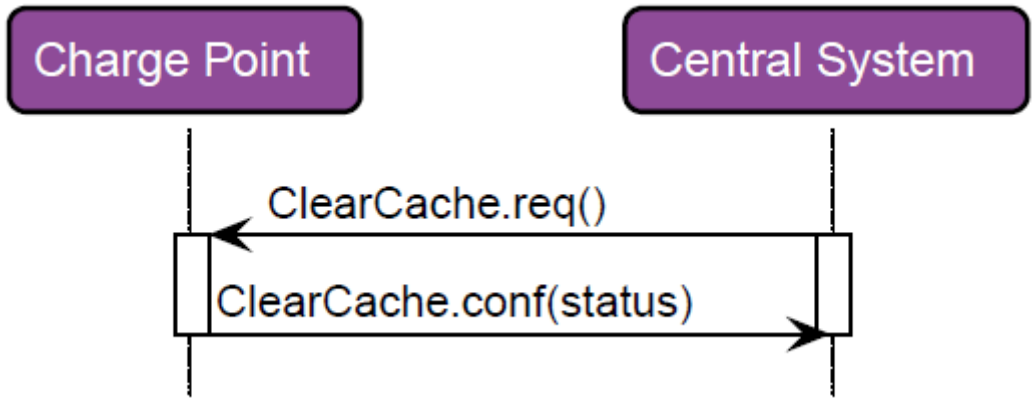
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_ConfigurationStatus [▶ 193]	Response whether the Charge Point was able to implement the requested configuration change.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.32 RespClearCache



With this method, an OCPP client responds to a Clear Cache request from the corresponding OCPP server.



Syntax

```
METHOD RespClearCache : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_ClearCacheStatus;
END_VAR
```

Return value

Name	Type	Description
RespClearCache	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

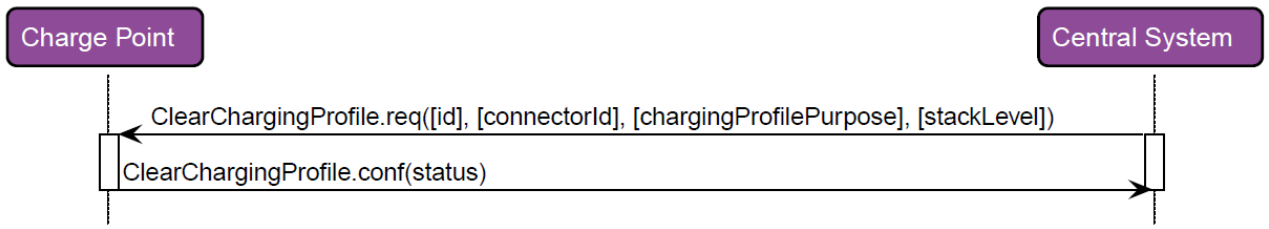
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_ClearCacheStatus [▶ 192]	Response whether the Charge Point has implemented the requested clearing of the cache.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.33 RespClearChargingProfile

With this method, an OCPP client responds to a Clear Charging Profile request from the corresponding OCPP server.



Syntax

```
METHOD RespClearChargingProfile : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    :  E_OCPP1_ClearChargingProfileStatus;
END_VAR
```

Return value

Name	Type	Description
RespClearChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

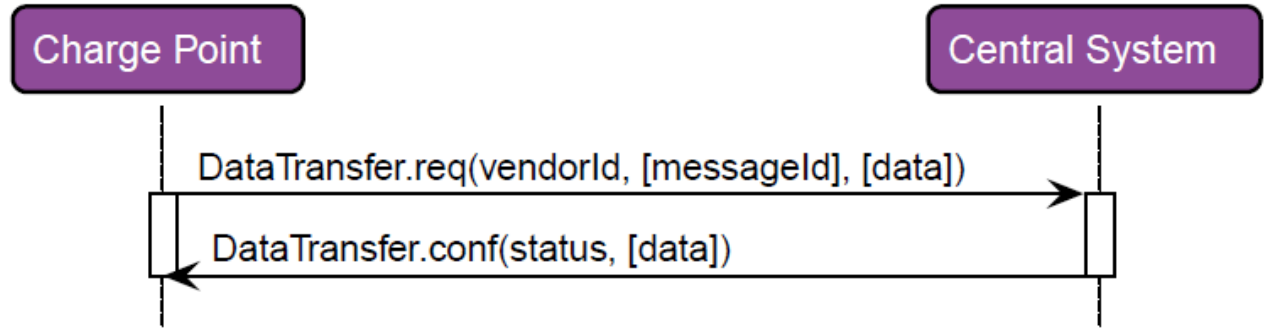
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_ClearChargingProfileStatus [▶ 192]	Response whether the Charge Point was able to implement the requested clearing.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.34 RespDataTransfer

With this method, an OCPP client responds to a Data Transfer request from the corresponding OCPP server.



Syntax

```
METHOD RespDataTransfer : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_DataTransferStatus;
END_VAR
```

Return value

Name	Type	Description
RespDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

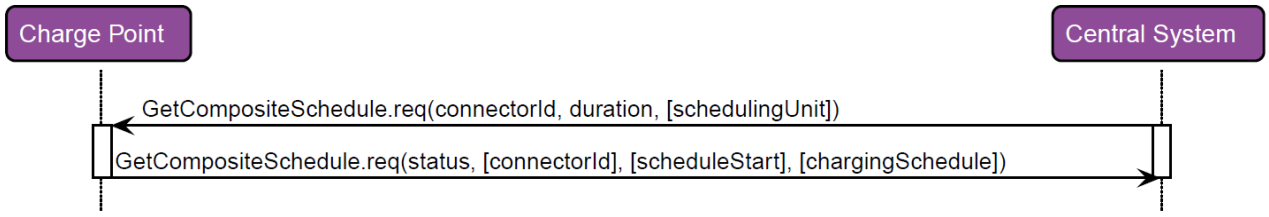
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_DataTransferStatus [▶ 193]	Response whether the Data Transfer request from the OCPP server was successfully completed.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.35 RespGetCompositeSchedule

With this method, an OCPP client responds to a Get Composite Schedule request from the corresponding OCPP server.



Syntax

```
METHOD RespGetCompositeSchedule : BOOL
VAR_INPUT
    hMessageId        : T_OCPP_MessageId;
    eStatus           : E_OCPP1_GetCompositeScheduleStatus;
    nConnectorId      : UDINT := 0;
    nScheduleStart    : ULINT := 0;
    mChargingSchedule : REFERENCE TO ST_OCPP1_ChargingSchedule REF= 0;
END_VAR
```

Return value

Name	Type	Description
RespGetCompositeSchedule	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

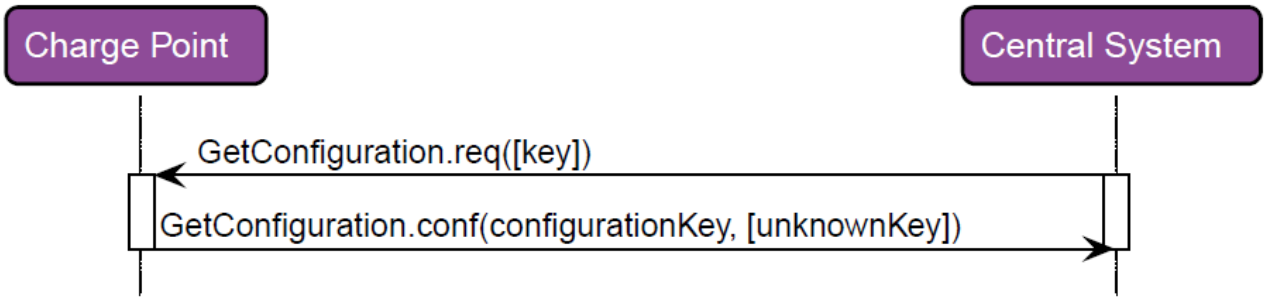
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_GetCompositeScheduleStatus [▶ 194]	Response whether the Schedule could be queried on the Charge Point side.
nConnectorId	UDINT	Can optionally contain the ID of a Connector of a Charge Point. The returned Schedule would then apply to this Connector.
nScheduleStart	ULINT	Optionally contains the time, all values contained here are displayed relative to this time.
mChargingSchedule	REFERENCE TO ST_OCPP1_ChargingSchedule [▶ 205]	Optionally contains the planned Schedule over time.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.36 RespGetConfiguration

With this method, an OCPP client responds to a Get Configuration request from the corresponding OCPP server.



Syntax

```
METHOD RespGetConfiguration : BOOL
VAR_INPUT
    hMessageId    : T_OCPP_MessageId;
    nKeysCount    : UDINT := 0;
```

```
    nUnknownCount : UDINT := 0;
END_VAR
VAR_IN_OUT CONSTANT
    aKeys         : ARRAY[*] OF ST_OCPP1_ConfigKeyValue;
    aUnknown      : ARRAY[*] OF T_OCPP1_ConfigKey;
END_VAR
```

Return value

Name	Type	Description
RespGetConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nKeysCount	UDINT	Number of the following Configuration Keys.
nUnknownCount	UDINT	Number of the following Unknown Configuration Keys.

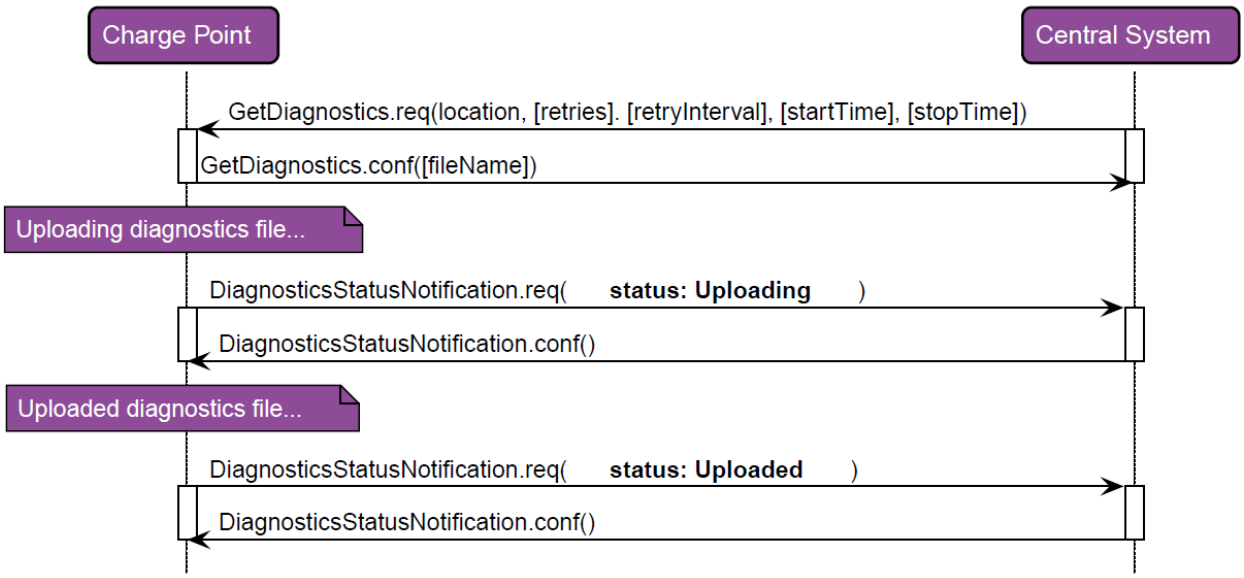
Inputs/outputs

Name	Type	Description
aKeys	ARRAY[*] OF ST_OCPP1_ConfigKeyValue [▶ 207]	List of requested or known Configuration Keys.
aUnknown	ARRAY[*] OF T_OCPP1_ConfigKey [▶ 208]	Requested Configuration Keys that are not known.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.37 RespGetDiagnostics

With this method, an OCPP client responds to a Get Diagnostics request from the corresponding OCPP server.



Syntax

```
METHOD RespGetDiagnostics : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    sFileName  : STRING(255) := '';
END_VAR
```

Return value

Name	Type	Description
RespGetDiagnostics	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sFileName	STRING(255)	Optionally contains the file name of the diagnostic file that is uploaded.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.38 RespGetLocalListVersion

With this method, an OCPP client responds to a Get Local List Version request from the corresponding OCPP server.

Syntax

```
METHOD RespGetLocalListVersion : BOOL
VAR_INPUT
    hMessageId   : T_OCPP_MessageId;
    nListVersion : UDINT;
END_VAR
```

Return value

Name	Type	Description
RespGetLocalListVersion	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

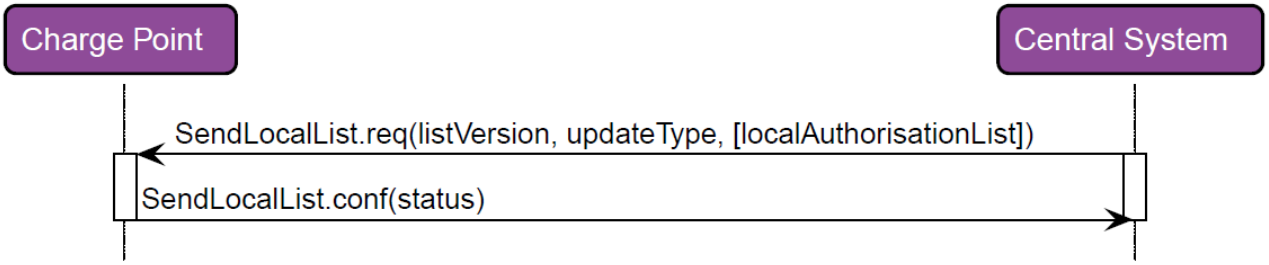
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nListVersion	UDINT	Contains the current version number of the Local Authorization List in the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.39 RespLocalList

With this method, an OCPP client responds to a Send Local List request from the corresponding OCPP server.



Syntax

```
METHOD RespLocalList : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_UpdateStatus;
END_VAR
```

Return value

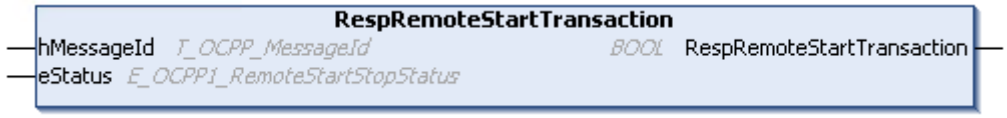
Name	Type	Description
RespLocalList	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

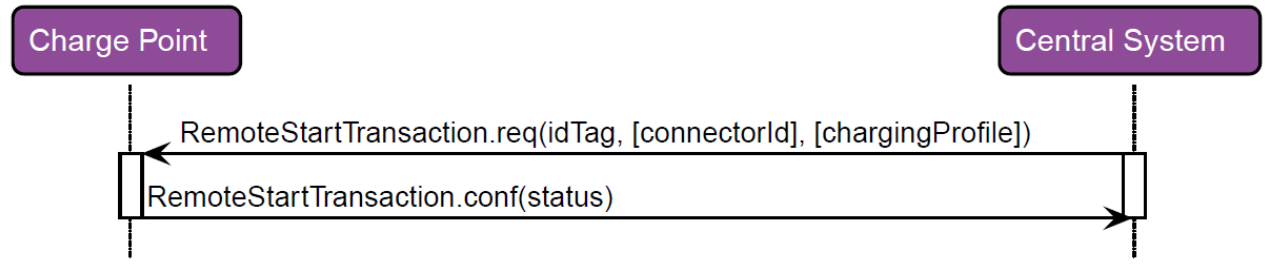
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_UpdateStatus [▶ 197]	Response whether the Charge Point was able to receive the sent Local Authorization List and perform the update.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.40 RespRemoteStartTransaction



With this method, an OCPP client responds to a Remote Start Transaction request from the corresponding OCPP server.



Syntax

```
METHOD RespRemoteStartTransaction : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

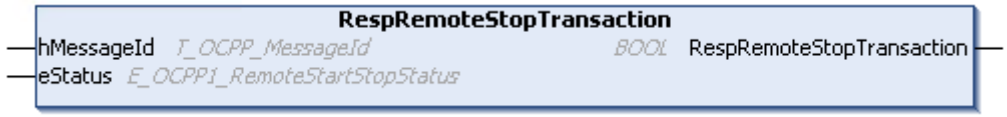
Name	Type	Description
RespRemoteStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

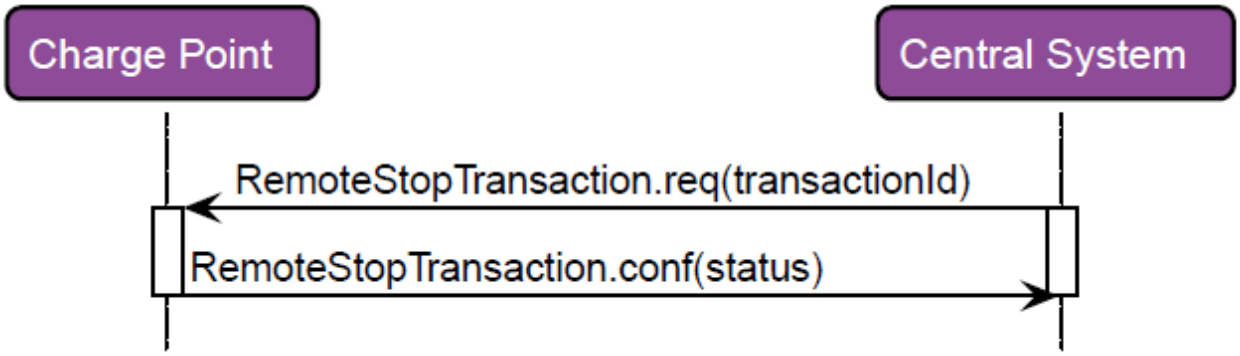
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status shows whether the Charge Point has accepted the request to start a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.41 RespRemoteStopTransaction



With this method, an OCPP client responds to a Remote Stop Transaction request from the corresponding OCPP server.



Syntax

```
METHOD RespRemoteStopTransaction : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

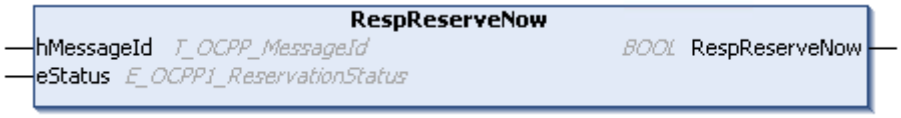
Name	Type	Description
RespRemoteStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

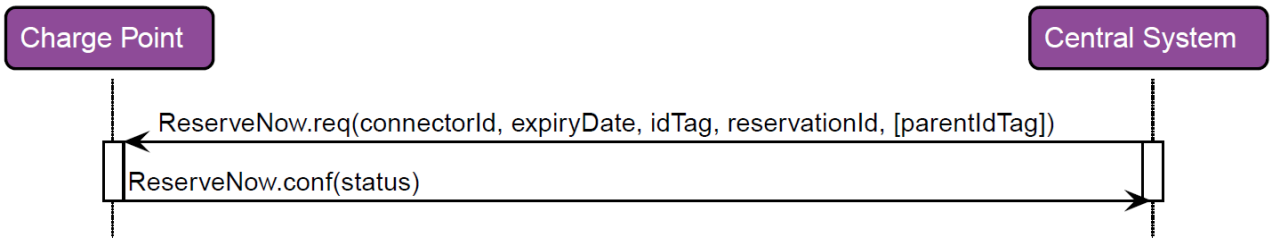
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status indicates whether the Charge Point has accepted the request to stop a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.42 RespReserveNow



With this method, an OCPP client responds to a Reserve Now request from the corresponding OCPP server.



Syntax

```
METHOD RespReserveNow : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_ReservationStatus;
END_VAR
```

Return value

Name	Type	Description
RespReserveNow	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

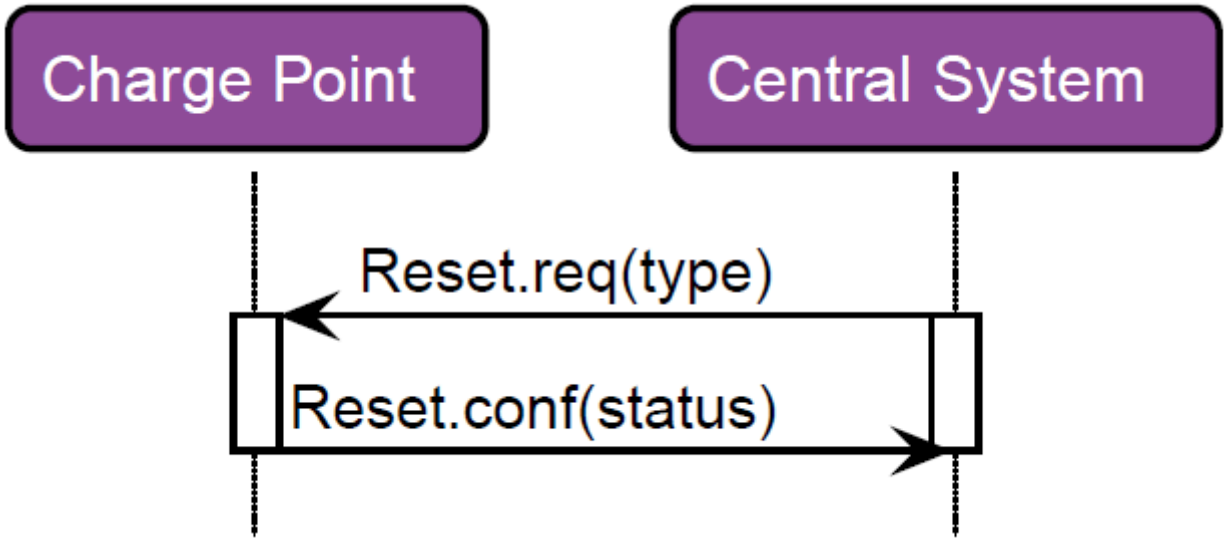
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_ReservationStatus [▶ 196]	Response whether the Charge Point has accepted or rejected the requested reservation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.43 RespReset

With this method, an OCPP client responds to a Reset request from the corresponding OCPP server.



Syntax

```
METHOD RespReset : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_ResetStatus;
END_VAR
```

Return value

Name	Type	Description
RespReset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

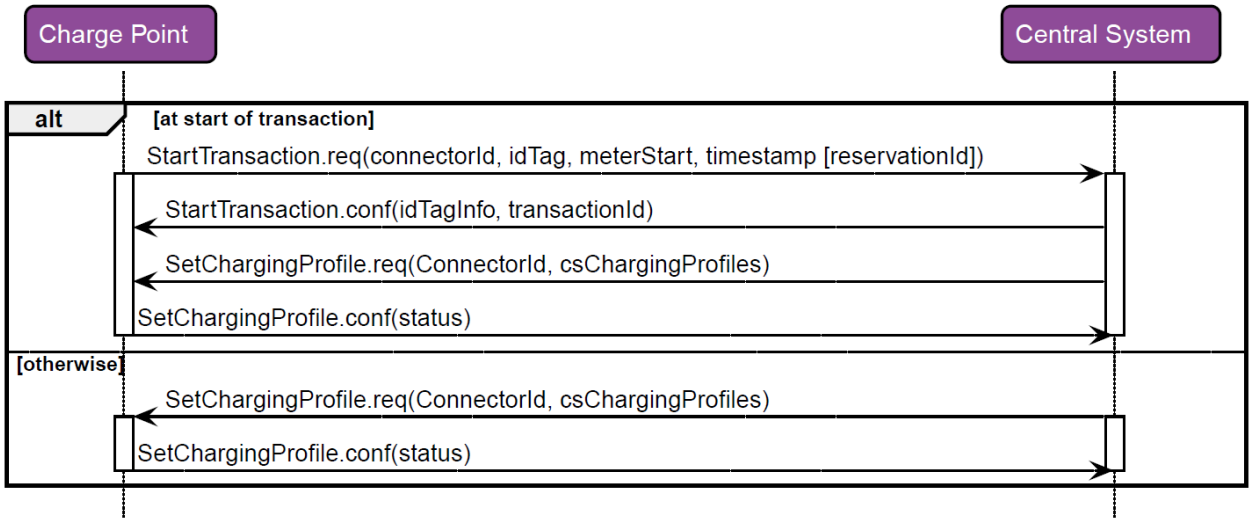
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_ResetStatus [▶ 196]	The status shows whether the Charge Point was able to accept the Reset request.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.44 RespSetChargingProfile

With this method, an OCPP client responds to a Set Charging Profile request from the corresponding OCPP server.



Syntax

```
METHOD RespSetChargingProfile : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_ChargingProfileStatus;
END_VAR
```

Return value

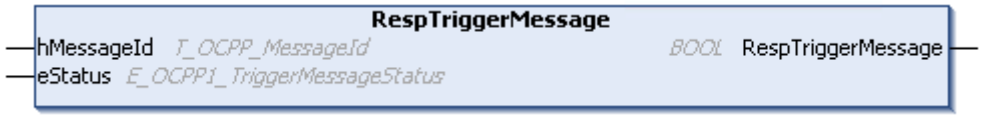
Name	Type	Description
RespSetChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

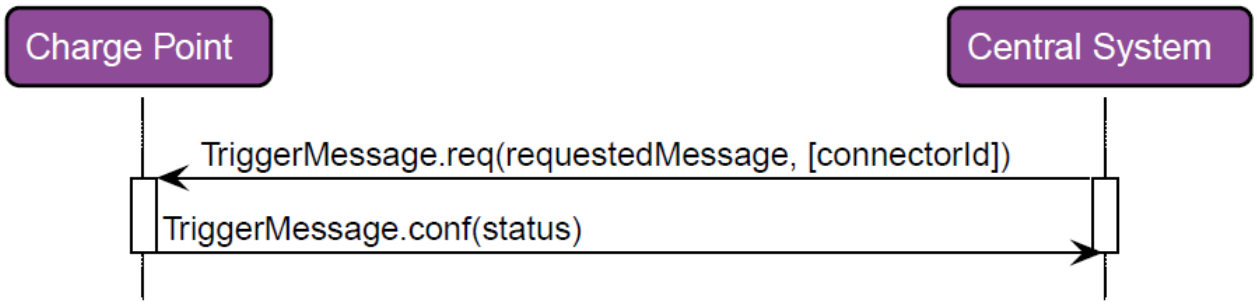
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_ChargingProfileStatus [▶ 191]	Response whether the Charge Point was able to implement the requested Charging Profile.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.45 RespTriggerMessage



With this method, an OCPP client responds to a Trigger Message request from the corresponding OCPP server.



Syntax

```
METHOD RespTriggerMessage : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    :E_OCPP1_TriggerMessageStatus;
END_VAR
```

Return value

Name	Type	Description
RespTriggerMessage	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

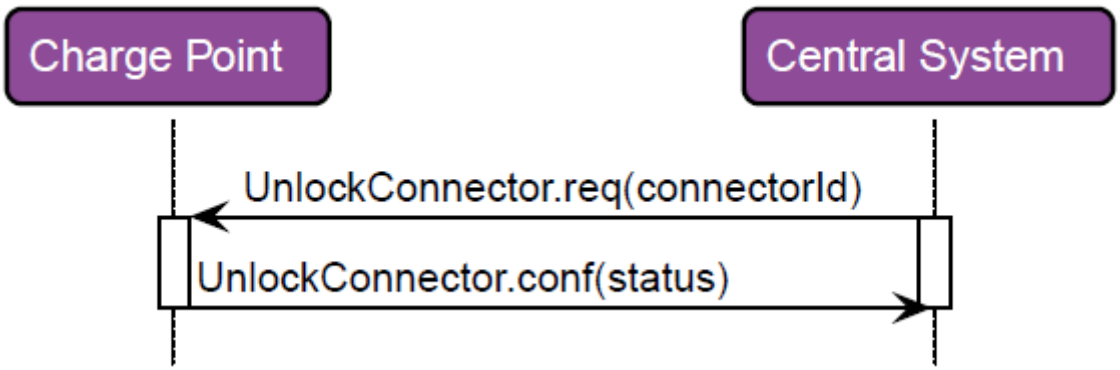
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_TriggerMessageStatus [▶ 196]	Response whether the Charge Point will send the requested message or not.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.46 RespUnlockConnector

With this method, an OCPP client responds to an Unlock Connector request from the corresponding OCPP server.



Syntax

```
METHOD RespUnlockConnector : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_UnlockStatus;
END_VAR
```

Return value

Name	Type	Description
RespUnlockConnector	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_UnlockStatus [▶ 197]	The status shows whether the Connector has been unlocked.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.47 RespUpdateFirmware

With this method, an OCPP client responds to an Update Firmware request from the corresponding OCPP server.

Syntax

```
METHOD RespUpdateFirmware : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

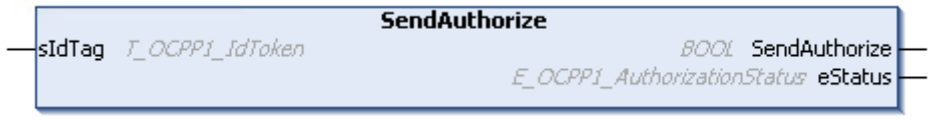
Name	Type	Description
RespUpdateFirmware	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

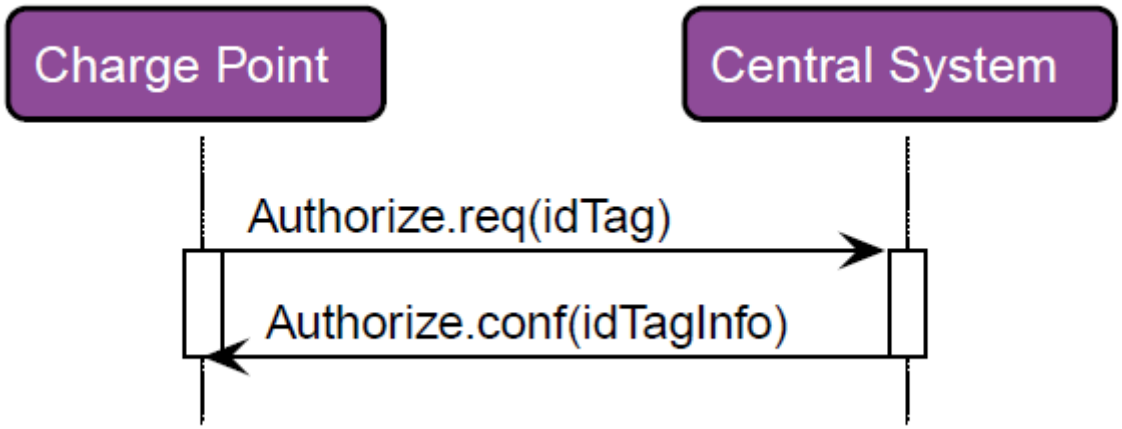
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.48 SendAuthorize



With this method, an OCPP client sends an Authorize request to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendAuthorize : BOOL
VAR_INPUT
    sIdTag  : T_OCPP1_IdToken;
END_VAR
VAR_OUTPUT
    eStatus : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

Name	Type	Description
SendAuthorize	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token with which the Charge Point wants to be authorized on the Central System.

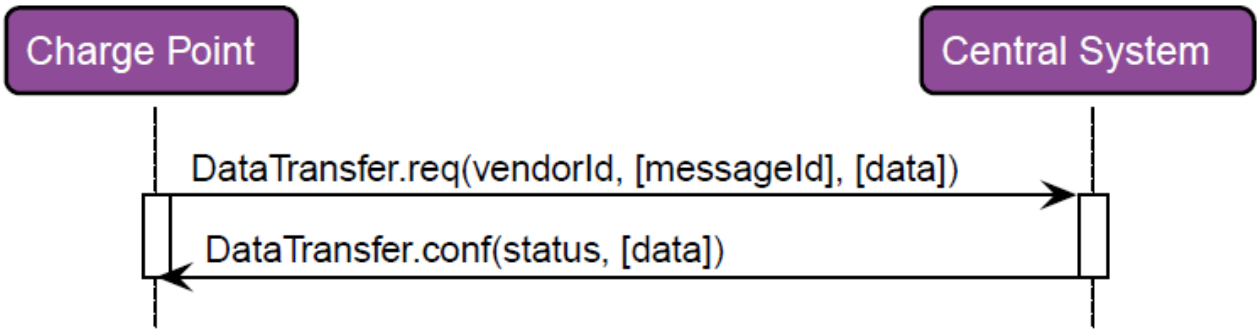
Outputs

Name	Type	Description
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization as a response from the Central System.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.49 SendDataTransfer

With this method, an OCPP client sends a Data Transfer request to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendDataTransfer : BOOL
VAR_INPUT
    sVendorId  : STRING(255);
    sMessageId : STRING(63) := '';
END_VAR
VAR_IN_OUT CONSTANT
    sData      : T_OCPP1_DataTransferData;
END_VAR
VAR_OUTPUT
    eStatus    : E_OCPP1_DataTransferStatus;
END_VAR
```

Return value

Name	Type	Description
SendDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sVendorId	STRING(255)	Identifier for the vendor, which identifies the vendor-specific implementation.
sMessageId	STRING(63)	Additional identification field for a single message.

Inputs/outputs

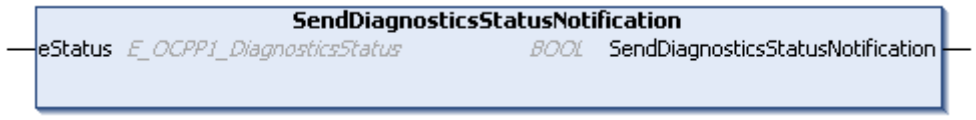
Name	Type	Description
sData	T_OCPP1_DataTransferData [▶ 208]	Text without specified length and format.

Outputs

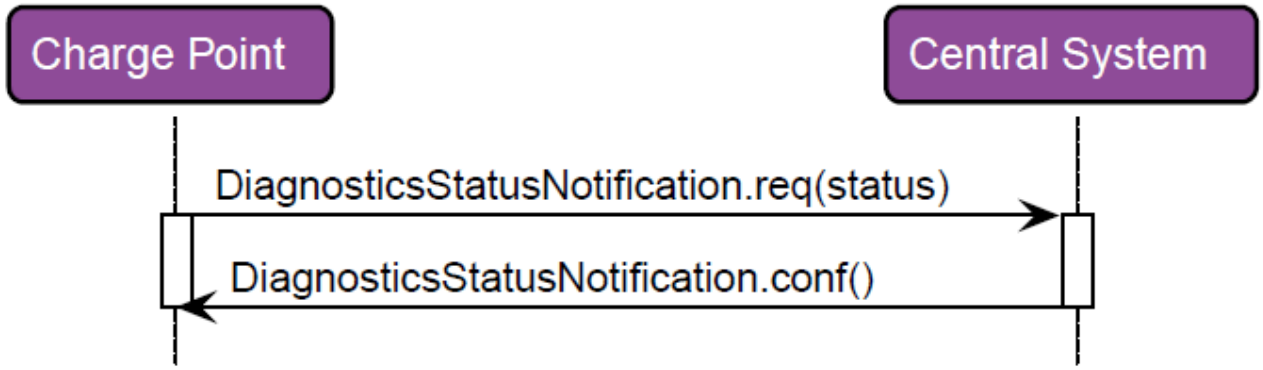
Name	Type	Description
eStatus	E_OCPP1_DataTransferStatus [▶ 193]	Status of the Data Transfer as a response from the Central System.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.50 SendDiagnosticsStatusNotification



With this method, an OCPP client sends a Diagnostics Status Notification to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendDiagnosticsStatusNotification : BOOL
VAR_INPUT
    eStatus : E_OCPP1_DiagnosticsStatus;
END_VAR
```

Return value

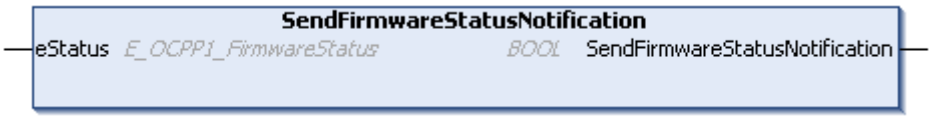
Name	Type	Description
SendDiagnosticsStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

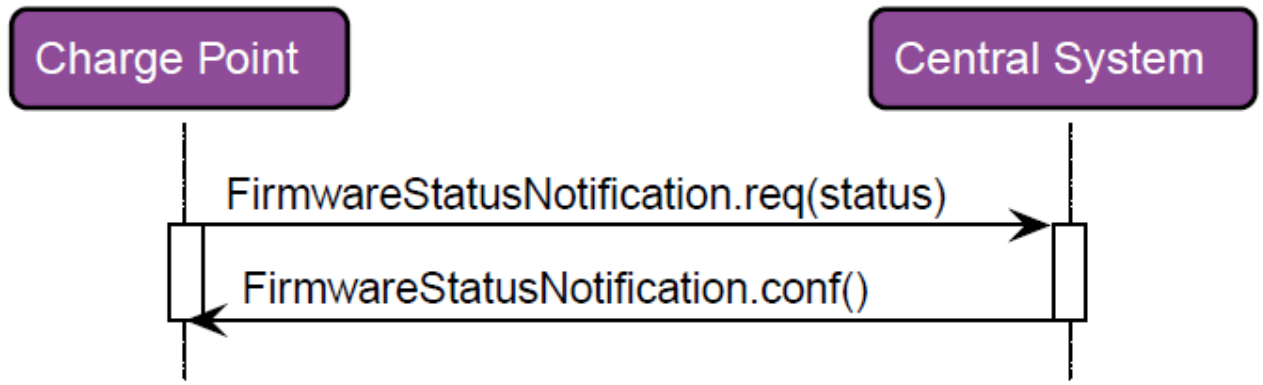
Name	Type	Description
eStatus	E_OCPP1_DiagnosticsStatus [▶ 193]	Contains the status of the upload of the diagnostic data.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.51 SendFirmwareStatusNotification



With this method, an OCPP client sends a Firmware Status Notification to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendFirmwareStatusNotification : BOOL
VAR_INPUT
    eStatus : E_OCPP1_FirmwareStatus;
END_VAR
```

Return value

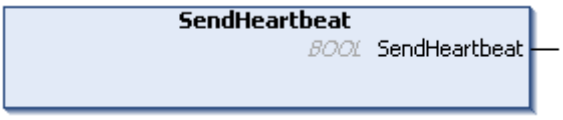
Name	Type	Description
SendFirmwareStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

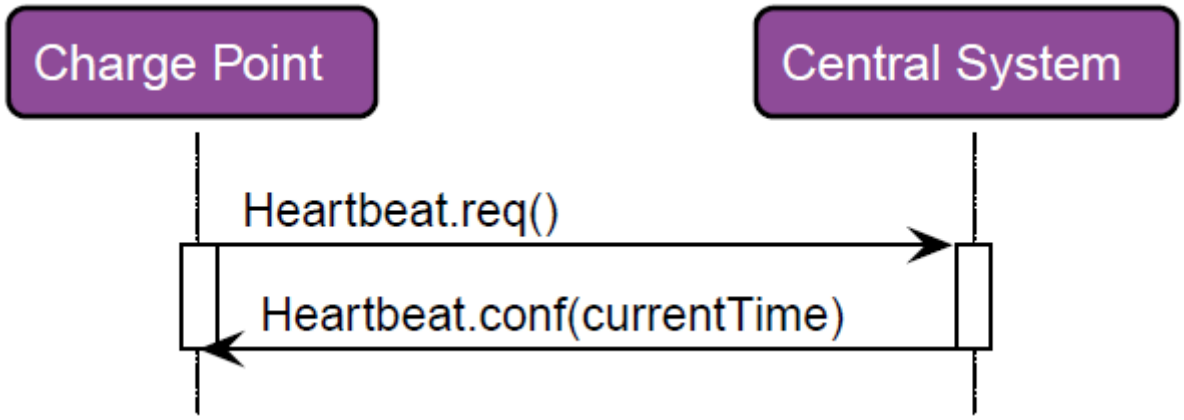
Name	Type	Description
eStatus	E_OCPP1_FirmwareStatus [▶ 193]	Contains the progress of the firmware installation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.52 SendHeartbeat



With this method, an OCPP client sends a Heartbeat to the corresponding OCPP server. The client already sends a Heartbeat internally, which can be configured via the HeartbeatInterval Property. This method would configure another Heartbeat in parallel if necessary.



Syntax

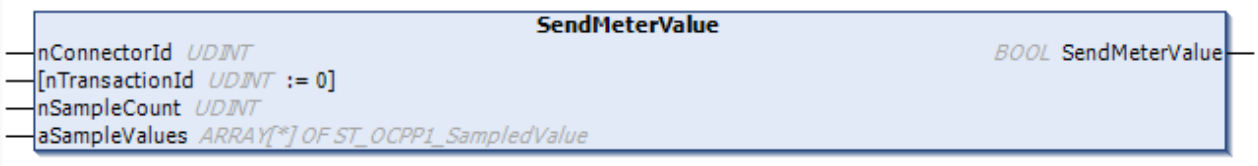
```
METHOD SendHeartbeat : BOOL
```

Return value

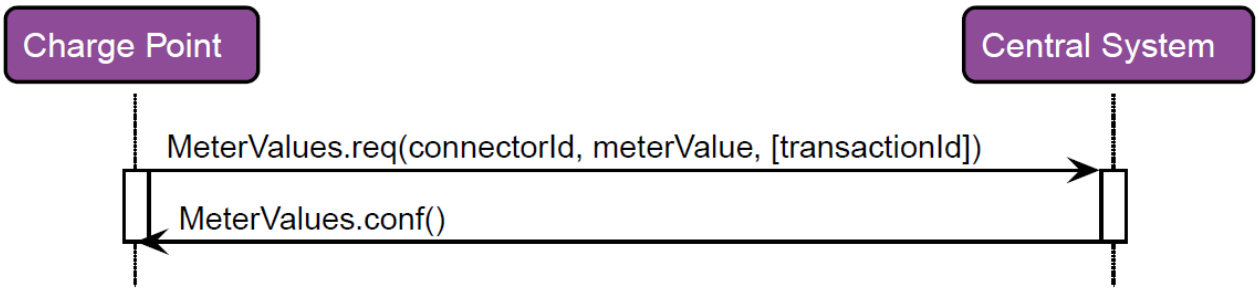
Name	Type	Description
SendHeartbeat	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.53 SendMeterValue



With this method, an OCPP client sends Meter Values to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendMeterValue : BOOL
VAR_INPUT
    nConnectorId    : UDINT;
    nTransactionId  : UDINT := 0;
    nSampleCount    : UDINT;
END_VAR
```

```
VAR_IN_OUT
    arrSampleValues : ARRAY[*] OF ST_OCPP1_SampledValue;
END_VAR
```

Return value

Name	Type	Description
SendMeterValue	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nTransactionId	UDINT	Optionally contains the ID of the transaction to which the specified Meter Values belong.
nSampleCount	UDINT	The number of the following sampled values.

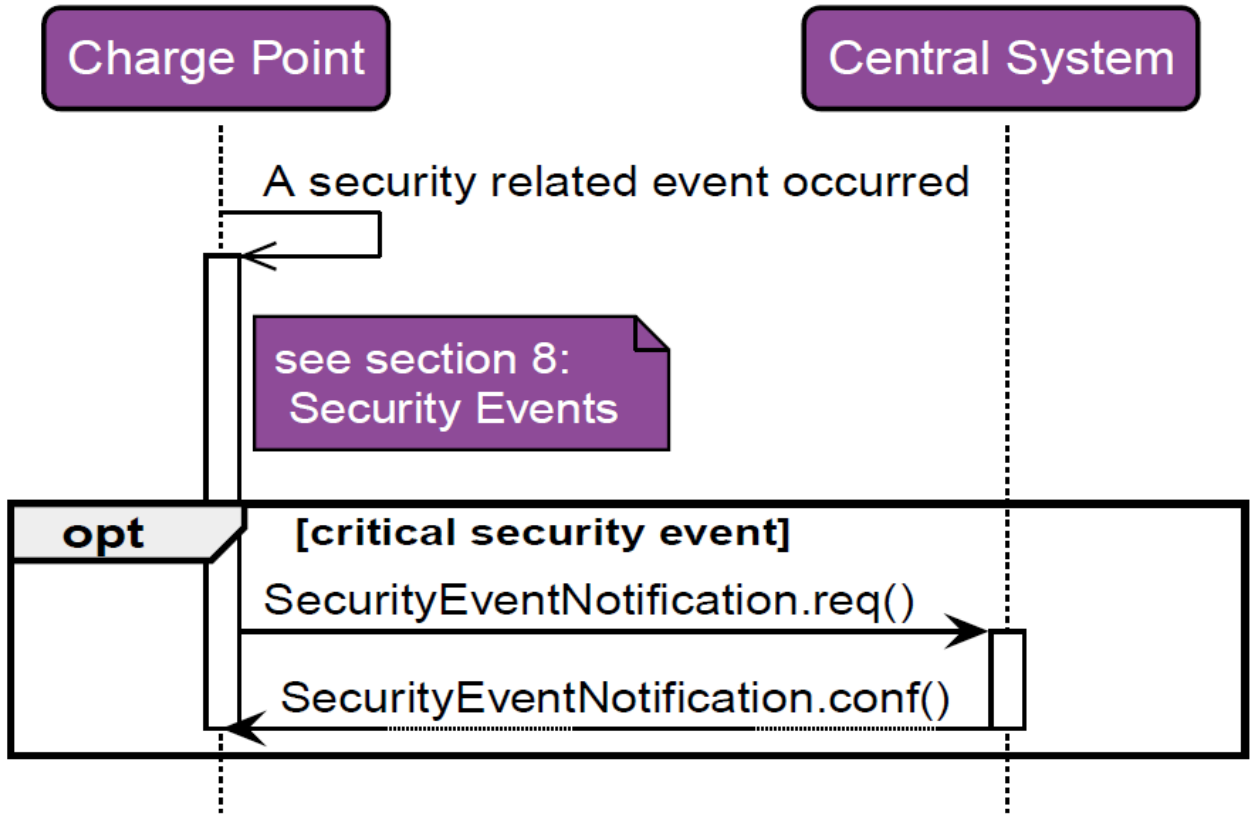
Inputs/outputs

Name	Type	Description
arrSampleValues	ARRAY [*] OF ST_OCPP1_SampledValue [▶ 207]	The sampled values to be sent to the Central System.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.54 SendSecurityEventNotification

With this method, an OCPP client sends a Security Event Notification to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendSecurityEventNotification : BOOL
VAR_INPUT
    nTimestamp : ULINT;
    sType      : STRING(63);
    sInfo      : STRING(255) := '';
END_VAR
```

Return value

Name	Type	Description
SendDiagnosticsStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

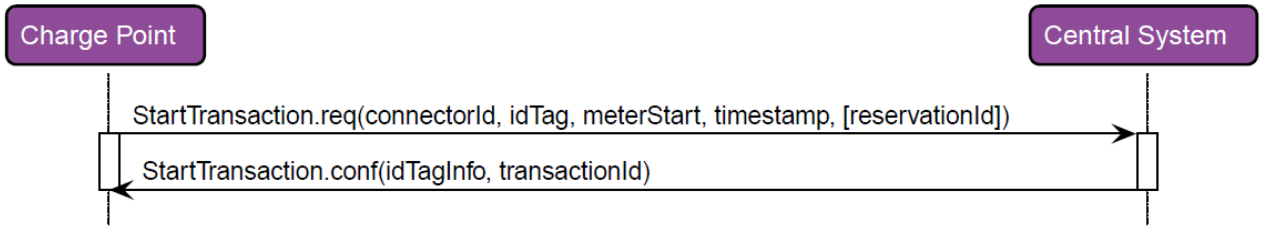
Inputs

Name	Type	Description
nTimestamp	ULINT	Date and time when the Security Event occurs.
sType	STRING(63)	Type of the Security Event.
sInfo	STRING(255)	Optionally contains additional information about the Security Event that has occurred.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.55 SendStartTransaction

With this method, an OCPP client sends a Start Transaction request to the corresponding OCPP server. The response from the OCPP server is processed directly within the method. The Transaction ID is managed internally so that the user does not have to pay attention to the administration.



Syntax

```
METHOD SendStartTransaction : BOOL
VAR_INPUT
    sIdTag          : T_OCPP1_IdToken;
    nConnectorId    : UDINT;
    nMeterStart     : UDINT;
    nReservationId : UDINT := 0;
END_VAR
VAR_OUTPUT
    eStatus         : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

Name	Type	Description
SendStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token with which the transaction is to be started.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nMeterStart	UDINT	Value in watt-hours at the start of the transaction. If the value is 0, the value is not checked. For all values greater than 0, this value must be greater than or equal to the last nMeterStop value.
nReservationId	UDINT	Optional reservation ID.

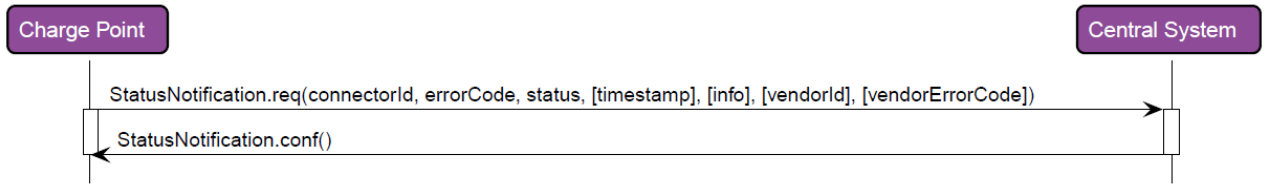
Outputs

Name	Type	Description
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response from the OCPP server.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.56 SendStatusNotification

With this method, an OCPP client sends a Status Notification to the corresponding OCPP server. The response from the OCPP server is processed directly within the method, but in this case it does not contain any separate variables anyway. The timestamp is set internally when the request is sent and cannot be set manually when the method is called.



Syntax

```
METHOD SendStatusNotification : BOOL
VAR_INPUT
    nConnectorId : UDINT;
    eError       : E_OCPP1_ChargePointError;
    eStatus      : E_OCPP1_ChargePointStatus;
    sInfo        : STRING(50) := '';
    sVendorError : STRING(50) := '';
    sVendorId    : STRING(255) := '';
END_VAR
```

Return value

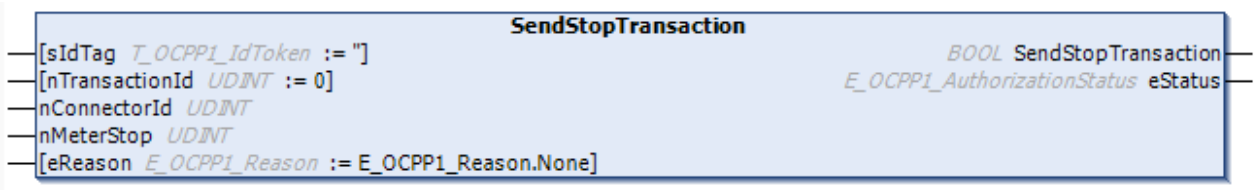
Name	Type	Description
SendStatusNotificationEx	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

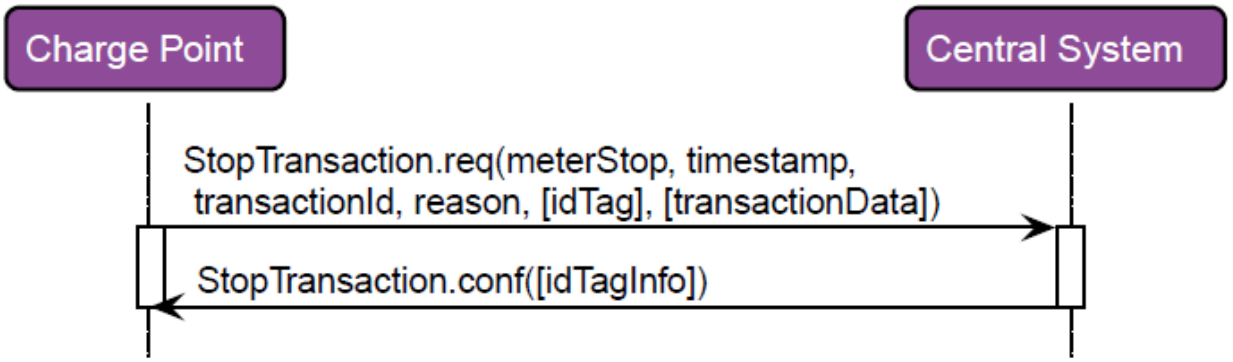
Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
eError	E_OCPP1_ChargePointError [▶ 190]	Error code to be sent in the Status Notification.
eStatus	E_OCPP1_ChargePointStatus [▶ 191]	Status to be sent in the Status Notification.
sInfo	STRING(50)	Contains optional, freely definable additional information on the error.
sVendorError	STRING(50)	Optionally contains the vendor-specific error code.
sVendorId	STRING(255)	Optionally contains the identifier for the vendor-specific implementation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.1.57 SendStopTransaction



With this method, an OCPP client sends a Stop Transaction request to the corresponding OCPP server. The response from the OCPP server is processed directly within the method.



Syntax

```
METHOD SendStopTransaction : BOOL
VAR_INPUT
    sIdTag          : T_OCPP1_IdToken := '';
    nTransactionId  : UDINT := 0;
    nConnectorId    : UDINT;
    nMeterStop      : UDINT;
    eReason         : E_OCPP1_Reason := E_OCPP1_Reason.None;
END_VAR
VAR_OUTPUT
    eStatus         : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

Name	Type	Description
SendStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token for which the transaction is to be stopped.
nTransactionId	UDINT	Alternatively, contains the Transaction ID received at Start Transaction if the Connector ID is 0.
nConnectorId	UDINT	ID of the Connector of a Charge Point if the Transaction ID is 0.
nMeterStop	UDINT	Value in watt-hours at the end of the transaction. This value must be greater than or equal to the nMeterStart value with which the transaction was started.
eReason	E_OCPP1_Reason [▶ 195]	Optionally contains the reason for stopping the transaction.

Outputs

Name	Type	Description
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response from the OCPP server.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2 FB_OCPP1_Server

This function block represents an OCPP server to which any number of OCPP clients can be connected. An OCPP client is identified via an ID. If a 1-to-1 relationship between OCPP client and OCPP server is required, the function block FB_OCPP1_Station [▶ 131] can be used instead of this function block.

The Send methods are used when sending requests to the client. In these methods, the response from the client is processed directly and stored in the output parameters of the methods.

If, on the other hand, a request is received from the client using one of the Receive methods, the response must be sent using the appropriate Response method.

Syntax

```
FUNCTION_BLOCK FB_OCPP1_Server
VAR_OUTPUT
    bValid              : BOOL;
```

```
    bBusy               : BOOL;
    bError              : BOOL;
    eErrorResult        : HRESULT;
    eErrorAction        : E_OCPP1_Action;
END_VAR
```

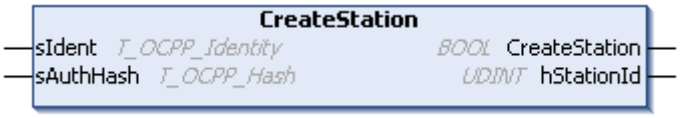
Outputs

Name	Type	Description
bValid	BOOL	The interface to the driver in the background exists.
bBusy	BOOL	Is TRUE as long as the function block is busy with processing.
bError	BOOL	Becomes TRUE when an error situation occurs.
eErrorResult	HRESULT	Last error present on the function block.
eErrorAction	E_OCPP1_Action [▶ 189]	OCPP command for which the error occurred.

Properties

Name	Type	Access	Description
IsOpen	BOOL	Get	Connections can be established.
IsPending	BOOL	Get	Waiting for completion of the request.

5.1.2.1 CreateStation



This method is used to create a new station in the OCPP server.

Syntax

```
METHOD CreateStation : HRESULT
VAR_INPUT
    sIdent      : T_OCPP_Identity;
    sAuthHash   : T_OCPP_Hash;
END_VAR
VAR_OUTPUT
    hStationId : UDINT;
END_VAR
```

Return value

Name	Type	Description
CreateStation	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

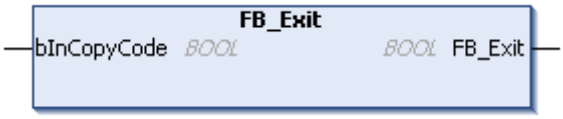
Name	Type	Description
sIdent	T_OCPP_Identity [▶ 209]	Identity of the OCPP client to be connected.
sAuthHash	T_OCPP_Hash [▶ 209]	Hash value for the respective OCPP client. The calculation is explained at Hash calculation [▶ 211].

Outputs

hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.2 FB_Exit



Syntax

```
METHOD FB_Exit : BOOL
VAR_INPUT
    bInCopyCode : BOOL;
END_VAR
```

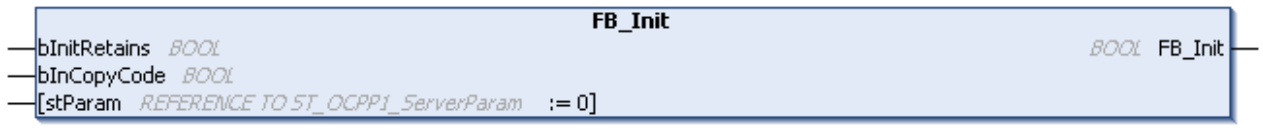
Return value

Name	Type	Description
FB_Exit	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bInCopyCode	BOOL	If TRUE, the Exit method is called to exit an instance, which is then copied (online change).

5.1.2.3 FB_Init



Syntax

```
METHOD FB_Init : BOOL
VAR_INPUT
    bInitRetains : BOOL;
    bInCopyCode  : BOOL;
    stParam      : REFERENCE TO ST_OCPP1_ServerParam REF=0;
END_VAR
```

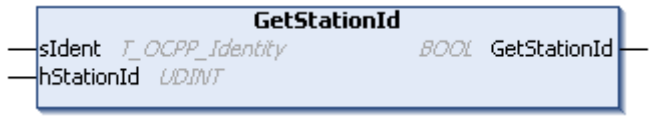
Return value

Name	Type	Description
FB_Init	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bInitRetains	BOOL	If TRUE, the Retain variables are initialized (warm start/cold start).
bInCopyCode	BOOL	If TRUE, the Exit method is called to exit an instance, which is then copied (online change).
stParam	REFERENCE TO ST_OCPP1_ServerParam [▶ 201]	Parameters for the OCPP server.

5.1.2.4 GetStationId



This method can be used to query the StationId of an OCPP client in the instance of the OCPP server.

Syntax

```
METHOD GetStationId : BOOL
VAR_INPUT
    sIdent     : T_OCPP_Identity;
END_VAR
VAR_IN_OUT
    hStationId : UDINT;
END_VAR
```

Return value

Name	Type	Description
GetStationId	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

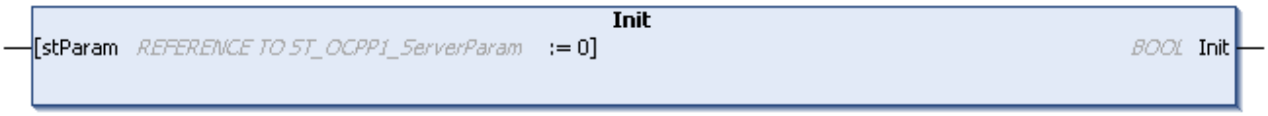
Inputs

Name	Type	Description
sIdent	T_OCPP_Identity [▶ 209]	Identity of the OCPP client.

Inputs/outputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

5.1.2.5 Init



This method is called when the function block is initialized in the FB_Init [▶ 23] method and specifies the parameters of the WebSockets server. If the connection parameters are to be changed afterwards, this method can be called again while the application is running.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    stParam : REFERENCE TO ST_OCPP1_ServerParam REF=0;
END_VAR
```

Return value

Name	Type	Description
Init	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
stParam	REFERENCE TO ST_OCPP1_ServerParam [▶ 201]	Connection parameters for the OCPP server.

5.1.2.6 PollRequest

This method must be called in order to receive incoming requests from OCPP clients.

Syntax

```
METHOD PollRequest : BOOL
VAR_OUTPUT
    hStationId : UDINT;
    eAction    : E_OCPP1_Action;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

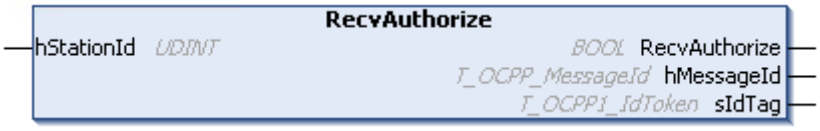
Name	Type	Description
PollRequest	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

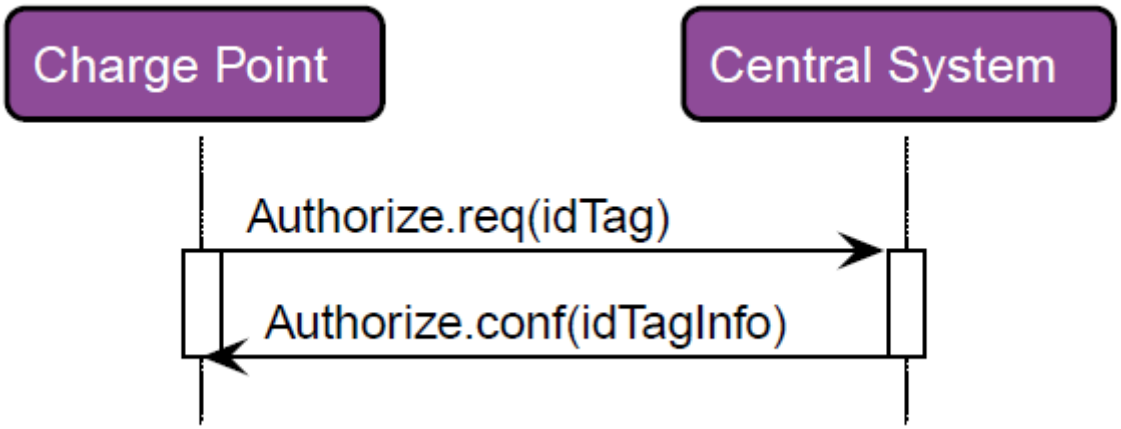
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
eAction	E_OCPP1_Action [▶ 189]	Type of OCPP request received from the server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.7 RecvAuthorize



With this method, an OCPP server receives an Authorize request from an OCPP client. To respond to the request, the method RespAuthorize [▶ 96] must be called.



Syntax

```
METHOD RecvAuthorize : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    sIdTag     : T_OCPP1_IdToken;
END_VAR
```

Return value

Name	Type	Description
RecvAuthorize	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

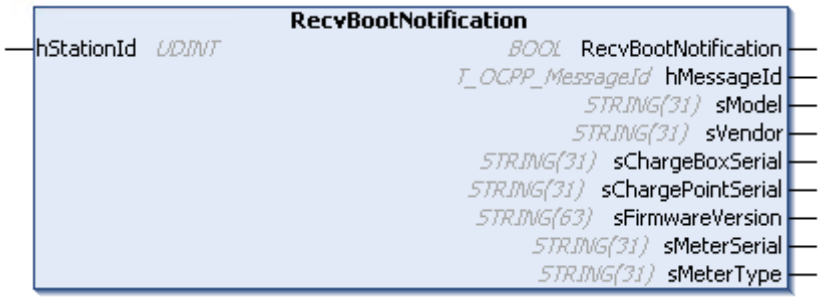
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

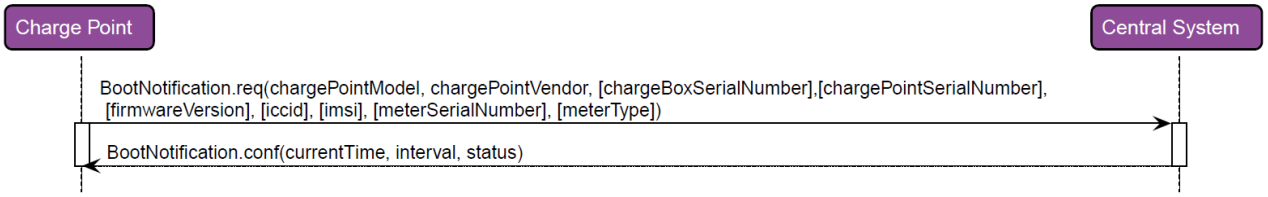
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token with which the Charge Point wants to be authorized on the Central System.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.8 RecvBootNotification



With this method, an OCPP server receives a Boot Notification from an OCPP client. To respond to the Boot Notification, the method RespBootNotification [▶ 97] must be called.



Syntax

```
METHOD RecvBootNotification : BOOL
VAR_INPUT
    hStationId         : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId         : T_OCPP_MessageId;
    sModel             : STRING(31);
    sVendor            : STRING(31);
    sChargeBoxSerial   : STRING(31);
    sChargePointSerial : STRING(31);
    sFirmwareVersion   : STRING(63);
    sMeterSerial       : STRING(31);
    sMeterType         : STRING(31);
END_VAR
```

Return value

Name	Type	Description
RecvBootNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

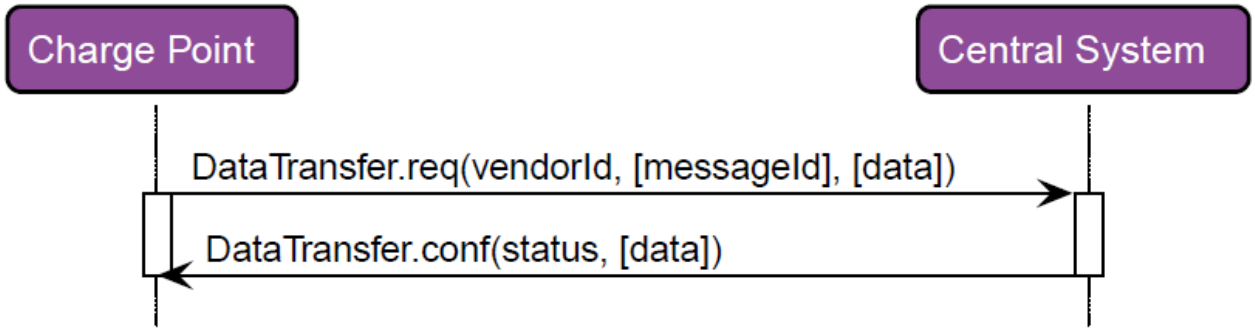
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sModel	STRING(31)	Model of the Charge Point.
sVendor	STRING(31)	Vendor of the Charge Point.
sChargeBoxSerial	STRING(31)	Serial number of the Charge Box within the Charge Point.
sChargePointSerial	STRING(31)	Serial number of the Charge Point.
sFirmwareVersion	STRING(63)	Firmware version of the Charge Point.
sMeterSerial	STRING(31)	Serial number of the main electricity meter of the Charge Point.
sMeterType	STRING(31)	Type of main electricity meter of the Charge Point

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.9 RecvDataTransfer

With this method, an OCPP server receives a Data Transfer request from an OCPP client. To respond to the request, the method RespDataTransfer [▶ 98] must be called.



Syntax

```
METHOD RecvDataTransfer : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    sVendorId  : STRING(255);
    sMessageId : STRING(63);
END_VAR
VAR_IN_OUT
    sData      : T_OCPP1_DataTransferData;
END_VAR
```

Return value

Name	Type	Description
RecvDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

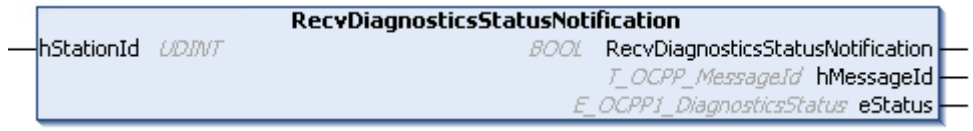
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sVendorId	STRING(255)	Identifier for the vendor-specific implementation.
sMessageId	STRING(63)	Identifier for the individual message.

Inputs/outputs

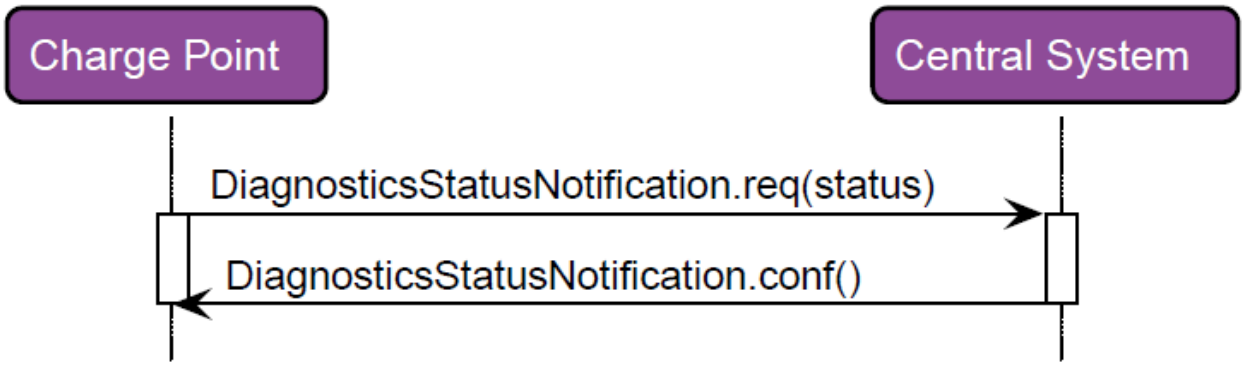
Name	Type	Description
sData	T_OCPP1_DataTransferData [▶ 208]	Text without specified length and format.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.10 RecvDiagnosticsStatusNotification



With this method, an OCPP server receives a Diagnostics Status Notification from an OCPP client. To respond to the Diagnostics Status Notification, the method RespDiagnosticsStatusNotification [▶ 99] must be called.



Syntax

```
METHOD RecvDiagnosticsStatusNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_DiagnosticsStatus;
END_VAR
```

Return value

Name	Type	Description
RecvDiagnosticsStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

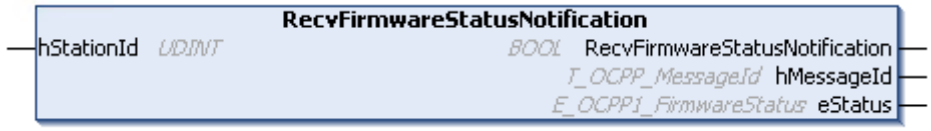
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

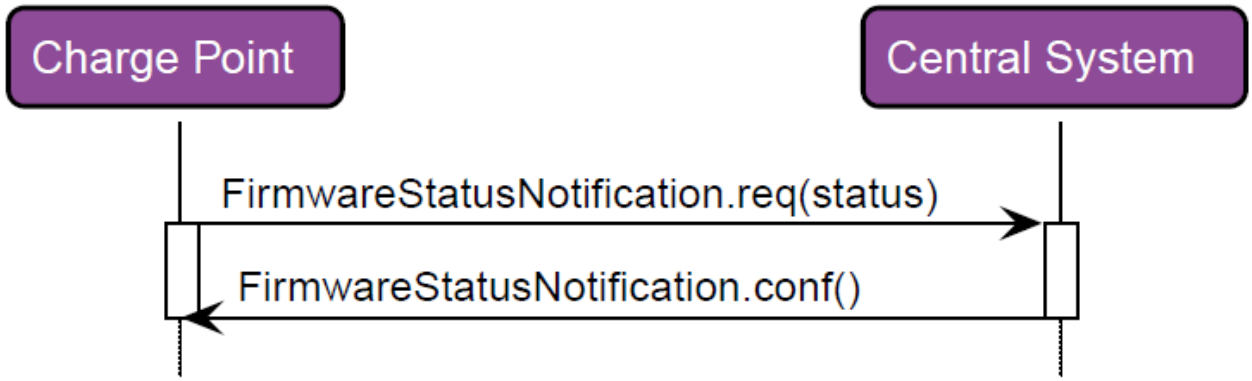
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_DiagnosticsStatus [▶ 193]	Status of the upload of the diagnostic data.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.11 RecvFirmwareStatusNotification



With this method, an OCPP server receives a Firmware Status Notification from an OCPP client. To respond to the Firmware Status Notification, the method RespFirmwareStatusNotification [▶ 101] must be called.



Syntax

```
METHOD RecvFirmwareStatusNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_FirmwareStatus;
END_VAR
```

Return value

Name	Type	Description
RecvFirmwareStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

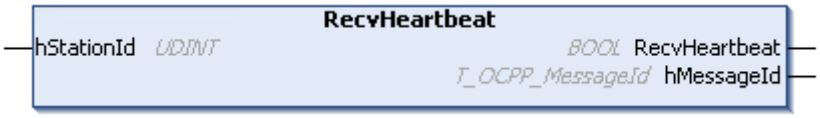
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

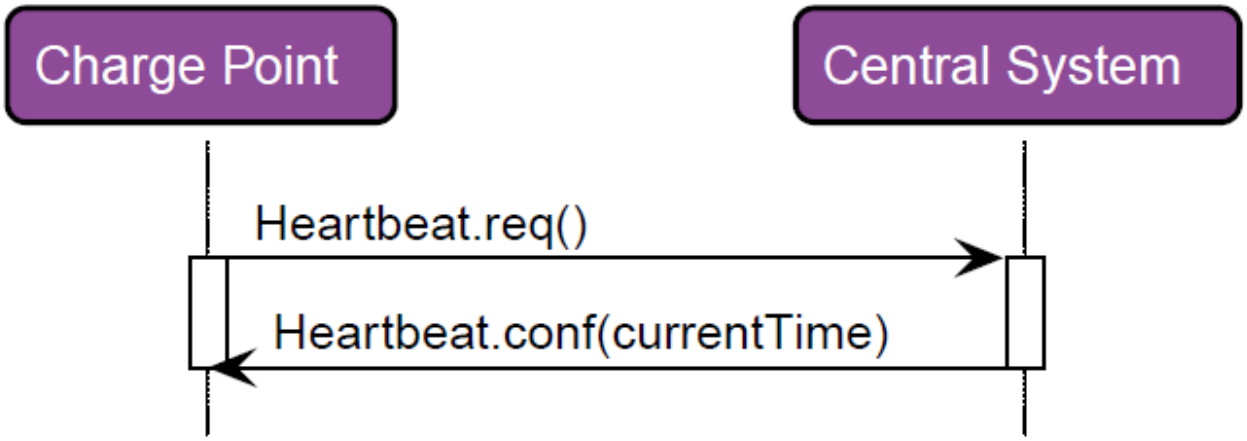
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_FirmwareStatus [▶ 193]	Status of a firmware installation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.12 RecvHeartbeat



With this method, an OCPP server receives a Heartbeat from an OCPP client. To respond to the Heartbeat, the method RespHeartbeat [▶ 102] must be called.



Syntax

```
METHOD RecvHeartbeat : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RecvHeartbeat	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

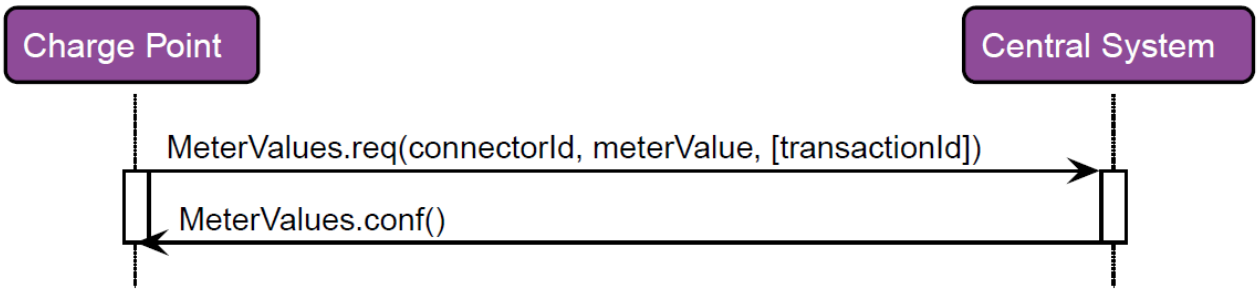
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.13 RecvMeterValue

With this method, an OCPP server receives Meter Values from an OCPP client. To respond to the receipt of the Meter Values, the method RespMeterValue [▶ 103] must be called.

Contrary to the specification, it is possible to receive Meter Value messages without Meter Values in order to increase compatibility with other vendors.



Syntax

```
METHOD RecvMeterValue : BOOL
VAR_INPUT
    hStationId      : UDINT;
END_VAR
VAR_IN_OUT
    arrSampleValues : ARRAY[*] OF ST_OCPP1_SampledValue;
END_VAR
VAR_OUTPUT
    hMessageId      : T_OCPP_MessageId;
    nConnectorId    : UDINT;
    nTransactionId  : UDINT;
    nSampleCount    : UDINT;
END_VAR
```

Return value

Name	Type	Description
RecvMeterValue	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Inputs/outputs

Name	Type	Description
arrSampleValues	ARRAY [*] OF ST_OCPP1_SampledValue [▶ 207]	The sampled values sent by the client.

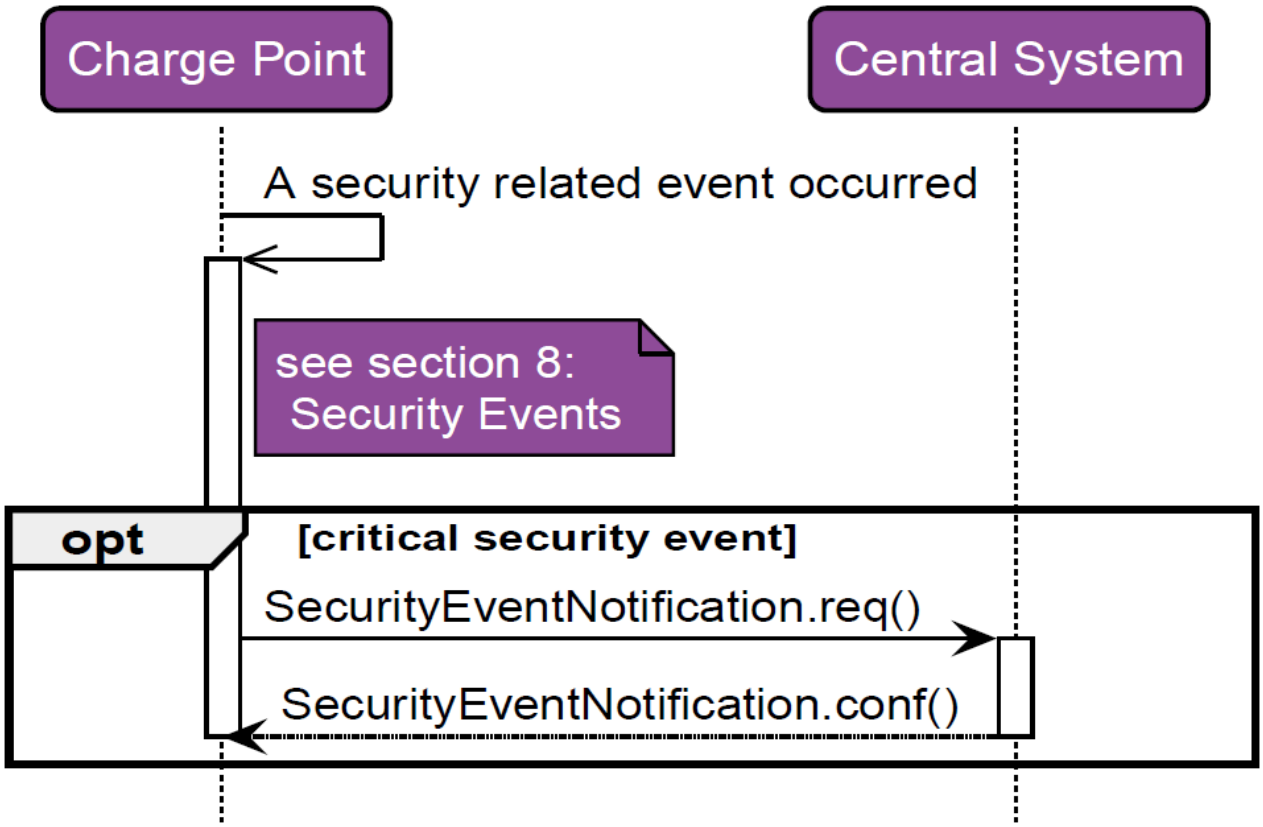
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nTransactionId	UDINT	The transaction to which the sampled values belong.
nSampleCount	UDINT	The number of sampled values contained.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.14 RecvSecurityEventNotification

With this method, an OCPP server receives a Security Event Notification from an OCPP client. To respond to the Security Event Notification, the method RespSecurityEventNotification [▶ 104] must be called.



Syntax

```
METHOD RecvSecurityEventNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    nTimestamp : ULINT;
    sType      : STRING(63);
    sInfo      : STRING(256);
END_VAR
```

Return value

Name	Type	Description
RecvSecurityEventNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

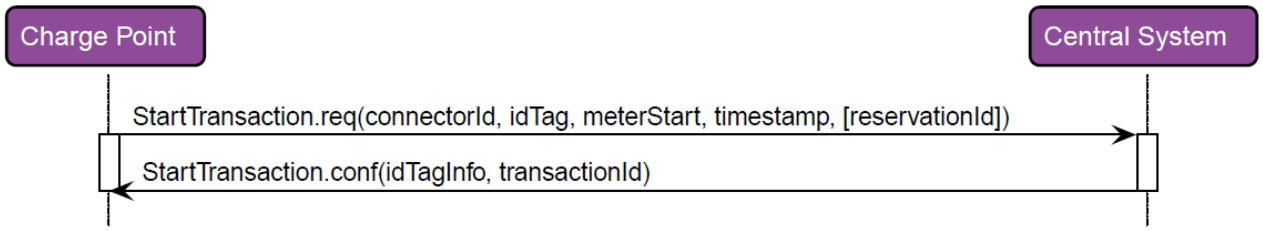
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nTimestamp	ULINT	Date and time when the event occurs.
sType	STRING(63)	Type of the Security Event according to the OCPP specification.
sInfo	STRING(256)	Additional information about the Security Event that has occurred.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.15 RecvStartTransaction

With this method, an OCPP server receives a Start Transaction request from an OCPP client. To respond to the request, the method RespStartTransaction [▶ 105] must be called.



Syntax

```
METHOD RecvStartTransaction : BOOL
VAR_INPUT
    hStationId     : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    sIdTag         : T_OCPP1_IdToken;
    nConnectorId   : UDINT;
    nMeterStart    : UDINT;
    nReservationId : UDINT;
    nTimestamp     : ULINT;
END_VAR
```

Return value

Name	Type	Description
RecvStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

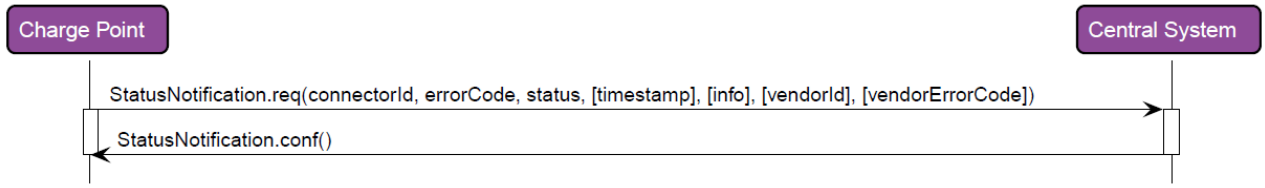
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token with which the transaction is to be started.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nMeterStart	UDINT	Value in watt-hours at the start of the transaction.
nReservationId	UDINT	Optional reservation ID.
nTimestamp	ULINT	Date and time at the start of the transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.16 RecvStatusNotification

With this method, an OCPP server receives a Status Notification from an OCPP client. To respond to the Status Notification, the method RespStatusNotification [▶ 106] must be called.



Syntax

```
METHOD RecvStatusNotification : BOOL
VAR_INPUT
    hStationId   : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId   : T_OCPP_MessageId;
    nConnectorId : UDINT;
    eError       : E_OCPP1_ChargePointError;
    eStatus      : E_OCPP1_ChargePointStatus;
    nTimestamp   : ULINT := 0;
    sInfo        : STRING(63) := '';
```

```
    sVendorError : STRING(63) := '';
    sVendorId    : STRING(255) := '';
END_VAR
```

Return value

Name	Type	Description
RecvStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

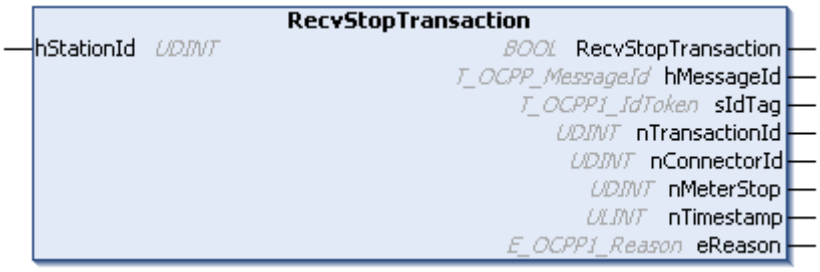
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

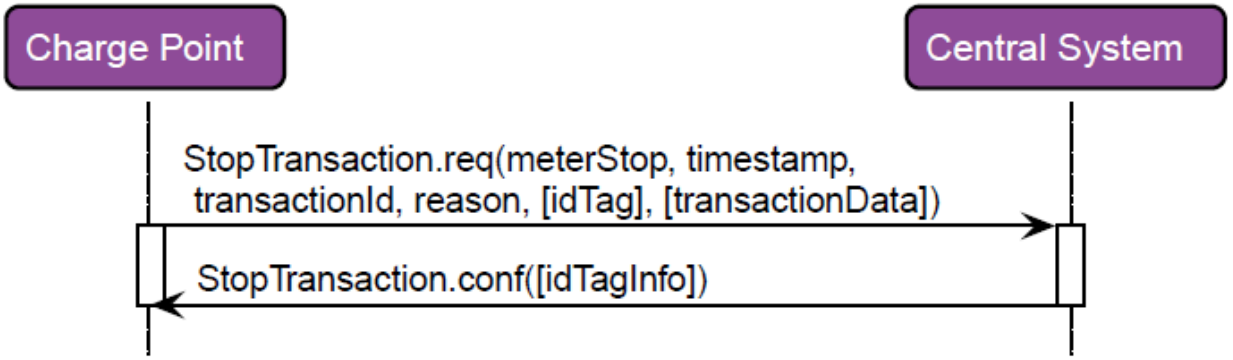
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
eError	E_OCPP1_ChargePointError [▶ 190]	Error code received in the Status Notification.
eStatus	E_OCPP1_ChargePointStatus [▶ 191]	Status received in the Status Notification.
nTimestamp	ULINT	The timestamp of the Status Notification.
sInfo	STRING(63)	Contains optional, freely definable additional information on the error.
sVendorError	STRING(63)	Optionally contains the vendor-specific error code.
sVendorId	STRING(255)	Optionally contains the identifier for the vendor-specific implementation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.17 RecvStopTransaction



With this method, an OCPP server receives a Stop Transaction request from an OCPP client. To respond to the request, the method RespStopTransaction [▶ 106] must be called.



Syntax

```
METHOD RecvStopTransaction : BOOL
VAR_INPUT
    hStationId     : UDINT;
END_VAR
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    sIdTag         : T_OCPP1_IdToken;
    nTransactionId : UDINT;
    nConnectorId   : UDINT;
    nMeterStop     : UDINT;
    nTimestamp     : ULINT;
    eReason        : E_OCPP1_Reason;
END_VAR
```

Return value

Name	Type	Description
RecvStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

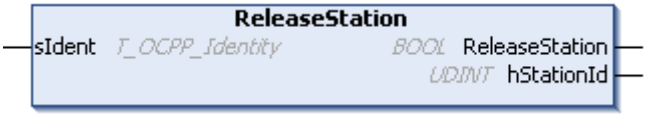
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token for which the transaction is to be stopped.
nTransactionId	UDINT	ID of the transaction to be stopped. This ID is contained in the Central System's response to a Start Transaction request.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nMeterStop	UDINT	Value in watt-hours at the end of the transaction.
nTimestamp	ULINT	Date and time when the transaction is stopped.
eReason	E_OCPP1_Reason [▶ 195]	Can optionally contain the reason for stopping the transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.18 ReleaseStation



This method is used to remove a connected station from the OCPP server.

Syntax

```
METHOD ReleaseStation : HRESULT
VAR_INPUT
    sIdent     : T_OCPP_Identity;
END_VAR
VAR_OUTPUT
    hStationId : UDINT;
END_VAR
```

Return value

Name	Type	Description
ReleaseStation	HRESULT	

Inputs

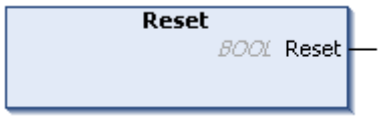
Name	Type	Description
sIdent	T_OCPP_Identity [▶ 209]	Identity of the OCPP client to be connected.

Outputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.19 Reset



This method is used to reset the last error present on the function block.

Syntax

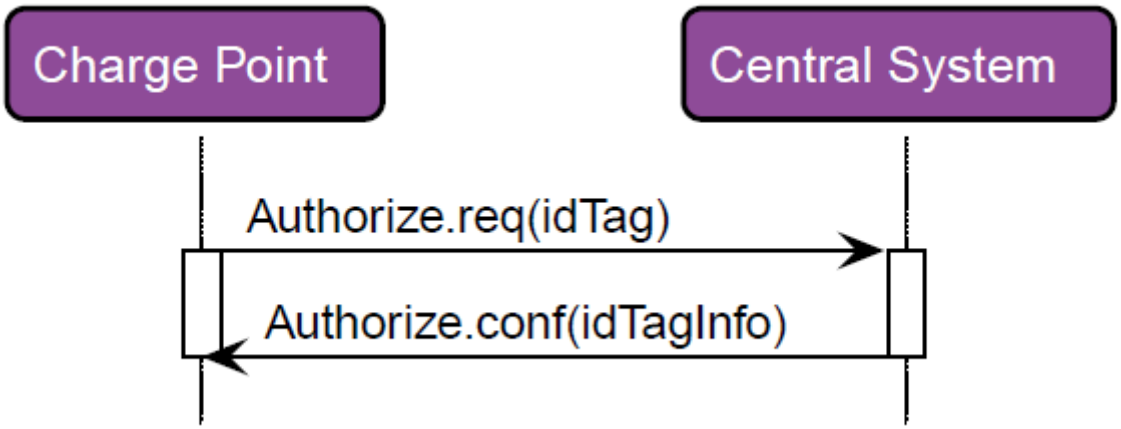
```
METHOD Reset : BOOL
```

Return value

Name	Type	Description
Reset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

5.1.2.20 RespAuthorize

With this method, an OCPP server responds to an Authorize request from an OCPP client.



Syntax

```
METHOD RespAuthorize : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

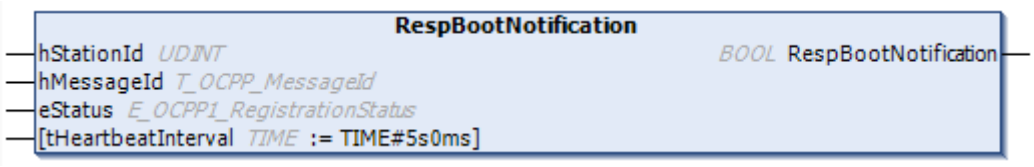
Name	Type	Description
RespAuthorize	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

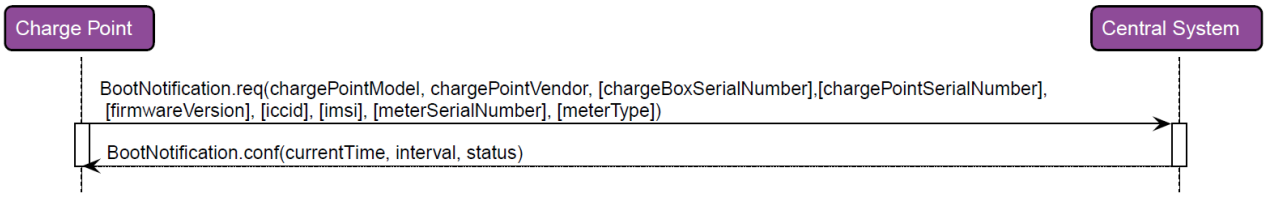
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response to the OCPP client.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.21 RespBootNotification



With this method, an OCPP server responds to a Boot Notification from an OCPP client.



Syntax

```
METHOD RespBootNotification : BOOL
VAR_INPUT
    hStationId         : UDINT;
    hMessageId         : T_OCPP_MessageId;
    eStatus            : E_OCPP1_RegistrationStatus;
    tHeartbeatInterval : TIME := T#5S;
END_VAR
```

Return value

Name	Type	Description
RespBootNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

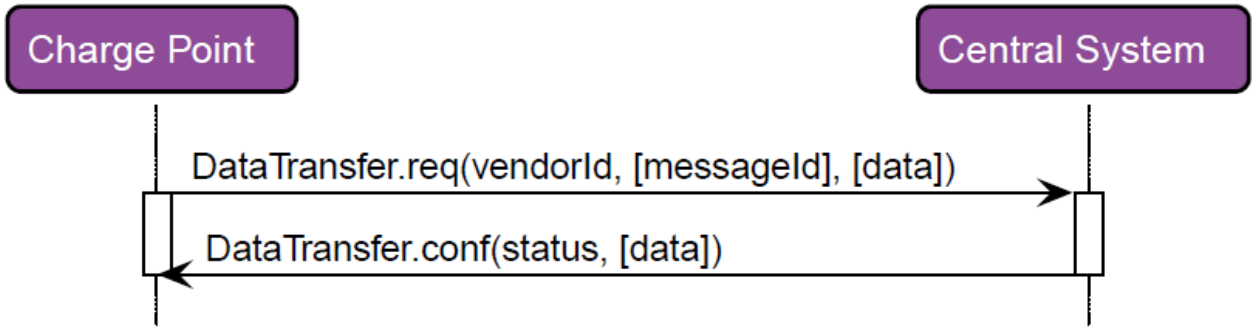
Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_RegistrationStatus [▶ 196]	Status of the registration in response to the OCPP client.
tHeartbeatInterval	TIME	If the registration status is Accepted, this variable contains the Heartbeat Interval for the connection.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.22 RespDataTransfer

With this method, an OCPP server responds to a Data Transfer request from an OCPP client.



Syntax

```
METHOD RespDataTransfer : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_DataTransferStatus;
END_VAR
```

Return value

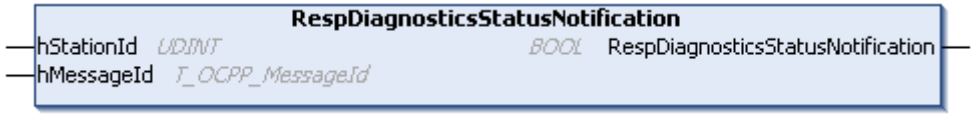
Name	Type	Description
RespDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

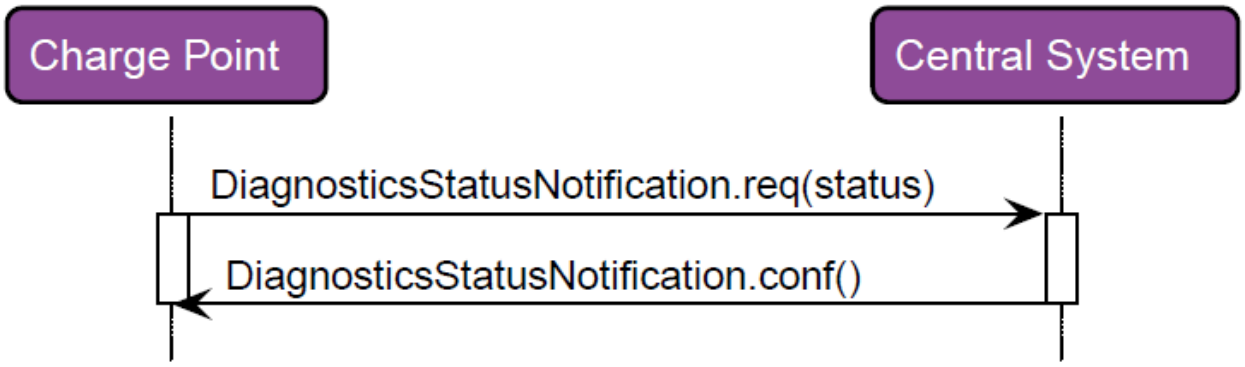
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_DataTransferStatus [▶ 193]	Status of the Data Transfer in response to the OCPP client.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.23 RespDiagnosticsStatusNotification



With this method, an OCPP server responds to a Diagnostics Status Notification from an OCPP client.



Syntax

```
METHOD RespDiagnosticsStatusNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

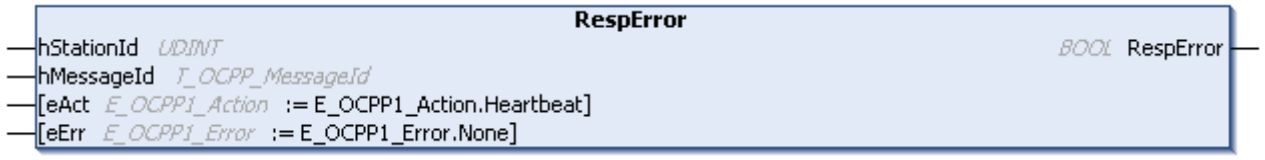
Name	Type	Description
RespDiagnosticsStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.24 RespError



With this method, an OCPP server responds to an OCPP client if the sent request has triggered an internal error. This may be the case, for example, if a requested action is not available.

Syntax

```
METHOD RespError : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
    eAct       : E_OCPP1_Action := E_OCPP1_Action.Heartbeat;
    eErr       : E_OCPP1_Error := E_OCPP1_Error.None;
END_VAR
```

Return value

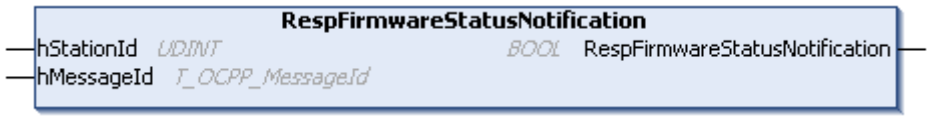
Name	Type	Description
RespError	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

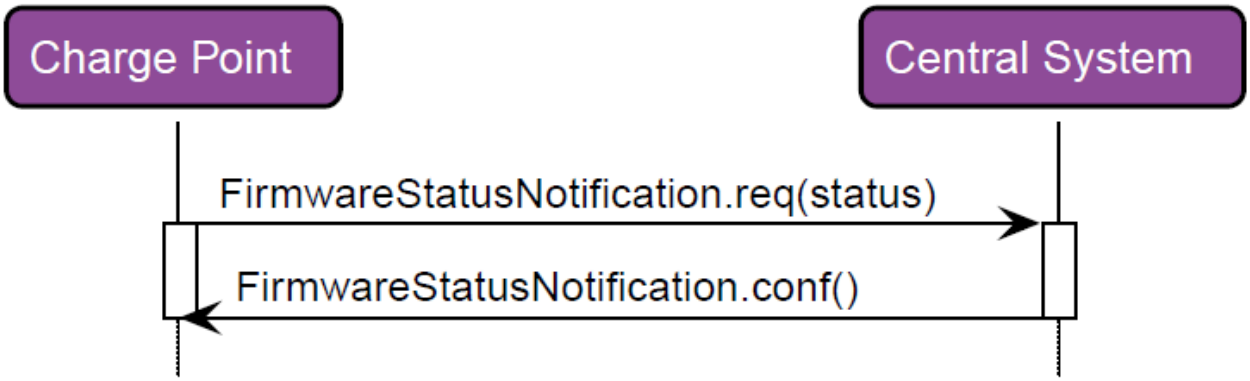
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eAct	E_OCPP1_Action [▶ 189]	Type of OCPP request for which the error occurred.
eErr	E_OCPP1_Error [▶ 193]	OCPP error that will be the response to the request.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.25 RespFirmwareStatusNotification



With this method, an OCPP server responds to a Firmware Status Notification from an OCPP client.



Syntax

```
METHOD RespFirmwareStatusNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

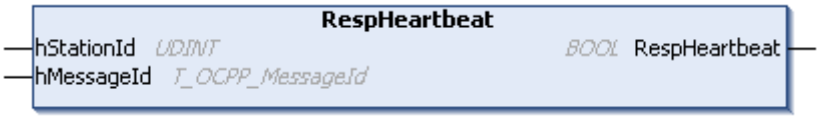
Name	Type	Description
RespFirmwareStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

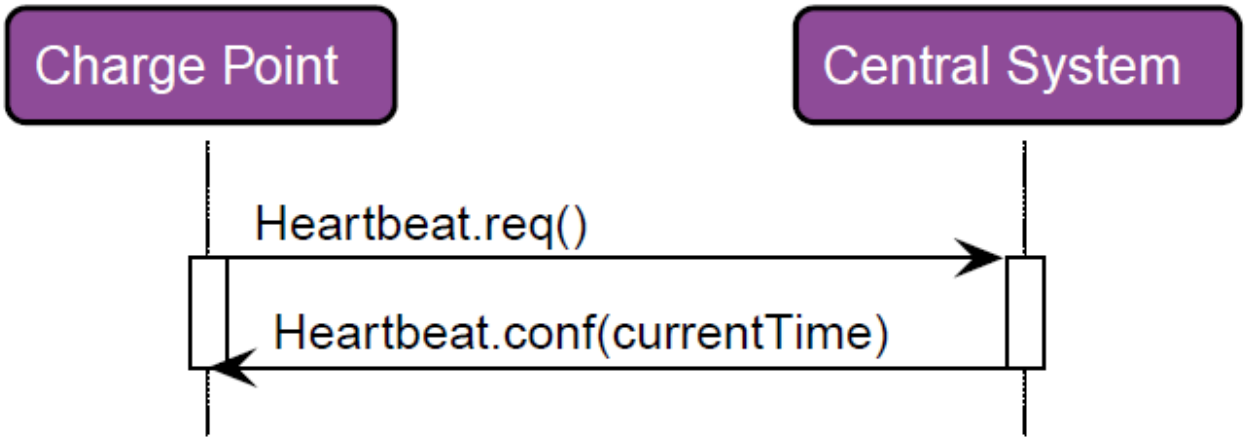
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.26 RespHeartbeat



With this method, an OCPP server responds to a Heartbeat from an OCPP client. The timestamp is set internally and does not need to be set separately in the PLC.



Syntax

```
METHOD RespHeartbeat : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RespHeartbeat	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

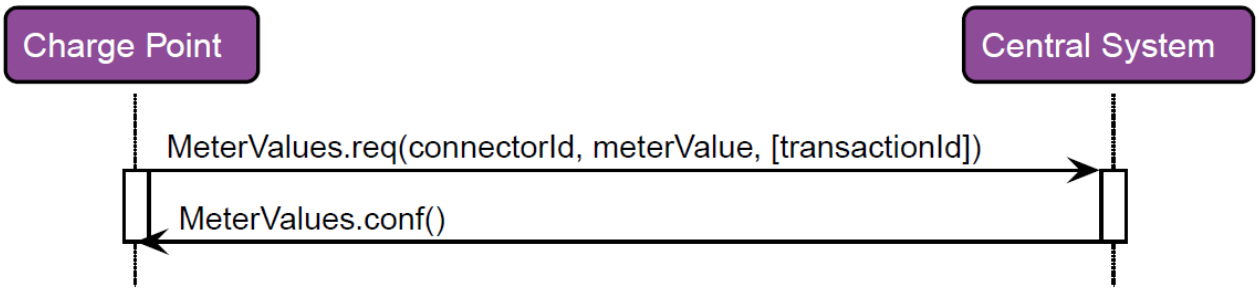
Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.27 RespMeterValue

With this method, an OCPP server responds to the sending of Meter Values from an OCPP client.



Syntax

```
METHOD RespMeterValue : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

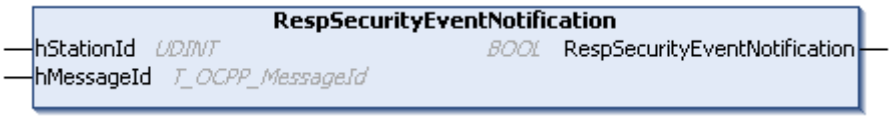
Name	Type	Description
RespMeterValue	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

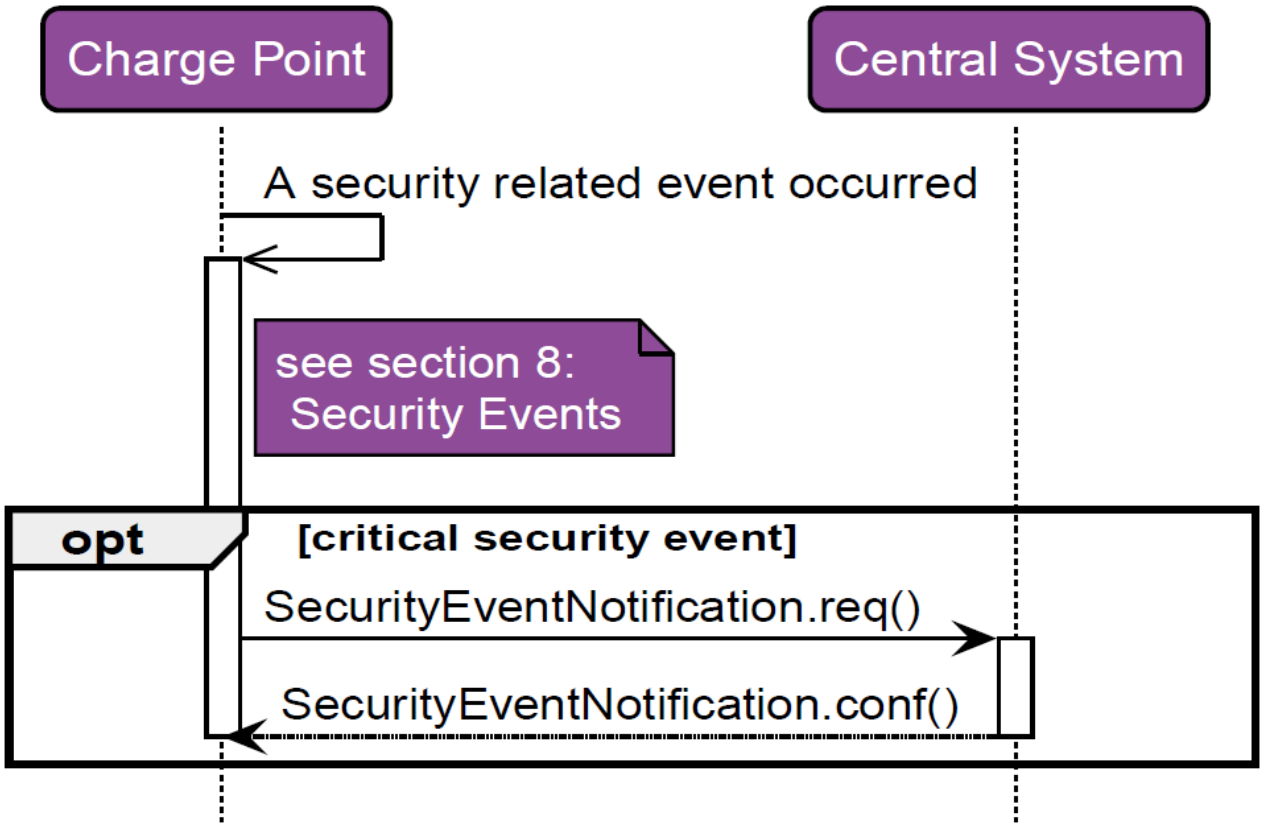
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.28 RespSecurityEventNotification



With this method, an OCPP server responds to a Security Event Notification from an OCPP client.



Syntax

```
METHOD RespSecurityEventNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

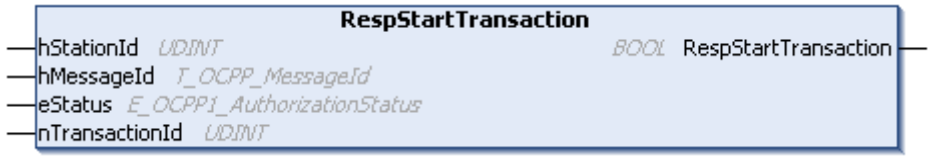
Name	Type	Description
RespSecurityEventNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

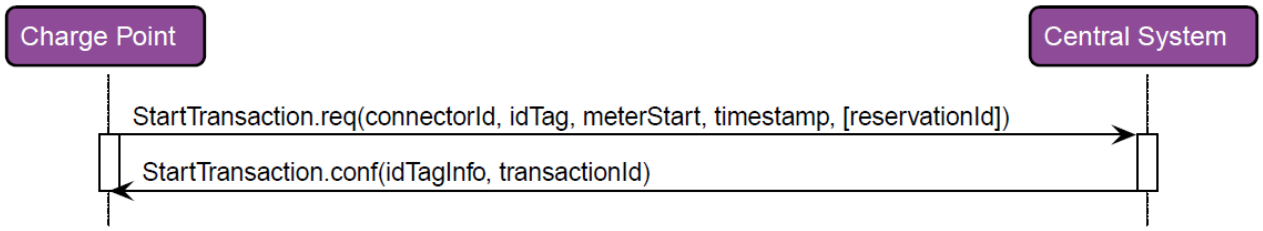
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.29 RespStartTransaction



With this method, an OCPP server responds to a Start Transaction request from an OCPP client.



Syntax

```
METHOD RespStartTransaction : BOOL
VAR_INPUT
    hStationId     : UDINT;
    hMessageId     : T_OCPP_MessageId;
    eStatus        : E_OCPP1_AuthorizationStatus;
    nTransactionId : UDINT;
END_VAR
```

Return value

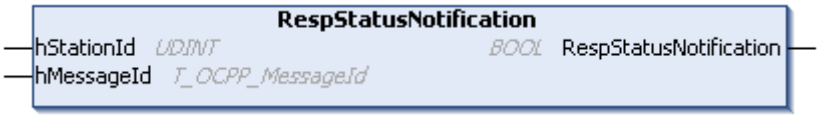
Name	Type	Description
RespStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

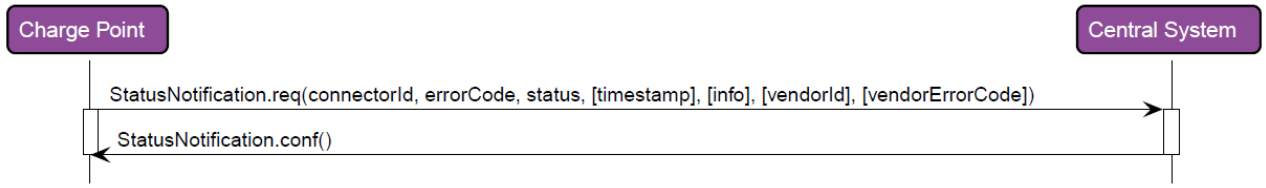
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response to the OCPP client.
nTransactionId	UDINT	ID defined by the Central System for the transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.30 RespStatusNotification



With this method, an OCPP server responds to a Status Notification from an OCPP client.



Syntax

```
METHOD RespStatusNotification : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RespStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

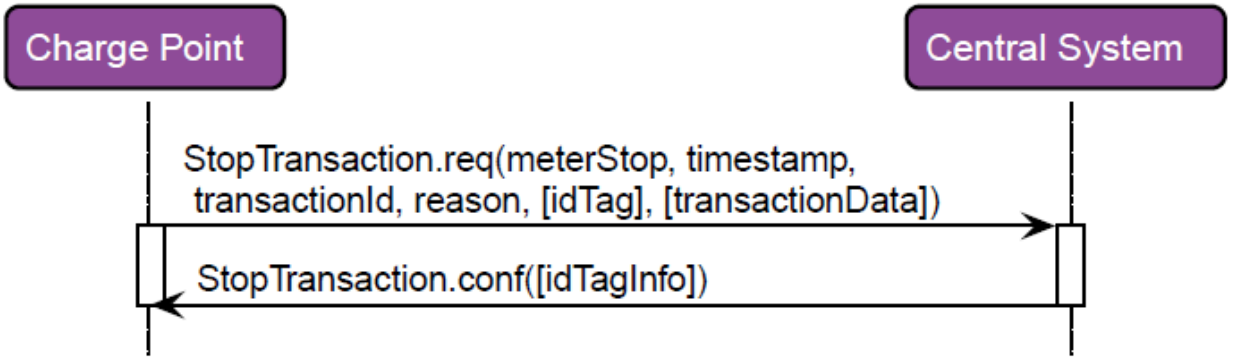
Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.31 RespStopTransaction

With this method, an OCPP server responds to a Stop Transaction request from an OCPP client.



Syntax

```
METHOD RespStopTransaction : BOOL
VAR_INPUT
    hStationId : UDINT;
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

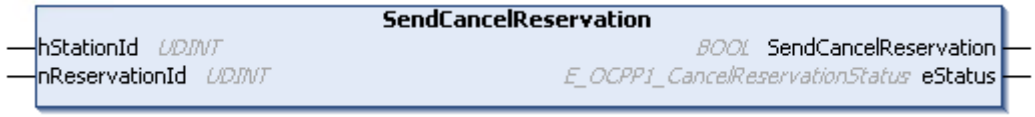
Name	Type	Description
RespStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

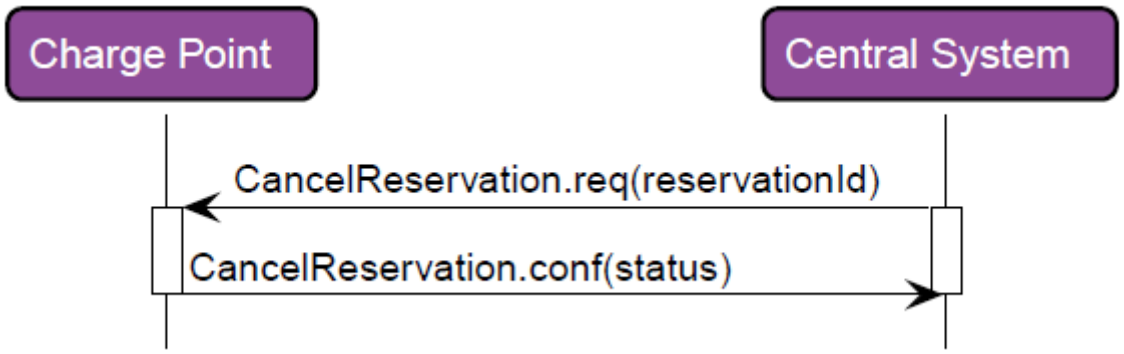
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response to the OCPP client.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.32 SendCancelReservation



With this method, an OCPP server sends a Cancel Reservation request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendCancelReservation : BOOL
VAR_INPUT
    hStationId     : UDINT;
    nReservationId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus        : E_OCPP1_CancelReservationStatus;
END_VAR
```

Return value

Name	Type	Description
SendCancelReservation	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nReservationId	UDINT	ID of the reservation to be canceled.

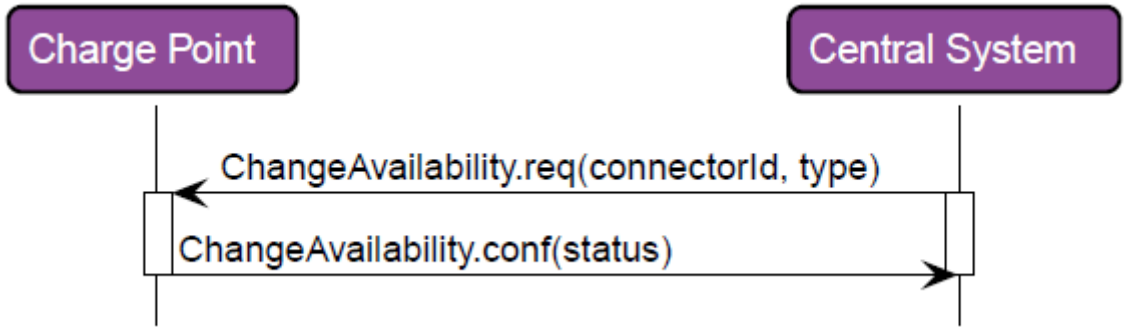
Outputs

Name	Type	Description
eStatus	E_OCPP1_CancelReservationStatus [▶ 190]	The status shows whether the reservation was successfully canceled.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.33 SendChangeAvailability

With this method, an OCPP server sends a Change Availability request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendChangeAvailability : BOOL
VAR_INPUT
    hStationId   : UDINT;
    nConnectorId : UDINT;
    eType        : E_OCPP1_AvailabilityType;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_AvailabilityStatus;
END_VAR
```

Return value

Name	Type	Description
SendChangeAvailability	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
eType	E_OCPP1_AvailabilityType [▶ 190]	Type of availability change that the Charge Point is to carry out.

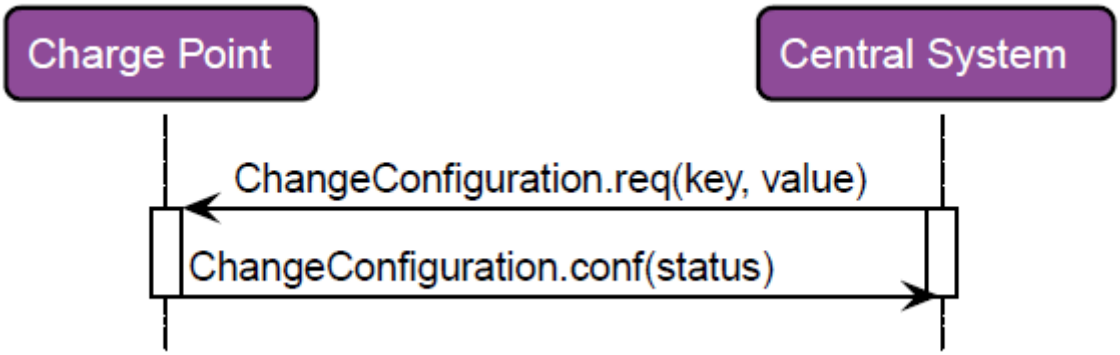
Outputs

Name	Type	Description
eStatus	E_OCPP1_AvailabilityStatus [▶ 190]	Response whether the Charge Point was able to implement the requested availability change.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.34 SendChangeConfiguration

With this method, an OCPP server sends a Change Configuration request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendChangeConfiguration : BOOL
VAR_INPUT
    hStationId : UDINT;
    sKey       : T_OCPP1_ConfigKey;
    sValue     : T_OCPP1_ConfigValue;
END_VAR
VAR_OUTPUT
    eStatus    : E_OCPP1_ConfigurationStatus;
END_VAR
```

Return value

Name	Type	Description
SendChangeConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

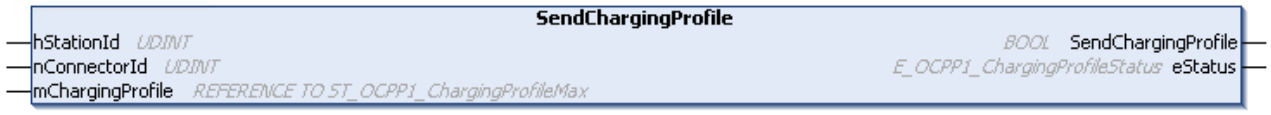
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
sKey	T_OCPP1_ConfigKey [▶ 208]	The name of the configuration setting to be changed.
sValue	T_OCPP1_ConfigValue [▶ 208]	The new value of the configuration setting.

Outputs

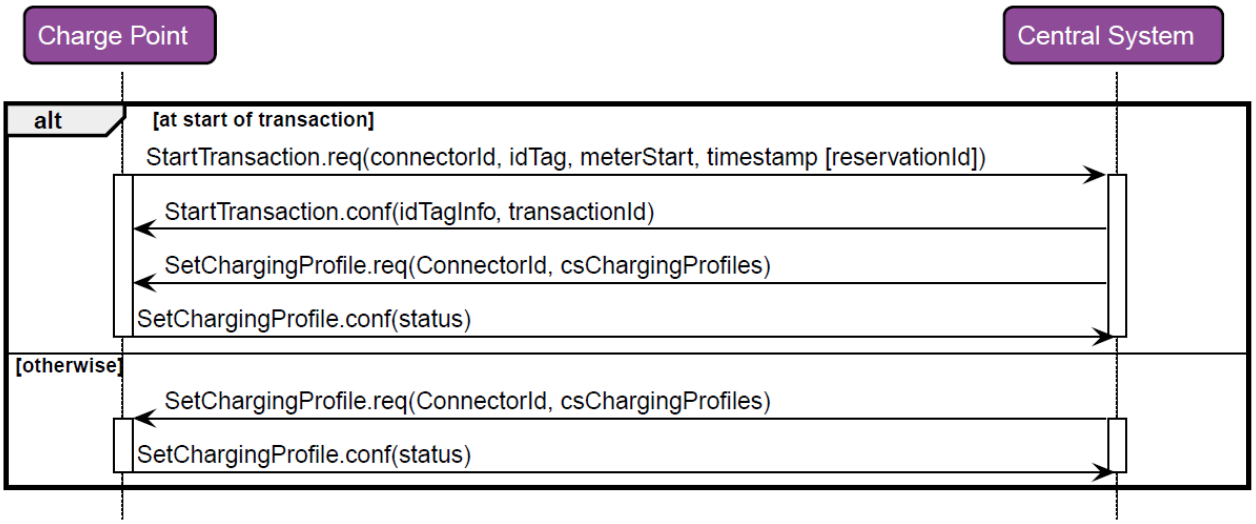
Name	Type	Description
eStatus	E_OCPP1_ConfigurationStatus [▶ 193]	The status indicates whether the configuration change has been accepted.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.35 SendChargingProfile



With this method, an OCPP server sends a Charging Profile to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendChargingProfile : BOOL
VAR_INPUT
    hStationId       : UDINT;
    nConnectorId     : UDINT;
    mChargingProfile : REFERENCE TO ST_OCPP1_ChargingProfileMax;
END_VAR
VAR_OUTPUT
    eStatus          : E_OCPP1_ChargingProfileStatus;
END_VAR
```

Return value

Name	Type	Description
SendChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

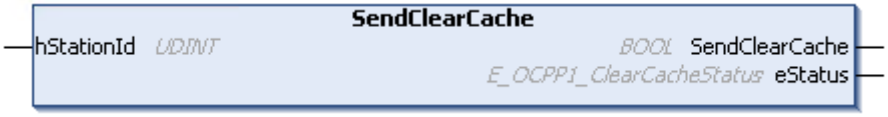
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
mChargingProfile	REFERENCE TO ST_OCPP1_ChargingProfileMax [▶ 204]	Charging Profile to be sent to the client.

Outputs

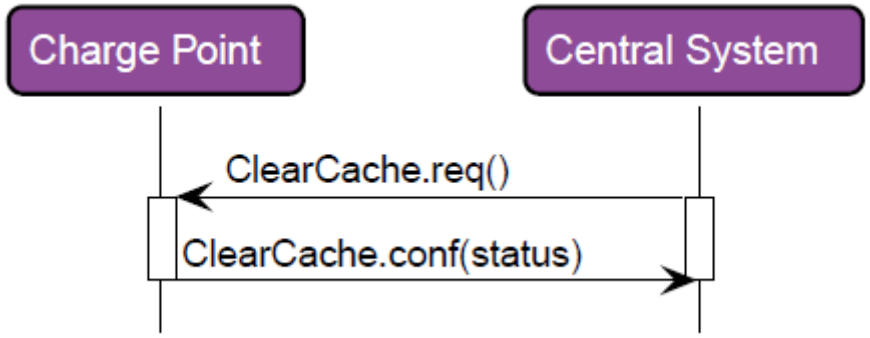
Name	Type	Description
eStatus	E_OCPP1_ChargingProfileStatus [▶ 191]	The status indicates whether the Charging Profile has been accepted by the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.36 SendClearCache



With this method, an OCPP server sends a Clear Cache request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendClearCache : BOOL
VAR_INPUT
    hStationId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus    : E_OCPP1_ClearCacheStatus;
END_VAR
```

Return value

Name	Type	Description
SendClearCache	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

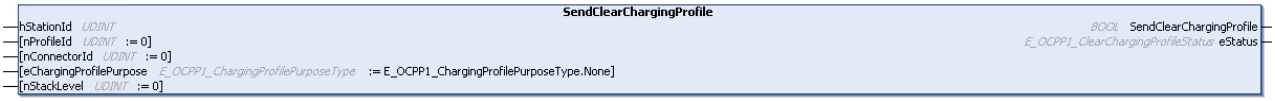
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Outputs

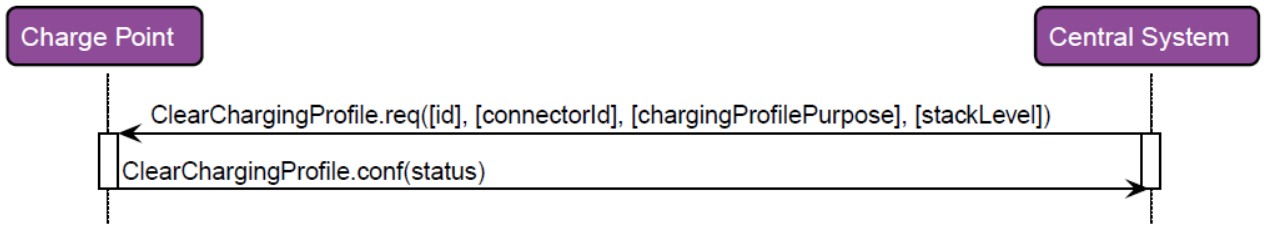
Name	Type	Description
eStatus	E_OCPP1_ClearCacheStatus [▶ 192]	The status indicates whether the clearing of the Cache Profile has been accepted by the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.37 SendClearChargingProfile



With this method, an OCPP server sends a Clear Charging Profile request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendClearChargingProfile : BOOL
VAR_INPUT
    hStationId              : UDINT;
    nProfileId              : UDINT := 0;
    nConnectorId            : UDINT := 0;
    eChargingProfilePurpose : E_OCPP1_ChargingProfilePurposeType := E_OCPP_ChargingProfilePurposeTyp
e.None;
    nStackLevel             : UDINT := 0;
END_VAR
VAR_OUTPUT
    eStatus                 : E_OCPP1_ClearChargingProfileStatus;
END_VAR
```

Return value

Name	Type	Description
SendClearChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

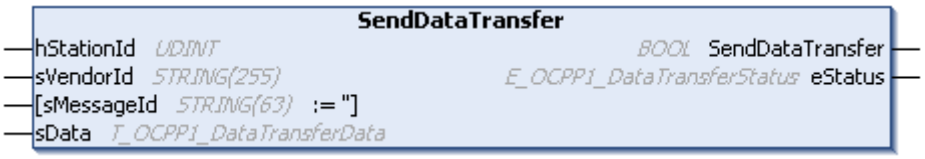
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nProfileId	UDINT	Identifier of the Charging Profile to be cleared.
nConnectorId	UDINT	ID of the Connector of a Charge Point. The value 0 indicates that the clearing relates to the entire Charge Point.
eChargingProfilePurpose	E_OCPP1_ChargingProfilePurposeType [▶ 191]	Specifies the purpose of the Charging Profiles to be cleared.
nStackLevel	UDINT	Specifies the StackLevel for which Charging Profiles are cleared.

Outputs

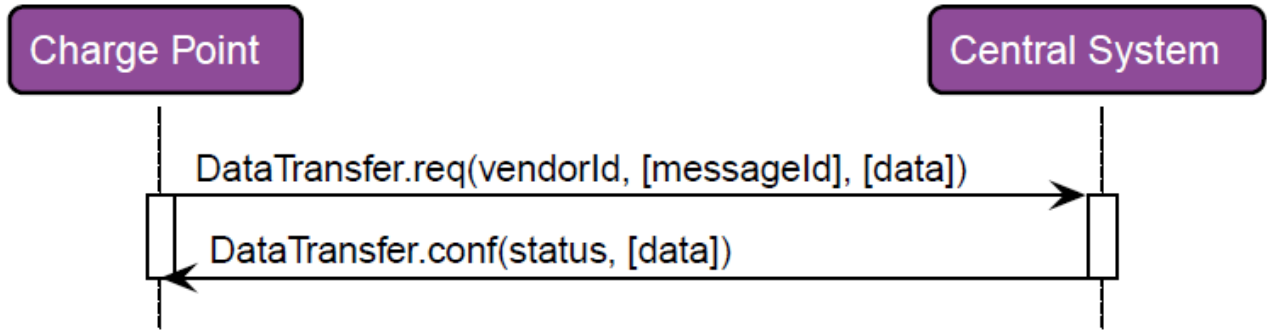
Name	Type	Description
eStatus	E_OCPP1_ClearChargingProfileStatus [▶ 192]	The status indicates whether the Charging Profile could be cleared from the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.38 SendDataTransfer



With this method, an OCPP server sends a Data Transfer request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendDataTransfer : BOOL
VAR_INPUT
    hStationId : UDINT;
    sVendorId  : STRING(255);
    sMessageId : STRING(63) := '';
END_VAR
VAR_IN_OUT CONSTANT
    sData      : T_OCPP1_DataTransferData;
END_VAR
VAR_OUTPUT
    eStatus    : E_OCPP1_DataTransferStatus;
END_VAR
```

Return value

Name	Type	Description
SendDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
sVendorId	STRING(255)	Identifier for the vendor, which identifies the vendor-specific implementation.
sMessageId	STRING(63)	Additional identification field for a single message.

Inputs/outputs

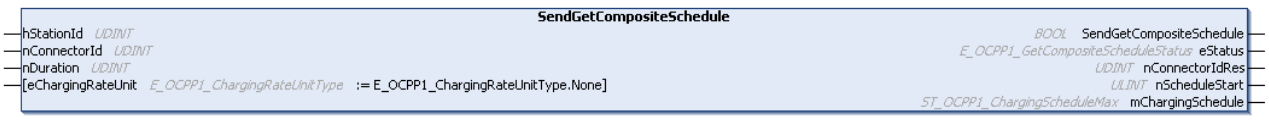
Name	Type	Description
sData	T_OCPP1_DataTransferData [▶ 208]	Text without specified length and format.

Outputs

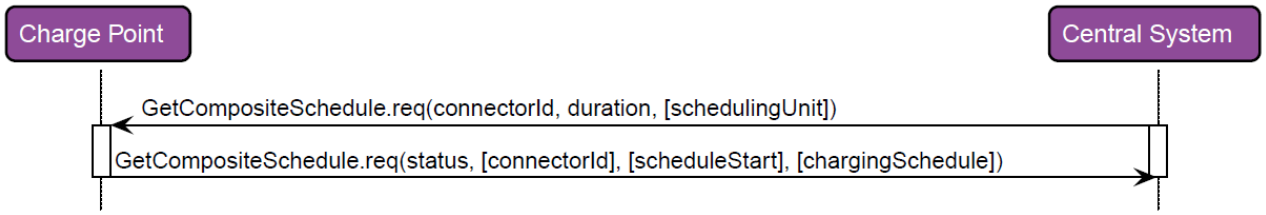
Name	Type	Description
eStatus	E_OCPP1_DataTransferStatus [▶ 193]	Status of the Data Transfer.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.39 SendGetCompositeSchedule



With this method, an OCPP server sends a Get Composite Schedule request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendGetCompositeSchedule : BOOL
VAR_INPUT
    hStationId        : UDINT;
    nConnectorId      : UDINT;
    nDuration         : UDINT;
    eChargingRateUnit : E_OCPP1_ChargingRateUnitType := E_OCPP_ChargingRateUnitType.None;
END_VAR
VAR_OUTPUT
    eStatus           : E_OCPP1_GetCompositeScheduleStatus;
    nConnectorIdRes   : UDINT;
    nScheduleStart    : ULINT := 0;
    mChargingSchedule : ST_OCPP1_ChargingScheduleMax;
END_VAR
```

Return value

Name	Type	Description
SendGetCompositeSchedule	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nDuration	UDINT	Length of the requested schedule.
eChargingRateUnit	E_OCPP1_ChargingRateUnitType [▶ 192]	Used to specify the unit for the request.

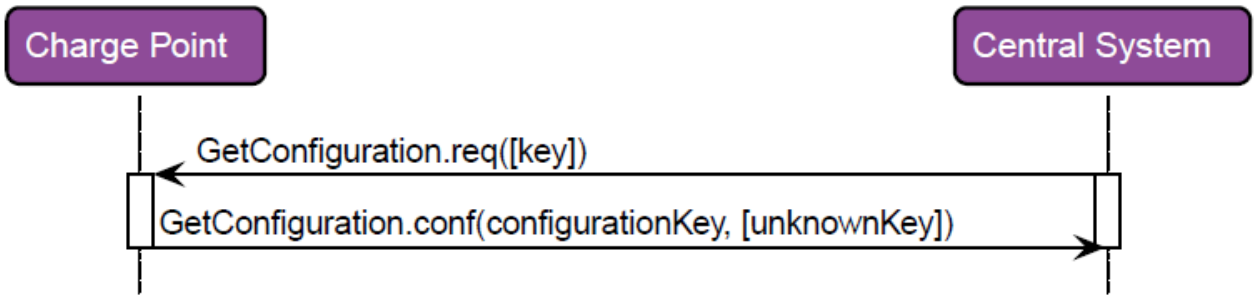
Outputs

Name	Type	Description
eStatus	E_OCPP1_GetCompositeScheduleStatus [▶ 194]	The status shows whether the schedule request could be answered by the Charge Point.
nConnectorIdRes	UDINT	Identifier of the Connector to which the schedule in the response refers.
nScheduleStart	ULINT	Start time of the schedule.
mChargingSchedule	ST_OCPP1_ChargingScheduleMax [▶ 204]	Requested schedule.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.40 SendGetConfiguration

With this method, an OCPP server sends a Get Configuration request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendGetConfiguration : BOOL
VAR_INPUT
    hStationId    : UDINT;
END_VAR
VAR_IN_OUT
    arrKeys       : ARRAY[*] OF ST_OCPP1_ConfigKeyValue;
END_VAR
VAR_OUTPUT
    nKeysCount    : UDINT := 0;
    nUnknownCount : UDINT := 0;
END_VAR
```

Return value

Name	Type	Description
SendGetConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

Inputs/outputs

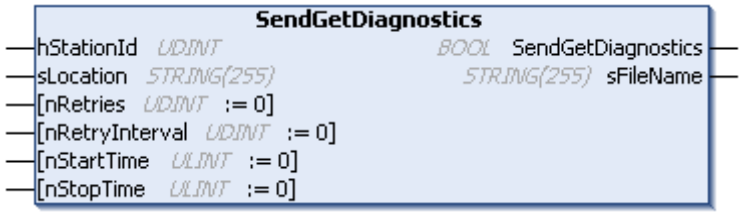
Name	Type	Description
arrKeys	ARRAY [*] OF ST_OCPP1_ConfigKeyValue [▶ 207]	List of requested or known configuration keys for which the configuration is requested.

Outputs

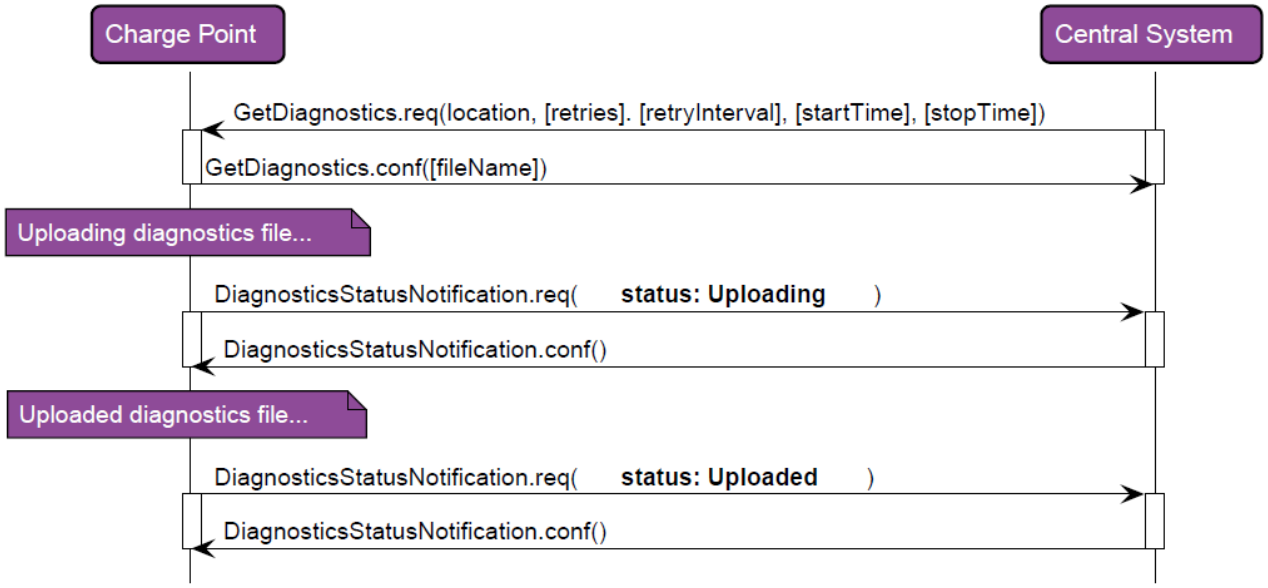
Name	Type	Description
nKeysCount	UDINT	Number of the following configuration keys.
nUnknownCount	UDINT	Number of the following unknown configuration keys.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.41 SendGetDiagnostics



With this method, an OCPP server sends a Get Diagnostics request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendGetDiagnostics : BOOL
VAR_INPUT
    hStationId      : UDINT;
    sLocation       : STRING(255);
    nRetries        : UDINT := 0;
    nRetryInterval  : UDINT := 0;
    nStartTime      : ULINT := 0;
    nStopTime       : ULINT := 0;
END_VAR
VAR_OUTPUT
    sFileName       : STRING(255);
END_VAR
```

Return value

Name	Type	Description
SendGetDiagnostics	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

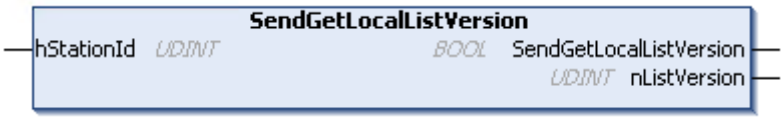
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
sLocation	STRING(255)	Directory to which the diagnostic file is to be uploaded.
nRetries	UDINT	Number of attempts made by the Charge Point if the upload fails.
nRetryInterval	UDINT	Interval after which a new upload is attempted.
nStartTime	ULINT	Start time of the logging information contained in the diagnostics.
nStopTime	ULINT	End time of the logging information contained in the diagnostics.

Outputs

Name	Type	Description
sFileName	STRING(255)	Contains the name of the diagnostic file that will be uploaded.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.42 SendGetLocalListVersion



With this method, an OCPP server sends a Get Local List Version request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.

Syntax

```
METHOD SendGetLocalListVersion : BOOL
VAR_INPUT
    hStationId   : UDINT;
END_VAR
VAR_OUTPUT
    nListVersion : UDINT;
END_VAR
```

Return value

Name	Type	Description
SendGetLocalListVersion	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.

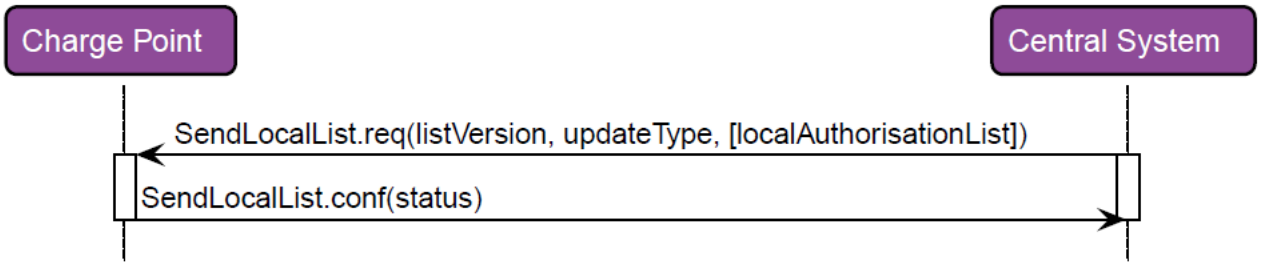
Outputs

Name	Type	Description
nListVersion	UDINT	Contains the version number of the Local Authorization List currently available in the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.43 SendLocalList

With this method, an OCPP server sends a Local List request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendLocalList : BOOL
VAR_INPUT
    hStationId   : UDINT;
    nListVersion : UDINT;
    eUpdateType  : E_OCPP1_UpdateType;
    nAuthCount   : UDINT := 0;
END_VAR
VAR_IN_OUT
    arrAuthList  : ARRAY[*] OF ST_OCPP1_AuthorizationData;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_UpdateStatus;
END_VAR
```

Return value

Name	Type	Description
SendLocalList	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nListVersion	UDINT	Version of the Local Authorization List provided as an update.
eUpdateType	E_OCPP1_UpdateType [▶ 197]	Definition of the update type, either as an update or as a complete replacement.
nAuthCount	UDINT	Number of entries in the Local Authorization List.

Inputs/outputs

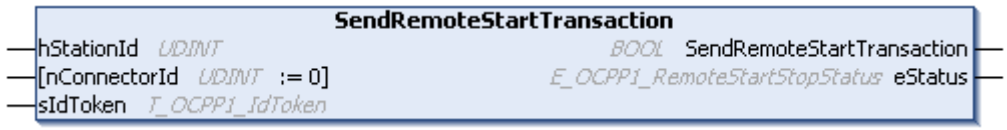
Name	Type	Description
arrAuthList	ARRAY [*] OF ST_OCPP1_AuthorizationData [▶ 202]	Either the values of the new Local Authorization List (complete replacement) or values to be added to the existing Local Authorization List (update).

Outputs

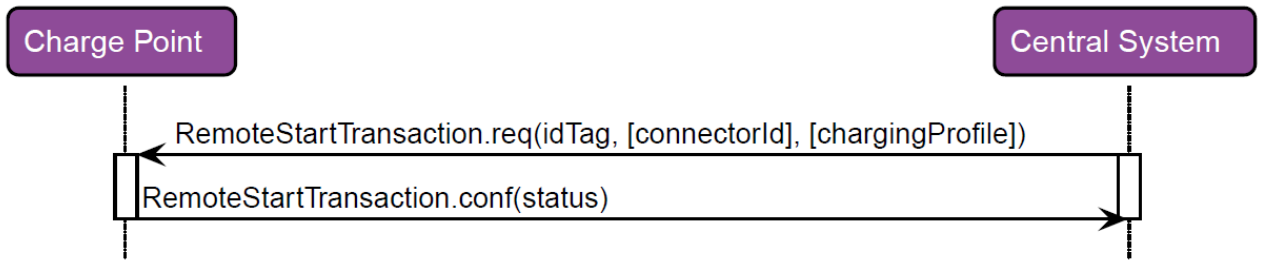
Name	Type	Description
eStatus	E_OCPP1_UpdateStatus [▶ 197]	The status indicates whether the Charge Point has received and carried out the Local Authorization List update.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.44 SendRemoteStartTransaction



With this method, an OCPP server sends a Remote Start Transaction request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendRemoteStartTransaction : BOOL
VAR_INPUT
    hStationId   : UDINT;
    nConnectorId : UDINT := 0;
    sIdToken     : T_OCPP1_IdToken;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

Name	Type	Description
SendRemoteStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
sIdToken	T_OCPP1_IdToken [▶ 209]	ID token of the Charge Point in the Central System.

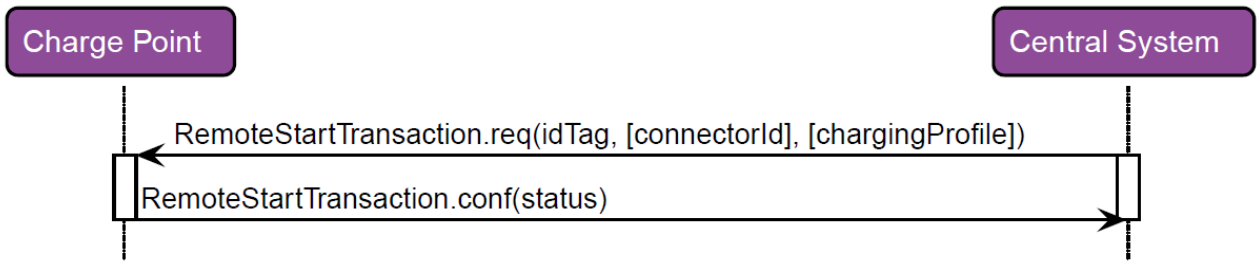
Outputs

Name	Type	Description
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status shows whether the Charge Point has accepted the request to start a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.45 SendRemoteStartTransactionProfile

With this method, an OCPP server sends a Remote Start Transaction request including a Charging Profile to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendRemoteStartTransactionProfile : BOOL
VAR_INPUT
    hStationId       : UDINT;
    nConnectorId     : UDINT := 0;
    sIdToken         : T_OCPP1_IdToken;
    mChargingProfile : REFERENCE TO ST_OCPP1_ChargingProfile := 0;
END_VAR
VAR_OUTPUT
    eStatus          : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

Name	Type	Description
SendRemoteStartTransactionProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

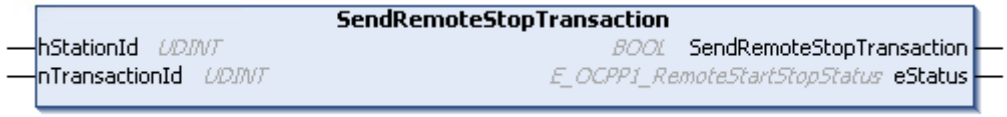
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
sIdToken	T_OCPP1_IdToken [▶ 209]	ID token of the Charge Point in the Central System.
mChargingProfile	REFERENCE TO ST_OCPP1_ChargingProfileMax [▶ 204]	Charging Profile to be sent to the client.

Outputs

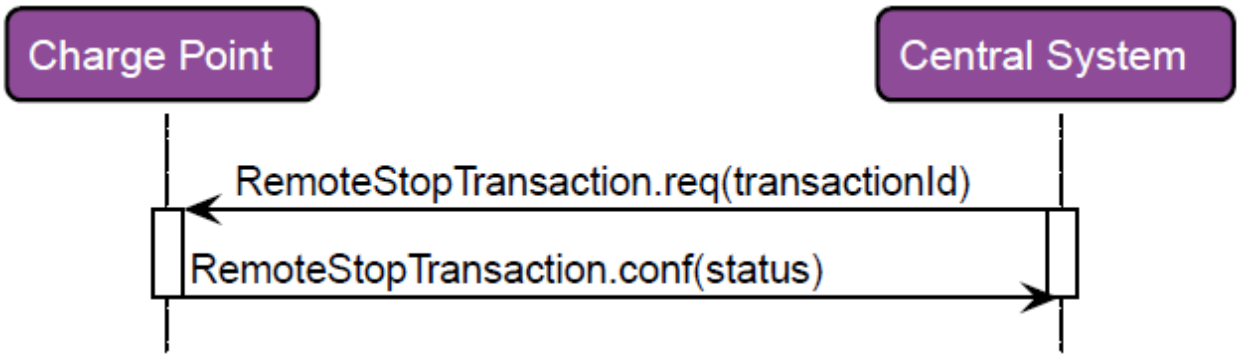
Name	Type	Description
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status shows whether the Charge Point has accepted the request to start a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.46 SendRemoteStopTransaction



With this method, an OCPP server sends a Remote Stop Transaction request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendRemoteStopTransaction : BOOL
VAR_INPUT
    hStationId     : UDINT;
    nTransactionId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus        : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

Name	Type	Description
SendRemoteStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nTransactionId	UDINT	Identifier of the transaction to be stopped.

Outputs

Name	Type	Description
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status indicates whether the Charge Point has accepted the request to stop a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.47 SendReserveNow

With this method, an OCPP server sends a Reserve Now request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.

Syntax

```
METHOD SendReserveNow : BOOL
VAR_INPUT
    hStationId     : UDINT;
    nConnectorId   : UDINT;
    nExpiryDate    : ULINT;
    sIdTag         : T_OCPP1_IdToken := '';
    sParentIdTag   : T_OCPP1_IdToken;
    nReservationId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus        : E_OCPP1_ReservationStatus;
END_VAR
```

Return value

Name	Type	Description
SendReserveNow	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nExpiryDate	ULINT	Date and time of the end of the reservation.
sIdTag	T_OCPP1_IdToken [▶ 209]	Identifier for which the Charge Point should reserve a Connector.
sParentIdTag	T_OCPP1_IdToken [▶ 209]	The parent of the identifier for which the Charge Point is to reserve a Connector.
nReservationId	UDINT	Unique ID of the reservation.

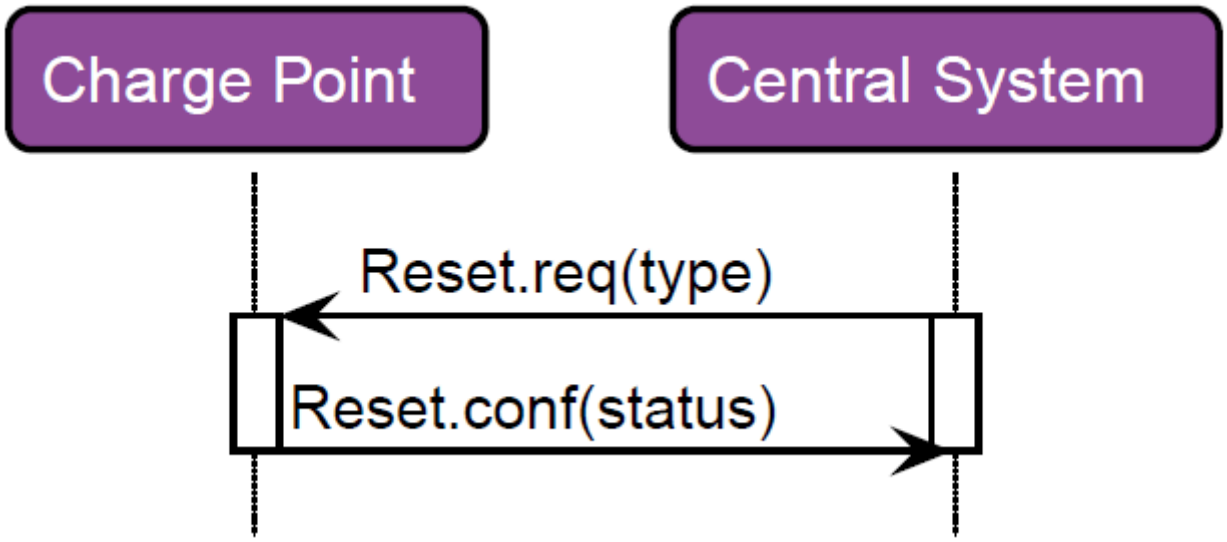
Outputs

Name	Type	Description
eStatus	E_OCPP1_ReservationStatus [▶ 196]	The status indicates whether the reservation was successful.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.48 SendReset

With this method, an OCPP server sends a Reset Request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendReset : BOOL
VAR_INPUT
    hStationId : UDINT;
    eType      : E_OCPP1_ResetType;
END_VAR
VAR_OUTPUT
    eStatus    : E_OCPP1_ResetStatus;
END_VAR
```

Return value

Name	Type	Description
SendReset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

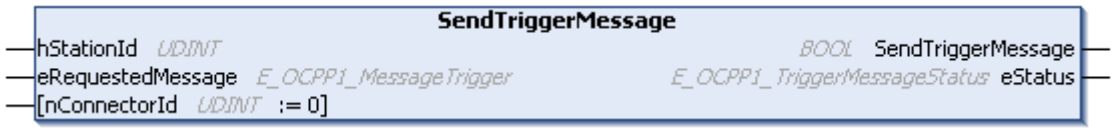
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
eType	E_OCPP1_ResetType [▶ 196]	Type of reset that the Charge Point should perform.

Outputs

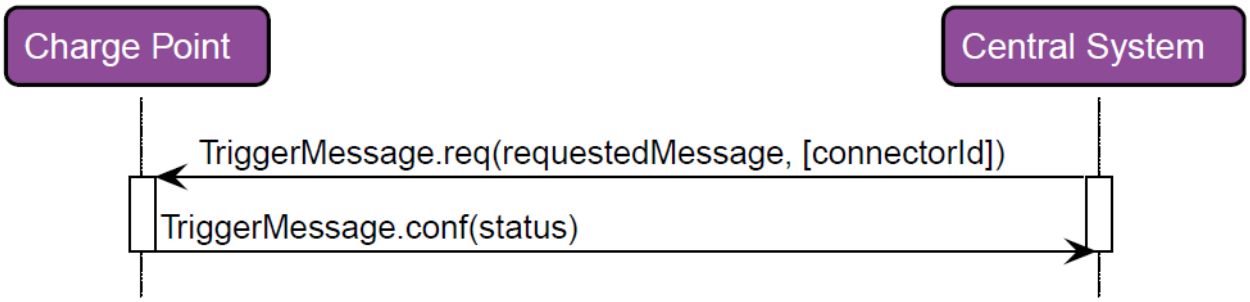
Name	Type	Description
eStatus	E_OCPP1_ResetStatus [▶ 196]	The status shows whether the Charge Point was able to accept the Reset request.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.49 SendTriggerMessage



With this method, an OCPP server sends a Trigger Message request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendTriggerMessage : BOOL
VAR_INPUT
    hStationId        : UDINT;
    eRequestedMessage : E_OCPP1_MessageTrigger;
    nConnectorId      : UDINT := 0;
END_VAR
VAR_OUTPUT
    eStatus           : E_OCPP1_TriggerMessageStatus;
END_VAR
```

Return value

Name	Type	Description
SendTriggerMessage	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
eRequestedMessage	E_OCPP1_MessageTrigger [▶ 194]	Type of message to be triggered.
nConnectorId	UDINT	ID of the Connector of a Charge Point.

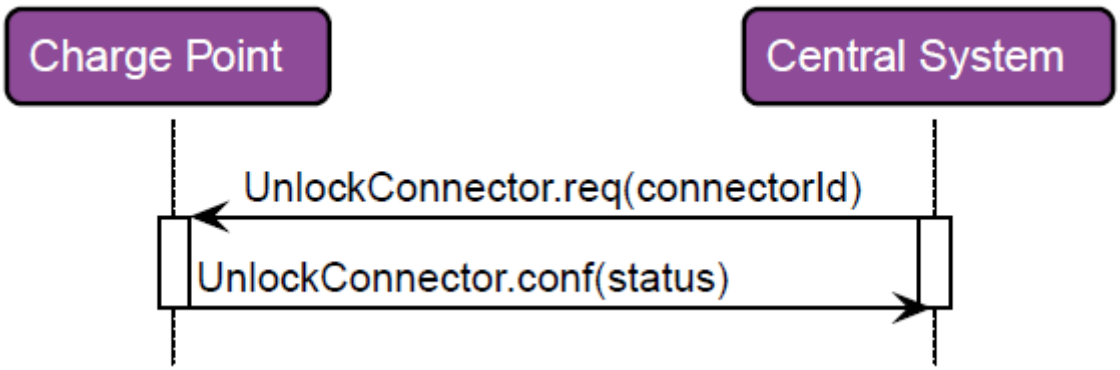
Outputs

Name	Type	Description
eStatus	E_OCPP1_TriggerMessageStatus [▶ 196]	The status indicates whether the Charge Point will send the requested message or not.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.50 SendUnlockConnector

With this method, an OCPP server sends an Unlock Connector request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendUnlockConnector : BOOL
VAR_INPUT
    hStationId   : UDINT;
    nConnectorId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_UnlockStatus;
END_VAR
```

Return value

Name	Type	Description
SendUnlockConnector	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

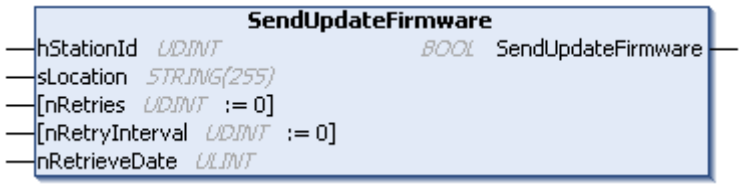
Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
nConnectorId	UDINT	ID of the Connector of a Charge Point.

Outputs

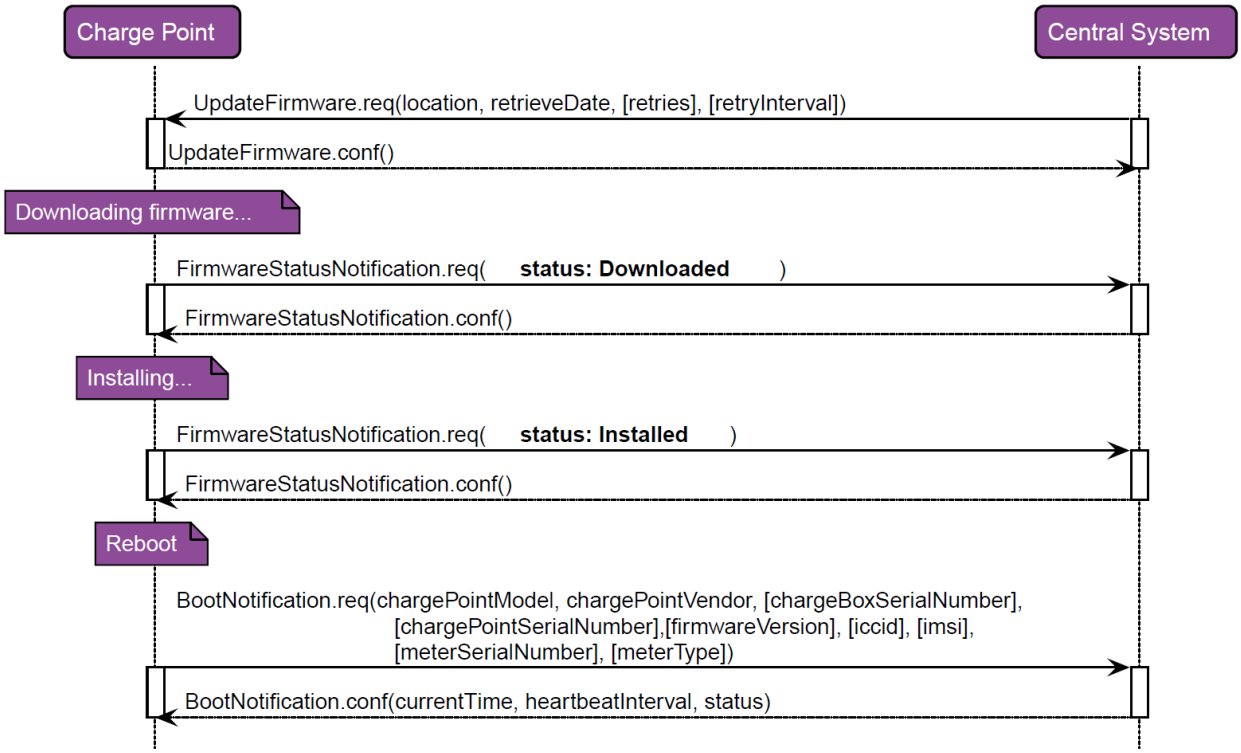
Name	Type	Description
eStatus	E_OCPP1_UnlockStatus [▶ 197]	The status shows whether the Connector has been unlocked.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.2.51 SendUpdateFirmware



With this method, an OCPP server sends an Update Firmware request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendUpdateFirmware : BOOL
VAR_INPUT
    hStationId     : UDINT;
    sLocation      : STRING(255);
    nRetries       : UDINT := 0;
    nRetryInterval : UDINT := 0;
    nRetrieveDate  : ULINT;
END_VAR
```

Return value

Name	Type	Description
SendUpdateFirmware	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hStationId	UDINT	Identifier of the OCPP client in the instance of the OCPP server.
sLocation	STRING(255)	URI from which the firmware is to be retrieved.
nRetries	UDINT	Number of attempts to download the firmware again if the download fails.
nRetryInterval	UDINT	Time after which the download is attempted again.
nRetrieveDate	ULINT	Time and date from which the Charge Point may receive the new firmware.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3 FB_OCPP1_Station

Syntax

```
FUNCTION BLOCK FB_OCPP1_Station
VAR_OUTPUT
    bBusy               : BOOL;
    bError              : BOOL;
    eErrorResult        : HRESULT;
    eErrorAction        : E_OCPP1_Action;
END_VAR
```

Outputs

Name	Type	Description
bValid	BOOL	The interface to the driver in the background exists.
bBusy	BOOL	Is TRUE as long as the function block is busy with processing.
bError	BOOL	Becomes TRUE when an error situation occurs.
eErrorResult	HRESULT	Last error present on the function block.
eErrorAction	E_OCPP1_Action [▶ 189]	OCPP command for which the error occurred.

Properties

Name	Type	Access	Description
IsConnected	BOOL	Get	Status of the connection to the OCPP server.
IsPending	BOOL	Get	Waiting for completion of the request.
StationId	UDINT	Get	Identifier of the OCPP client in the instance of the OCPP server.

5.1.3.1 FB_Exit

Syntax

```
METHOD FB_Exit : BOOL
VAR_INPUT
    bInCopyCode : BOOL;
END_VAR
```

Return value

Name	Type	Description
FB_Exit	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bInCopyCode	BOOL	If TRUE, the Exit method is called to exit an instance, which is then copied (online change).

5.1.3.2 FB_Init

Syntax

```
METHOD FB_Init : BOOL
VAR_INPUT
    bInitRetains : BOOL;
    bInCopyCode  : BOOL;
    sIdent       : T_OCPP_Identity;
```

```
    sAuthHash    : T_OCPP_Hash;
    fbServer     : REFERENCE TO FB_OCPP1_Server;
END_VAR
```

Return value

Name	Type	Description
FB_Init	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
bInitRetains	BOOL	If TRUE, the Retain variables are initialized (warm start/cold start).
bInCopyCode	BOOL	If TRUE, the Exit method is called to exit an instance, which is then copied (online change).
sIdent	T_OCPP_Identity [▶ 209]	Identity of the OCPP client.
sAuthHash	T_OCPP_Hash [▶ 209]	Hash generated from the identity and the password. More information on calculating the hash can be found in the appendix [▶ 211].
fbServer	REFERENCE TO FB_OCPP1_Server [▶ 76]	OCPP server instance to be used for this station function block.

5.1.3.3 Init

This method is called when the function block is initialized in the FB_Init [▶ 132] method and specifies the parameters of the WebSockets server. If the connection parameters are to be changed afterwards, this method can be called again while the application is running.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    sIdent    : T_OCPP_Identity;
    sAuthHash : T_OCPP_Hash;
    fbServer  : REFERENCE TO FB_OCPP1_Server;
END_VAR
```

Return value

Name	Type	Description
Init	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sIdent	T_OCPP_Identity [▶ 209]	Identity of the OCPP client to be connected.
sAuthHash	T_OCPP_Hash [▶ 209]	Hash value for the respective OCPP client. The calculation is explained at Hash calculation [▶ 211].
fbServer	REFERENCE TO FB_OCPP1_Server [▶ 76]	Instance of the OCPP server for which the station should be created.

5.1.3.4 PollRequest

This method must be called to receive incoming requests from the connected OCPP client.

Syntax

```
METHOD PollRequest : BOOL
VAR_OUTPUT
    eAction    : E_OCPP1_Action;
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

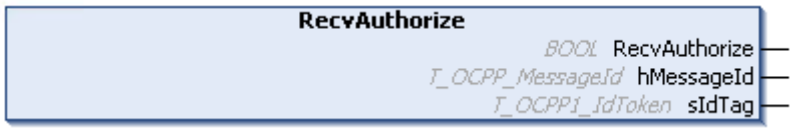
Name	Type	Description
PollRequest	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

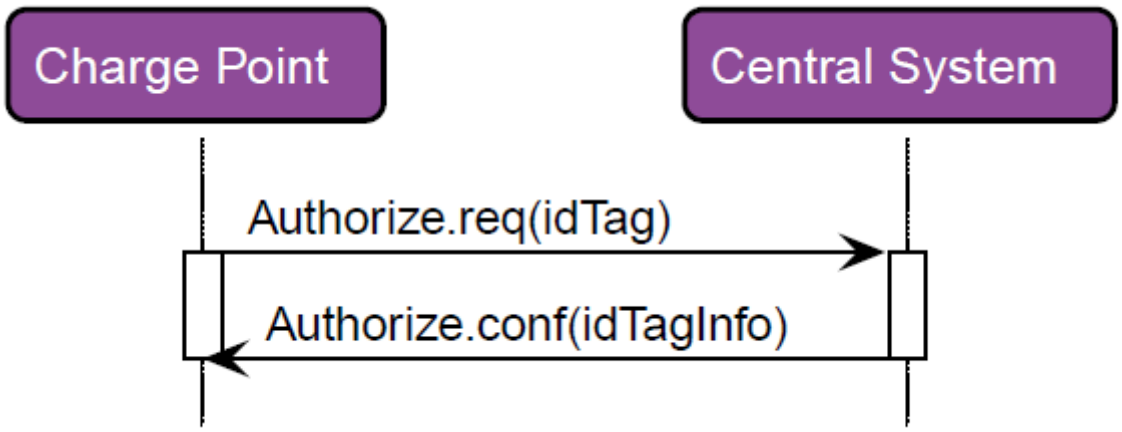
Name	Type	Description
eAction	E_OCPP1_Action [▶ 189]	Type of OCPP request received from the server.
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.5 RecvAuthorize



With this method, an OCPP server receives an Authorize request from the connected OCPP client. To respond to the request, the RespAuthorize [▶ 147] method must be called.



Syntax

```
METHOD RecvAuthorize : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    sIdTag     : T_OCPP1_IdToken;
END_VAR
```

Return value

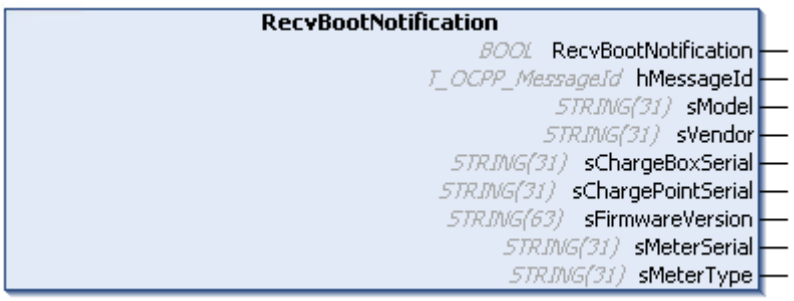
Name	Type	Description
RecvAuthorize	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

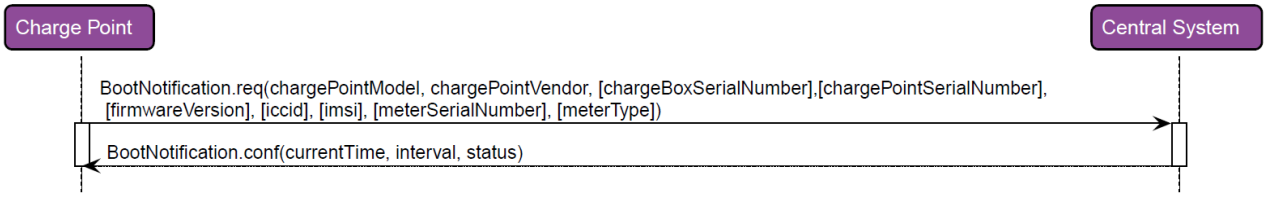
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token with which the Charge Point wants to be authorized on the Central System.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.6 RecvBootNotification



With this method, an OCPP server receives a Boot Notification from the connected OCPP client. To respond to the boot notification, the RespBootNotification [▶ 148] method must be called.



Syntax

```
METHOD RecvBootNotification : BOOL
VAR_OUTPUT
    hMessageId          : T_OCPP_MessageId;
    sModel              : STRING(31);
    sVendor             : STRING(31);
    sChargeBoxSerial    : STRING(31);
    sChargePointSerial  : STRING(31);
    sFirmwareVersion    : STRING(63);
    sMeterSerial        : STRING(31);
    sMeterType          : STRING(31);
END_VAR
```

Return value

Name	Type	Description
RecvBootNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

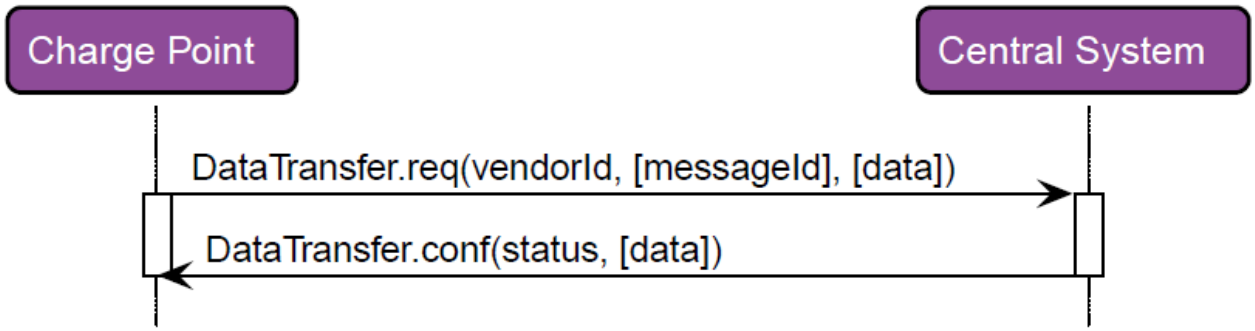
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sModel	STRING(31)	Model of the Charge Point.
sVendor	STRING(31)	Vendor of the Charge Point.
sChargeBoxSerial	STRING(31)	Serial number of the Charge Box within the Charge Point.
sChargePointSerial	STRING(31)	Serial number of the Charge Point.
sFirmwareVersion	STRING(63)	Firmware version of the Charge Point.
sMeterSerial	STRING(31)	Serial number of the main electricity meter of the Charge Point.
sMeterType	STRING(31)	Type of main electricity meter of the Charge Point

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.7 RecvDataTransfer

With this method, an OCPP server receives a Data Transfer request from the connected OCPP client. To respond to the request, the RespDataTransfer [▶ 149] method must be called.



Syntax

```
METHOD RecvDataTransfer : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    sVendorId  : STRING(255);
    sMessageId : STRING(63);
END_VAR
VAR_IN_OUT
    sData      : T_OCPP1_DataTransferData;
END_VAR
```

Return value

Name	Type	Description
RecvDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sVendorId	STRING(255)	Identifier for the vendor-specific implementation.
sMessageId	STRING(63)	Identifier for the individual message.

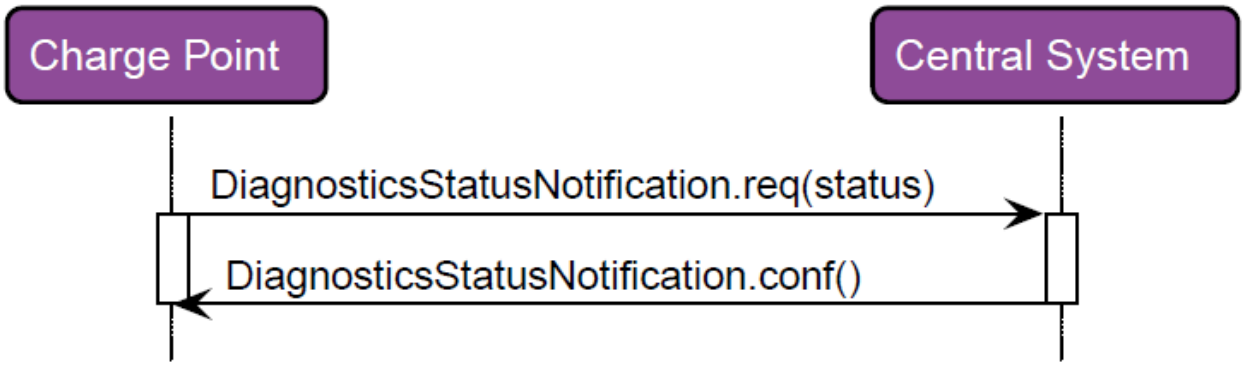
Inputs/outputs

Name	Type	Description
sData	T_OCPP1_DataTransferData [▶ 208]	Text without specified length and format.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.8 RecvDiagnosticsStatusNotification

With this method, an OCPP server receives a Diagnostics Status Notification from the connected OCPP client. To respond to the Diagnostics Status Notification, the RespDiagnosticsStatusNotification [▶ 150] method must be called.



Syntax

```
METHOD RecvDiagnosticsStatusNotification : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_DiagnosticsStatus;
END_VAR
```

Return value

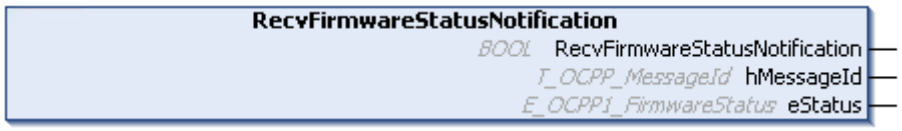
Name	Type	Description
RecvDiagnosticsStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

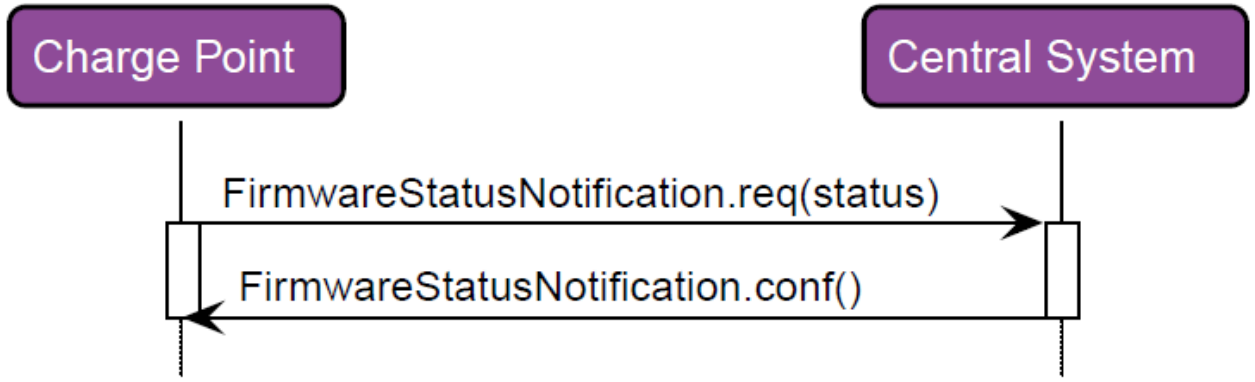
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_DiagnosticsStatus [▶ 193]	Status of the upload of the diagnostic data.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.9 RecvFirmwareStatusNotification



With this method, an OCPP server receives a Firmware Status Notification from the connected OCPP client. To respond to the Firmware Status Notification, the RespFirmwareStatusNotification [▶ 151] method must be called.



Syntax

```
METHOD RecvFirmwareStatusNotification : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_FirmwareStatus;
END_VAR
```

Return value

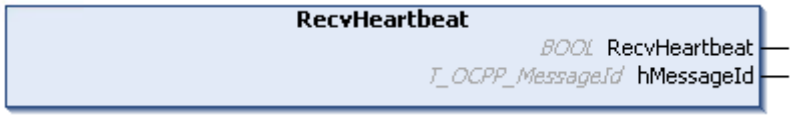
Name	Type	Description
RecvFirmwareStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

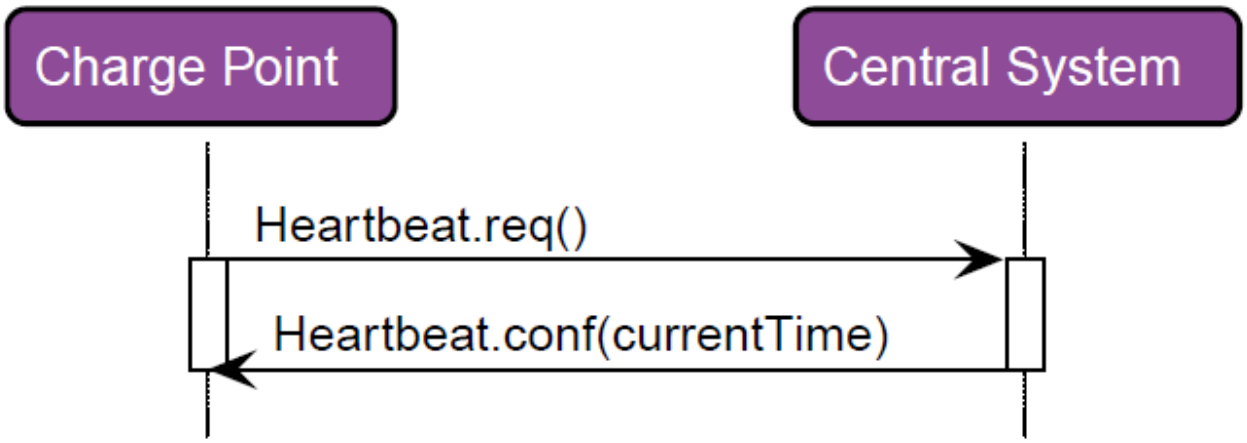
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_FirmwareStatus [▶ 193]	Status of a firmware installation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.10 RecvHeartbeat



With this method, an OCPP server receives a Heartbeat from the connected OCPP client. To respond to the Heartbeat, the RespHeartbeat [▶ 152] method must be called.



Syntax

```
METHOD RecvHeartbeat : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RecvHeartbeat	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

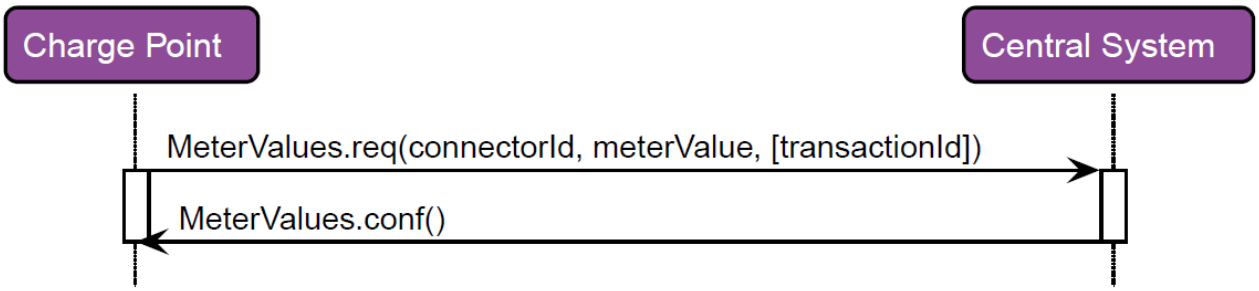
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.11 RecvMeterValue

With this method, an OCPP server receives Meter Values from the connected OCPP client. To respond to the receipt of the Meter Values, the RespMeterValue [▶ 153] method must be called.

Contrary to the specification, it is possible to receive Meter Value messages without Meter Values in order to increase compatibility with other vendors.



Syntax

```
METHOD RecvMeterValue : BOOL
VAR_IN_OUT
    arrSampleValues : ARRAY[*] OF ST_OCPP1_SampledValue;
END_VAR
VAR_OUTPUT
    hMessageId      : T_OCPP_MessageId;
    nConnectorId    : UDINT;
    nTransactionId  : UDINT;
    nSampleCount    : UDINT;
```

Return value

Name	Type	Description
RecvMeterValue	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs/outputs

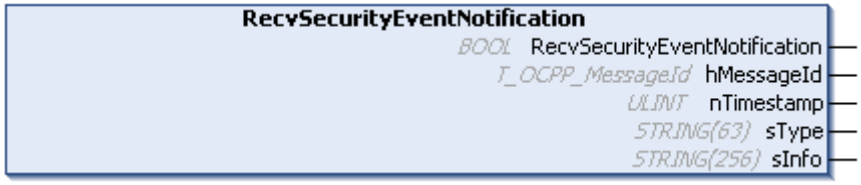
Name	Type	Description
arrSampleValues	ARRAY [*] OF ST_OCPP1_SampledValue [▶ 207]	The sampled values sent by the client.

Outputs

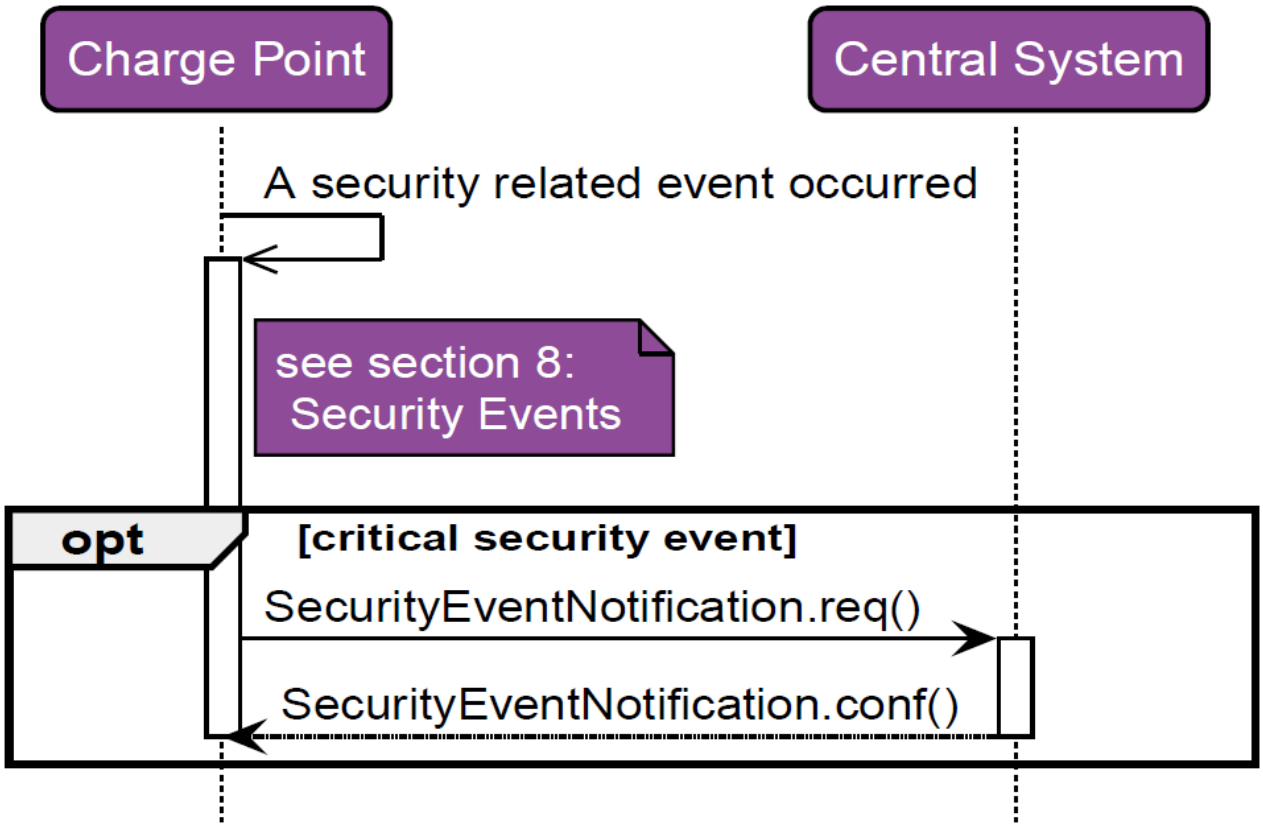
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nTransactionId	UDINT	The transaction to which the sampled values belong.
nSampleCount	UDINT	The number of sampled values contained.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.12 RecvSecurityEventNotification



With this method, an OCPP server receives a Security Event Notification from the connected OCPP client. To respond to the Security Event Notification, the RespSecurityEventNotification [▶ 154] method must be called.



Syntax

```
METHOD RecvSecurityEventNotification : BOOL
VAR_OUTPUT
    hMessageId : T_OCPP_MessageId;
    nTimestamp : ULINT;
    sType      : STRING(63);
    sInfo      : STRING(256);
END_VAR
```

Return value

Name	Type	Description
RecvSecurityEventNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

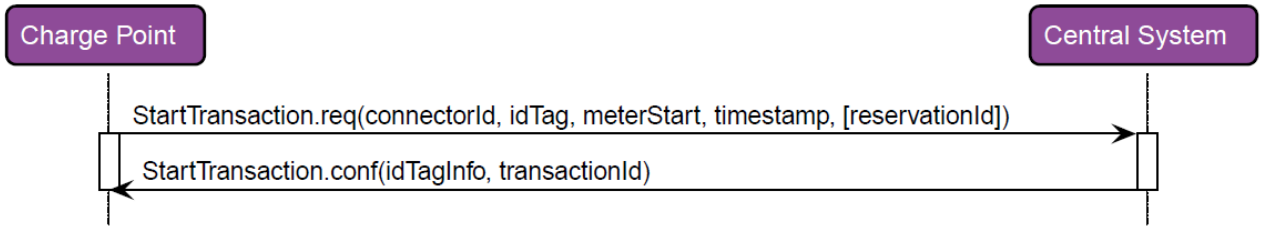
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nTimestamp	ULINT	Date and time when the event occurs.
sType	STRING(63)	Type of the Security Event according to the OCPP specification.
sInfo	STRING(256)	Additional information about the Security Event that has occurred.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.13 RecvStartTransaction

With this method, an OCPP server receives a Start Transaction request from the connected OCPP client. To respond to the request, the RespStartTransaction [▶ 155] method must be called.



Syntax

```
METHOD RecvStartTransaction : BOOL
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    sIdTag         : T_OCPP1_IdToken;
    nConnectorId   : UDINT;
    nMeterStart    : UDINT;
    nReservationId : UDINT;
    nTimestamp     : ULINT;
END_VAR
```

Return value

Name	Type	Description
RecvStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

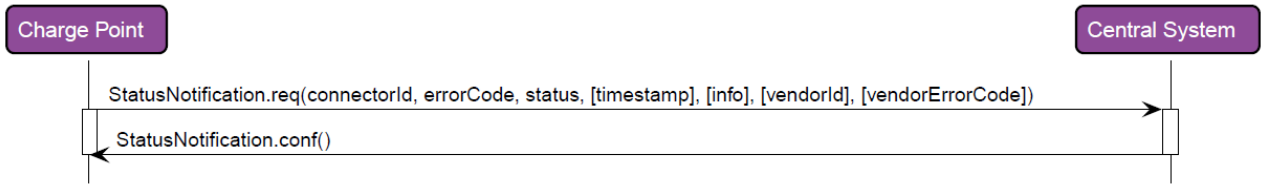
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token with which the transaction is to be started.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nMeterStart	UDINT	Value in watt-hours at the start of the transaction.
nReservationId	UDINT	Optional reservation ID.
nTimestamp	ULINT	Date and time at the start of the transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.14 RecvStatusNotification

With this method, an OCPP server receives a Status Notification from the connected OCPP client. To respond to the Status Notification, the RespStatusNotification [▶ 156] method must be called.



Syntax

```
METHOD RecvStatusNotification : BOOL
VAR_OUTPUT
    hMessageId    : T_OCPP_MessageId;
    nConnectorId  : UDINT;
    eError        : E_OCPP1_ChargePointError;
    eStatus       : E_OCPP1_ChargePointStatus;
    nTimestamp    : ULINT := 0;
    sInfo         : STRING(63) := '';
    sVendorError  : STRING(63) := '';
    sVendorId     : STRING(255) := '';
END_VAR
```

Return value

Name	Type	Description
RecvStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

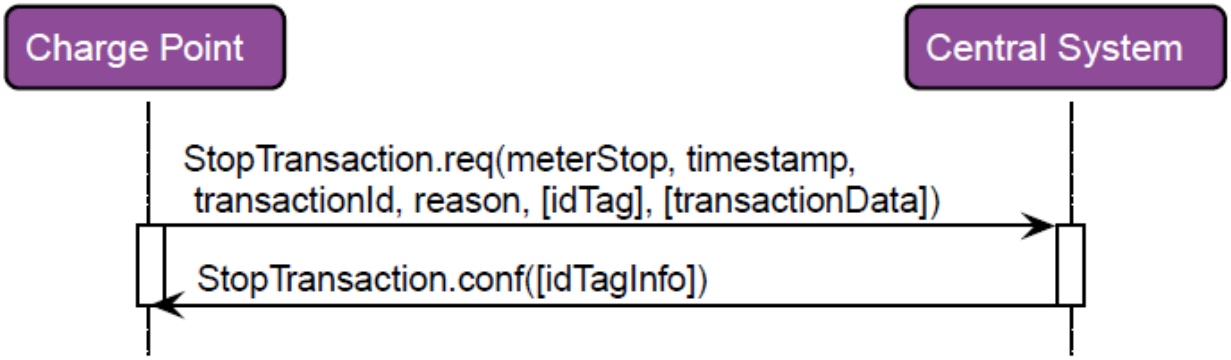
Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
eError	E_OCPP1_ChargePointError [▶ 190]	Error code received in the Status Notification.
eStatus	E_OCPP1_ChargePointStatus [▶ 191]	Status received in the Status Notification.
nTimestamp	ULINT	The timestamp of the Status Notification.
sInfo	STRING(63)	Contains optional, freely definable additional information on the error.
sVendorError	STRING(63)	Optionally contains the vendor-specific error code.
sVendorId	STRING(255)	Optionally contains the identifier for the vendor-specific implementation.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.15 RecvStopTransaction

With this method, an OCPP server receives a Stop Transaction request from the connected OCPP client. To respond to the request, the RespStopTransaction [▶ 157] method must be called.



Syntax

```
METHOD RecvStopTransaction : BOOL
VAR_OUTPUT
    hMessageId     : T_OCPP_MessageId;
    sIdTag         : T_OCPP1_IdToken;
    nTransactionId : UDINT;
    nConnectorId   : UDINT;
    nMeterStop     : UDINT;
    nTimestamp     : ULINT;
    eReason        : E_OCPP1_Reason;
END_VAR
```

Return value

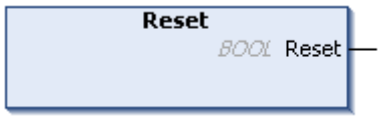
Name	Type	Description
RecvStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
sIdTag	T_OCPP1_IdToken [▶ 209]	ID token for which the transaction is to be stopped.
nTransactionId	UDINT	ID of the transaction to be stopped. Is contained in the Central System's response to a Start Transaction request.
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nMeterStop	UDINT	Value in watt-hours at the end of the transaction.
nTimestamp	ULINT	Date and time when the transaction is stopped.
eReason	E_OCPP1_Reason [▶ 195]	Can optionally contain the reason for stopping the transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.16 Reset



This method is used to reset the last error present on the function block.

Syntax

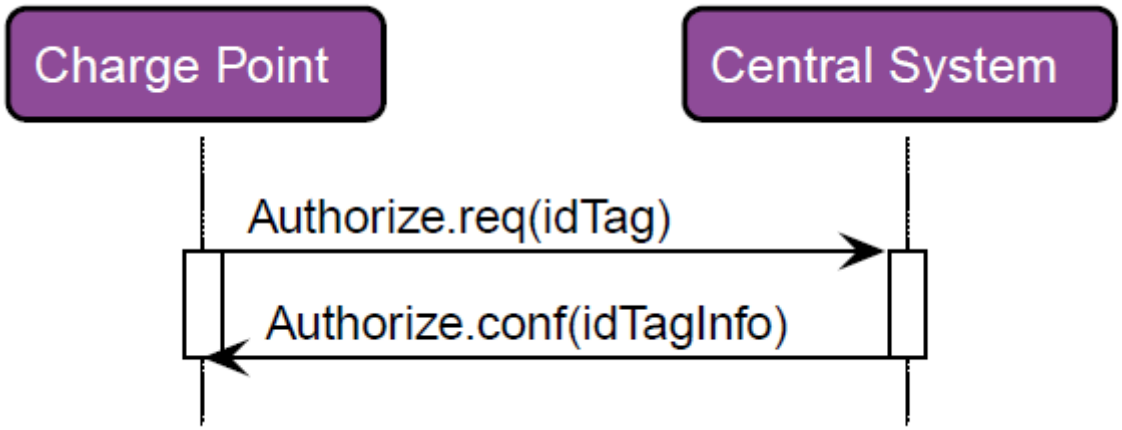
```
METHOD Reset : BOOL
```

Return value

Name	Type	Description
Reset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

5.1.3.17 RespAuthorize

With this method, an OCPP server responds to an Authorize request from the connected OCPP client.



Syntax

```
METHOD RespAuthorize : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

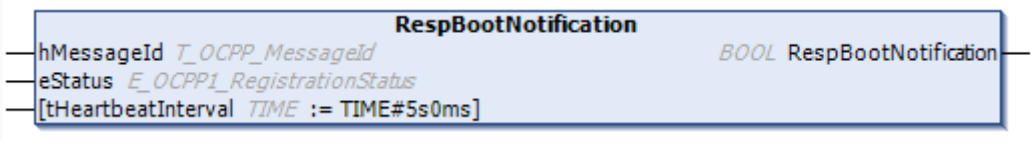
Name	Type	Description
RespAuthorize	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

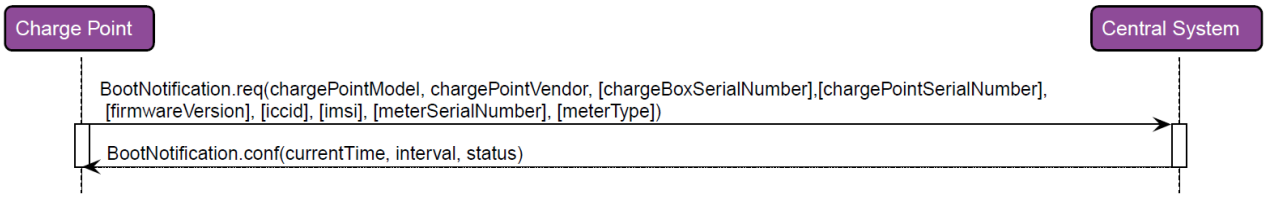
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response to the OCPP client.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.18 RespBootNotification



With this method, an OCPP server responds to a Boot Notification from the connected OCPP client.



Syntax

```
METHOD RespBootNotification : BOOL
VAR_INPUT
    hMessageId          : T_OCPP_MessageId;
    eStatus             : E_OCPP1_RegistrationStatus;
    tHeartbeatInterval : TIME := T#5S;
END_VAR
```

Return value

Name	Type	Description
RespBootNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

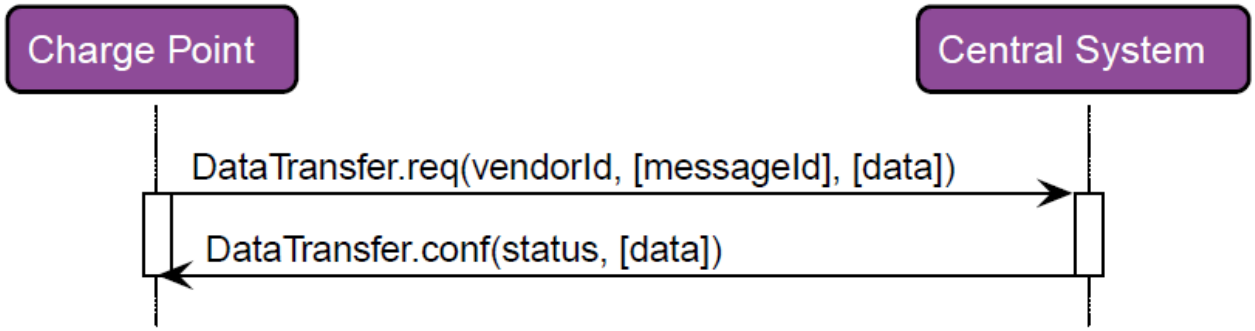
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_RegistrationStatus [▶ 196]	Status of the registration in response to the OCPP client.
tHeartbeatInterval	TIME	If the registration status is Accepted, this variable contains the Heartbeat Interval for the connection.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.19 RespDataTransfer

With this method, an OCPP server responds to a Data Transfer request from the connected OCPP client.



Syntax

```
METHOD RespDataTransfer : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_DataTransferStatus;
END_VAR
```

Return value

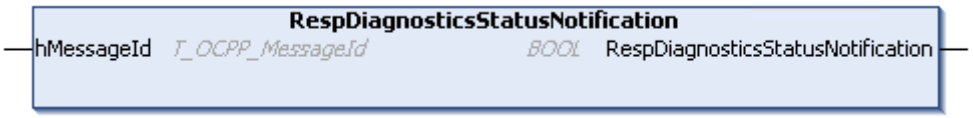
Name	Type	Description
RespDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

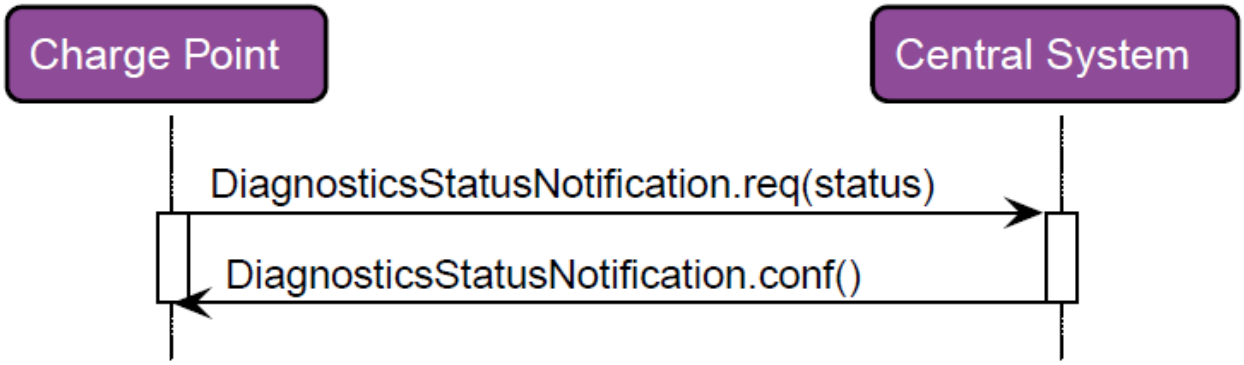
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_DataTransferStatus [▶ 193]	Status of the Data Transfer in response to the OCPP client.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.20 RespDiagnosticsStatusNotification



With this method, an OCPP server responds to a Status Notification from the connected OCPP client.



Syntax

```
METHOD RespDiagnosticsStatusNotification : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RespDiagnosticsStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.21 RespError

With this method, an OCPP server responds to the connected OCPP client if the sent request has triggered an internal error. This may be the case, for example, if a requested action is not available.

Syntax

```
METHOD RespError : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eAct       : E_OCPP1_Action := E_OCPP1_Action.Heartbeat;
    eErr       : E_OCPP1_Error := E_OCPP1_Error.None;
END_VAR
```

Return value

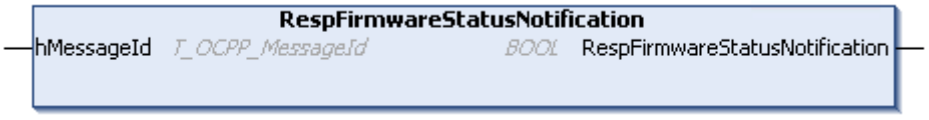
Name	Type	Description
RespError	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

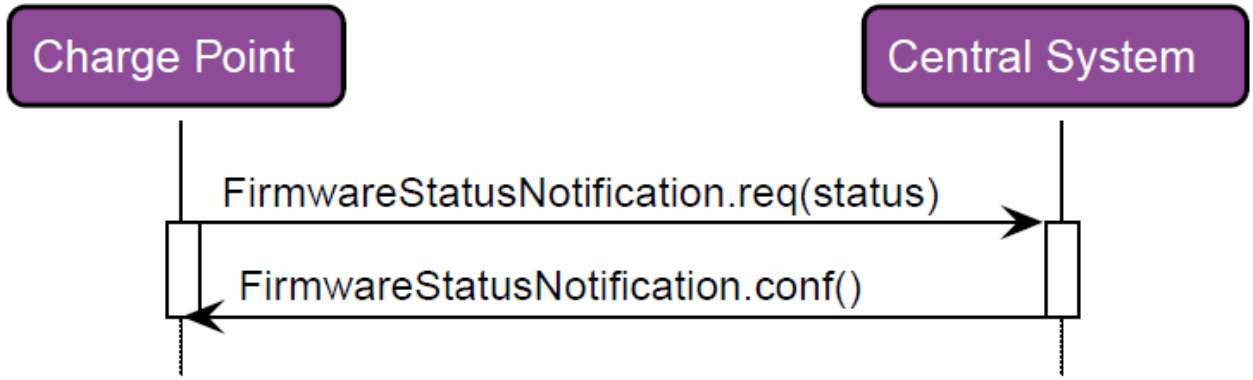
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eAct	E_OCPP1_Action [▶ 189]	Type of OCPP request for which the error occurred.
eErr	E_OCPP1_Error [▶ 193]	OCPP error that will be the response to the request.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.22 RespFirmwareStatusNotification



With this method, an OCPP server responds to a Firmware Status Notification from the connected OCPP client.



Syntax

```
METHOD RespFirmwareStatusNotification : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

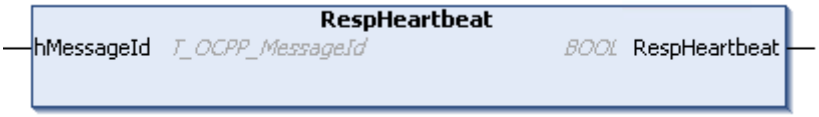
Name	Type	Description
RespFirmwareStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

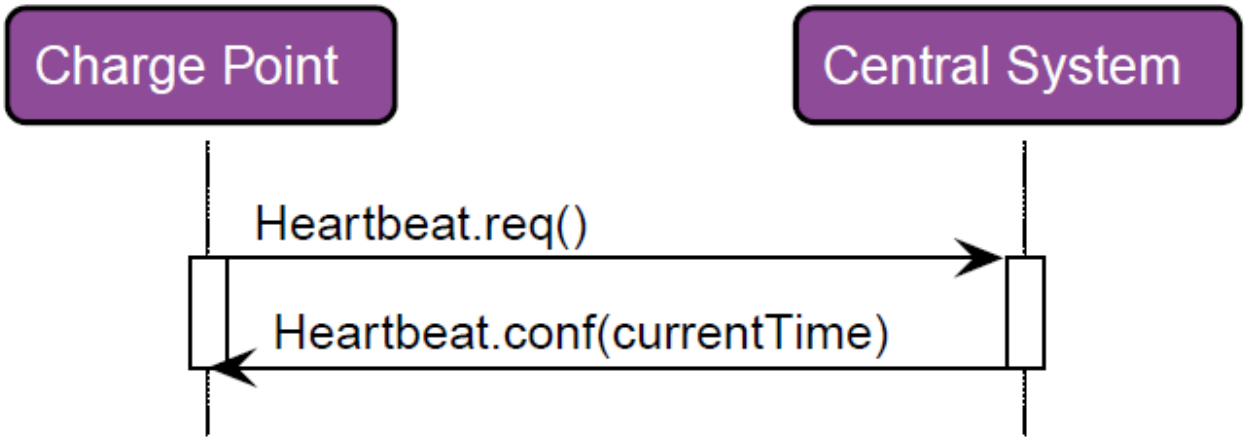
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.23 RespHeartbeat



With this method, an OCPP server responds to a Heartbeat from the connected OCPP client.



Syntax

```
METHOD RespHeartbeat : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RespHeartbeat	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

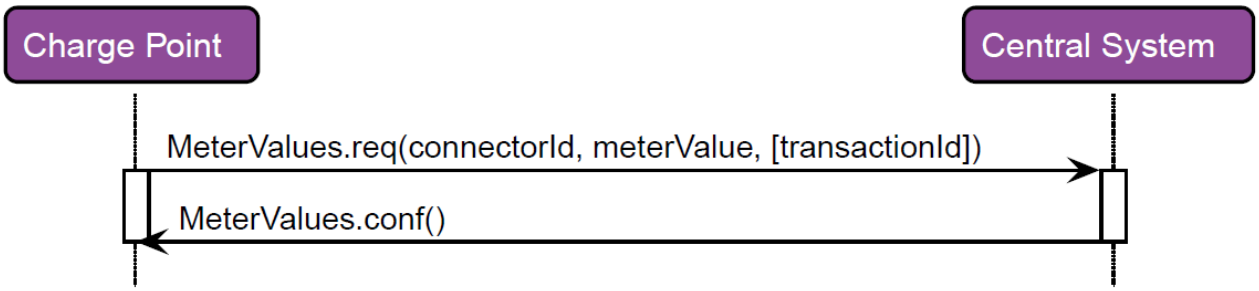
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.24 RespMeterValue

With this method, an OCPP server responds to the sending of Meter Values from the connected OCPP client.



Syntax

```
METHOD RespMeterValue : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

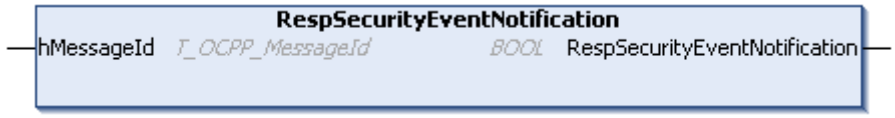
Name	Type	Description
RespMeterValue	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

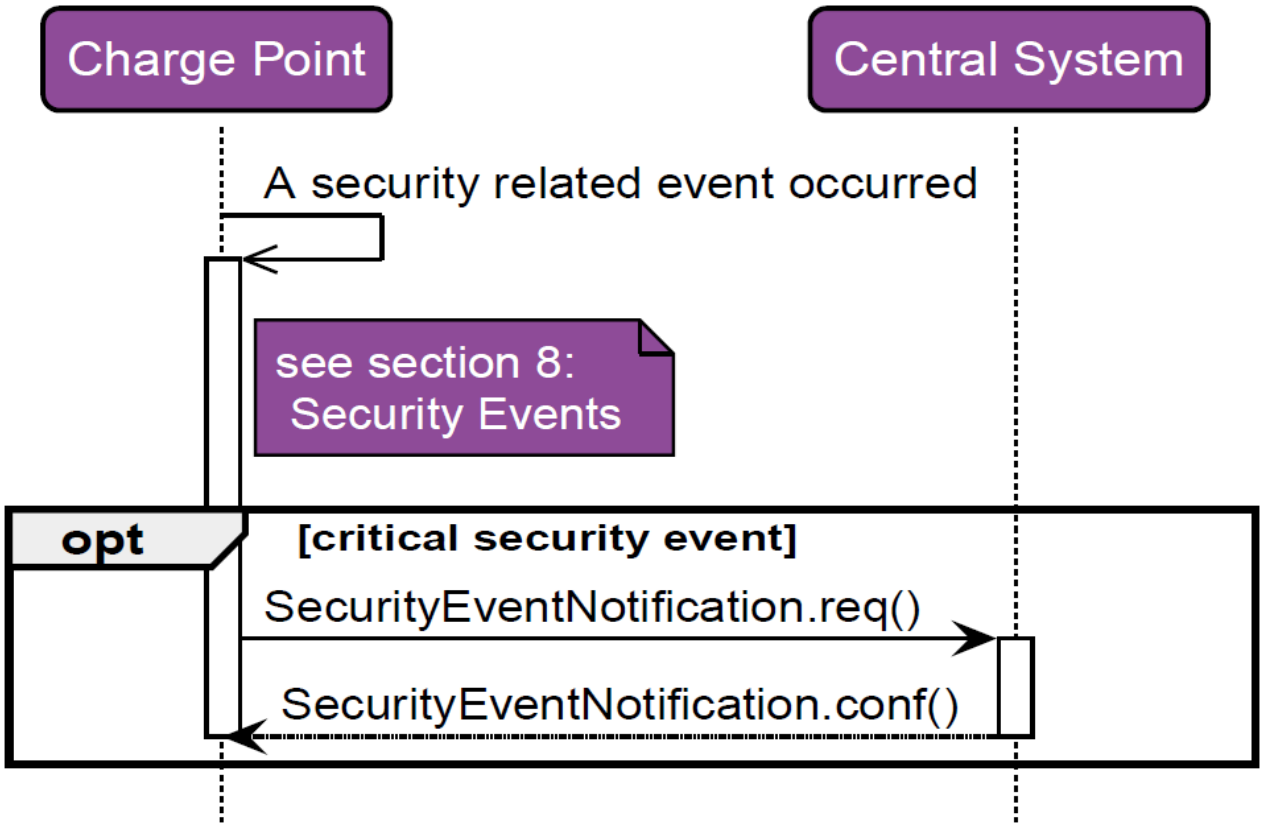
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.25 RespSecurityEventNotification



With this method, an OCPP server responds to a Security Event Notification from the connected OCPP client.



Syntax

```
METHOD RespSecurityEventNotification : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

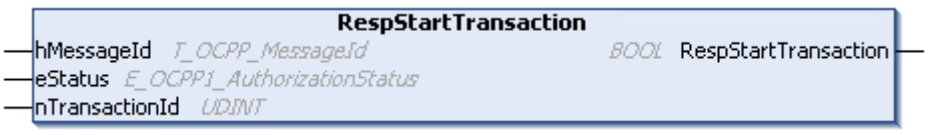
Name	Type	Description
RespSecurityEventNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

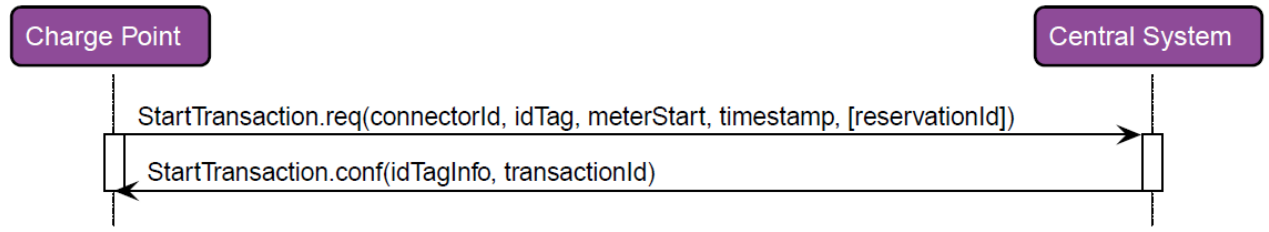
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.26 RespStartTransaction



With this method, an OCPP server responds to a Start Transaction request from the connected OCPP client.



Syntax

```
METHOD RespStartTransaction : BOOL
VAR_INPUT
    hMessageId     : T_OCPP_MessageId;
    eStatus        : E_OCPP1_AuthorizationStatus;
    nTransactionId : UDINT;
END_VAR
```

Return value

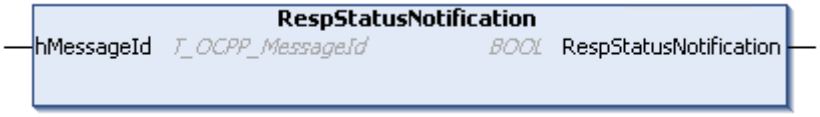
Name	Type	Description
RespStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

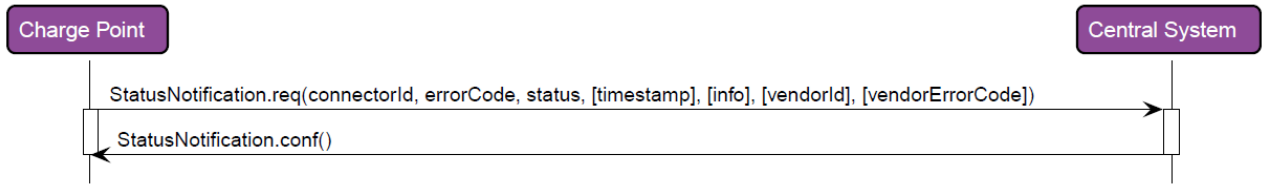
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response to the OCPP client.
nTransactionId	UDINT	ID defined by the Central System for the transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.27 RespStatusNotification



With this method, an OCPP server responds to a Status Notification from the connected OCPP client.



Syntax

```
METHOD RespStatusNotification : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
END_VAR
```

Return value

Name	Type	Description
RespStatusNotification	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

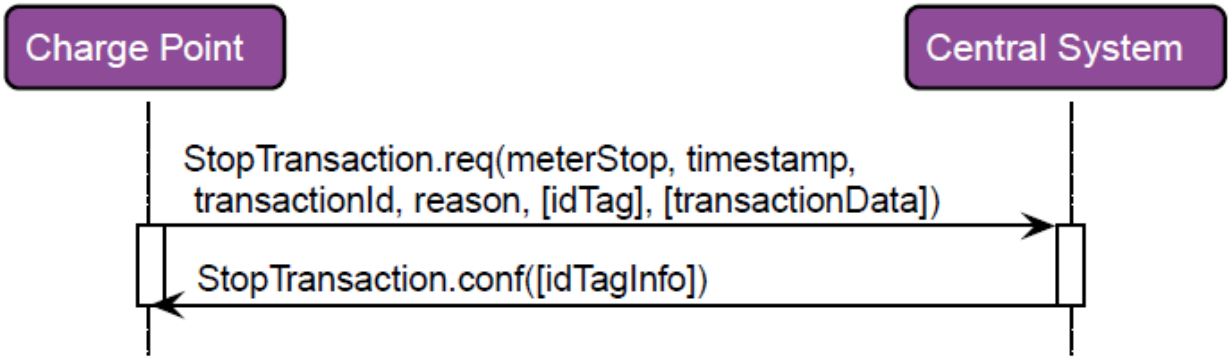
Inputs

Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.28 RespStopTransaction

With this method, an OCPP server responds to a Stop Transaction request from the connected OCPP client.



Syntax

```
METHOD RespStopTransaction : BOOL
VAR_INPUT
    hMessageId : T_OCPP_MessageId;
    eStatus    : E_OCPP1_AuthorizationStatus;
END_VAR
```

Return value

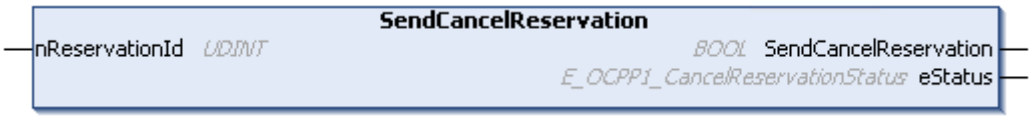
Name	Type	Description
RespStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

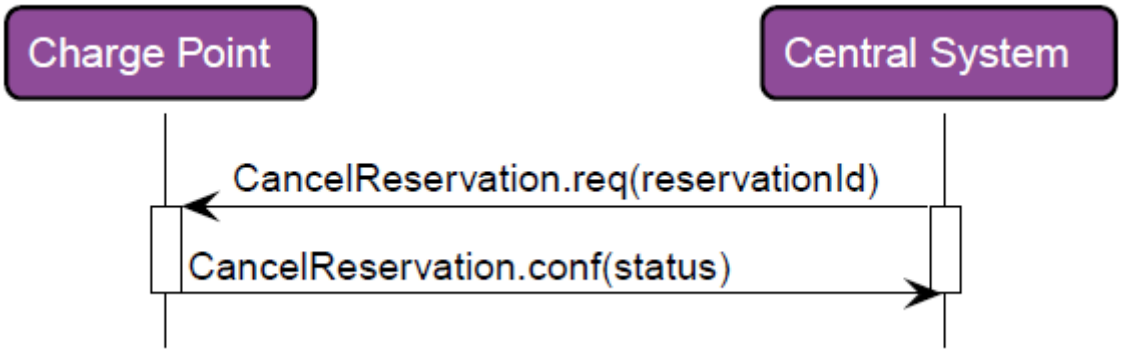
Name	Type	Description
hMessageId	T_OCPP_MessageId [▶ 209]	MessageId of the received message.
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	Status of the authorization in response to the OCPP client.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.29 SendCancelReservation



With this method, an OCPP server sends a Cancel Reservation request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendCancelReservation : BOOL
VAR_INPUT
    nReservationId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus        : E_OCPP1_CancelReservationStatus;
END_VAR
```

Return value

Name	Type	Description
SendCancelReservation	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

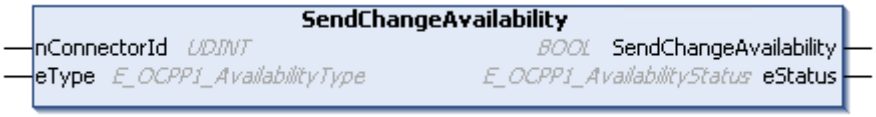
Name	Type	Description
nReservationId	UDINT	ID of the reservation to be canceled.

Outputs

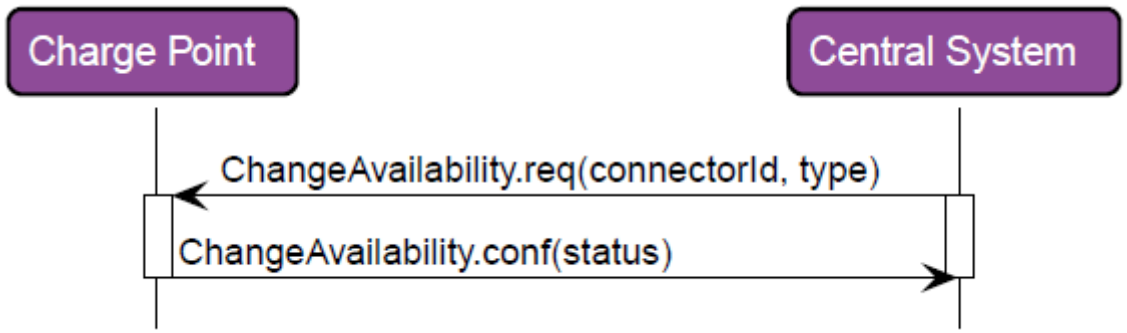
Name	Type	Description
eStatus	E_OCPP1_CancelReservationStatus [▶ 190]	The status shows whether the reservation was successfully canceled.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.30 SendChangeAvailability



With this method, an OCPP server sends a Change Availability request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendChangeAvailability : BOOL
VAR_INPUT
    nConnectorId : UDINT;
    eType        : E_OCPP1_AvailabilityType;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_AvailabilityStatus;
END_VAR
```

Return value

Name	Type	Description
SendChangeAvailability	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

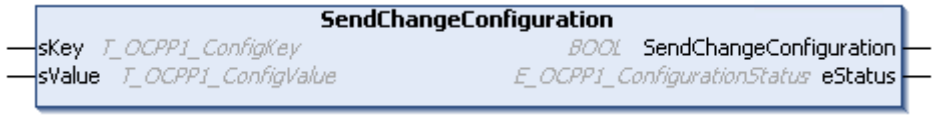
Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
eType	E_OCPP1_AvailabilityType [▶ 190]	Type of availability change that the Charge Point is to carry out.

Outputs

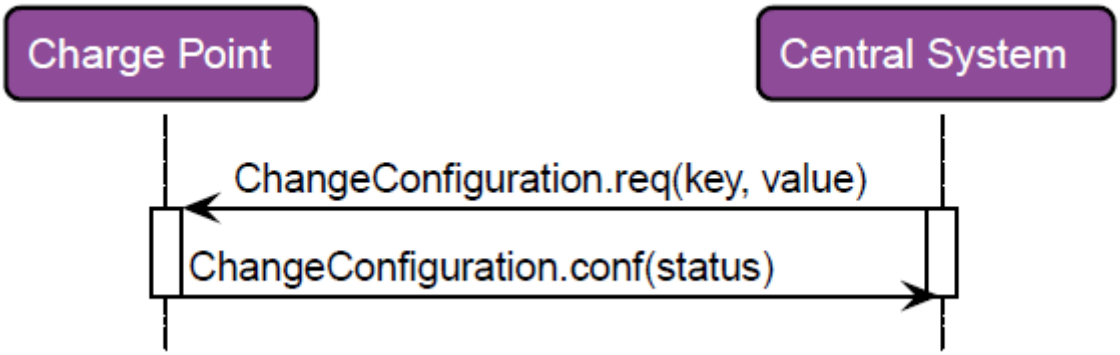
Name	Type	Description
eStatus	E_OCPP1_AvailabilityStatus [▶ 190]	Response whether the Charge Point was able to implement the requested availability change.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.31 SendChangeConfiguration



With this method, an OCPP server sends a Change Configuration request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendChangeConfiguration : BOOL
VAR_INPUT
    sKey        : T_OCPP1_ConfigKey;
    sValue      : T_OCPP1_ConfigValue;
END_VAR
VAR_OUTPUT
    eStatus     : E_OCPP1_ConfigurationStatus;
END_VAR
```

Return value

Name	Type	Description
SendChangeConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sKey	T_OCPP1_ConfigKey [▶ 208]	The name of the configuration setting to be changed.
sValue	T_OCPP1_ConfigValue [▶ 208]	The new value of the configuration setting.

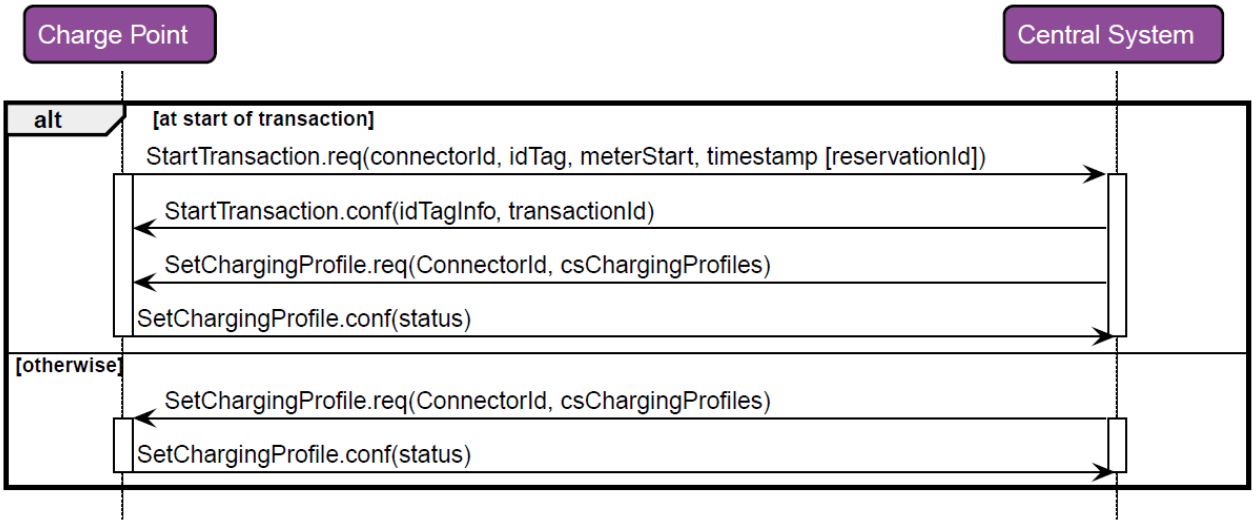
Outputs

Name	Type	Description
eStatus	E_OCPP1_ConfigurationStatus [▶ 193]	The status indicates whether the configuration change has been accepted.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.32 SendChargingProfile

With this method, an OCPP server sends a Charging Profile to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendChargingProfile : BOOL
VAR_INPUT
    nConnectorId     : UDINT;
```

```
    mChargingProfile : REFERENCE TO ST_OCPP1_ChargingProfileMax;
END_VAR
VAR_OUTPUT
    eStatus          : E_OCPP1_ChargingProfileStatus;
END_VAR
```

Return value

Name	Type	Description
SendChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

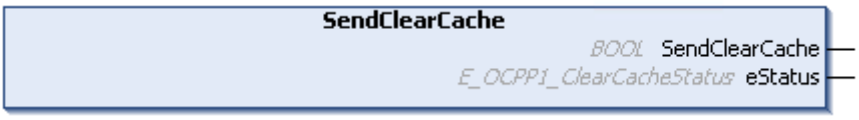
Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
mChargingProfile	REFERENCE TO ST_OCPP1_ChargingProfileMax [▶ 204]	Charging Profile to be sent to the client.

Outputs

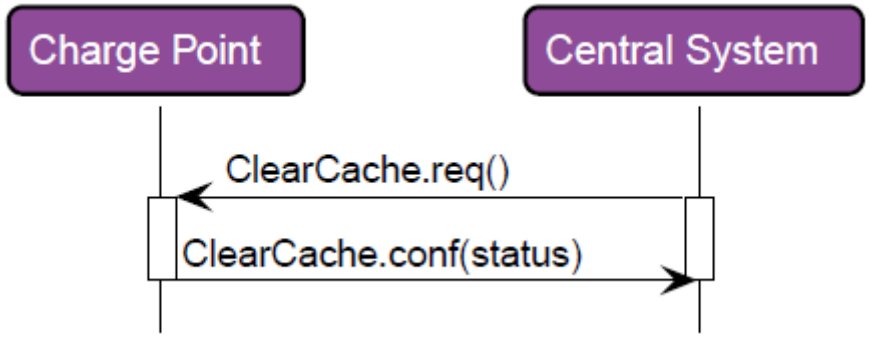
Name	Type	Description
eStatus	E_OCPP1_ChargingProfileStatus [▶ 191]	The status indicates whether the Charging Profile has been accepted by the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.33 SendClearCache



With this method, an OCPP server sends a Clear Cache request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendClearCache : BOOL
VAR_OUTPUT
    eStatus    : E_OCPP1_ClearCacheStatus;
END_VAR
```

Return value

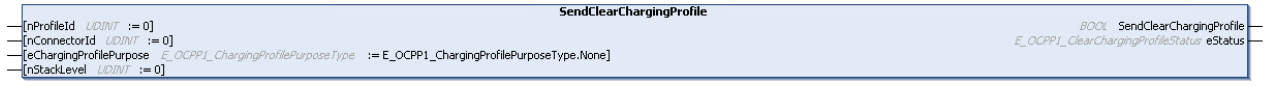
Name	Type	Description
SendClearCache	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Outputs

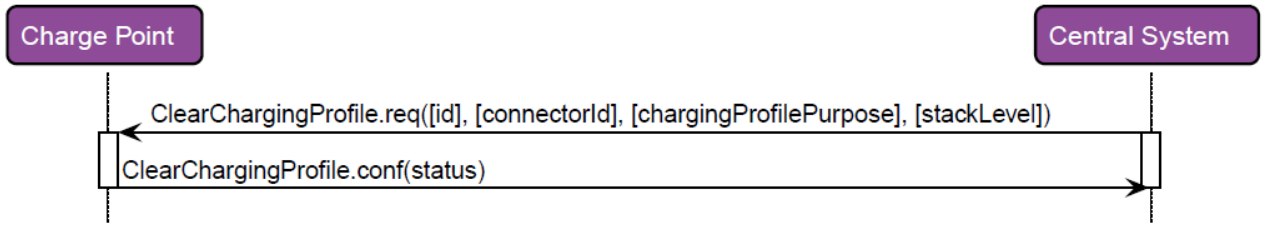
Name	Type	Description
eStatus	E_OCPP1_ClearCacheStatus [▶ 192]	The status indicates whether the clearing of the Cache Profile has been accepted by the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.34 SendClearChargingProfile



With this method, an OCPP server sends a Clear Charging Profile request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendClearChargingProfile : BOOL
VAR_INPUT
    nProfileId              : UDINT := 0;
    nConnectorId            : UDINT := 0;
    eChargingProfilePurpose : E_OCPP1_ChargingProfilePurposeType := E_OCPP_ChargingProfilePurposeTyp
e.None;
    nStackLevel             : UDINT := 0;
END_VAR
VAR_OUTPUT
    eStatus                 : E_OCPP1_ClearChargingProfileStatus;
END_VAR
```

Return value

Name	Type	Description
SendClearChargingProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

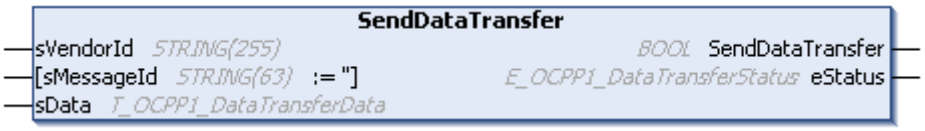
Name	Type	Description
nProfileId	UDINT	Identifier of the Charging Profile to be cleared.
nConnectorId	UDINT	ID of the Connector of a Charge Point. The value 0 indicates that the clearing relates to the entire Charge Point.
eChargingProfilePurpose	E_OCPP1_ChargingProfilePurposeType [▶ 191]	Specifies the purpose of the Charging Profiles to be cleared.
nStackLevel	UDINT	Specifies the StackLevel for which Charging Profiles are cleared.

Outputs

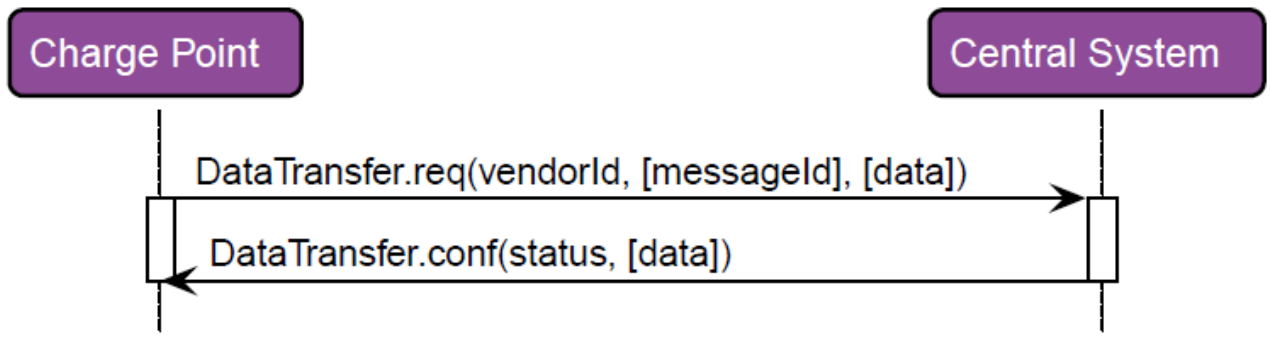
Name	Type	Description
eStatus	E_OCPP1_ClearChargingProfileStatus [▶ 192]	The status indicates whether the Charging Profile could be cleared from the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.35 SendDataTransfer



With this method, an OCPP server sends a Data Transfer request to the corresponding OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendDataTransfer : BOOL
VAR_INPUT
    sVendorId  : STRING(255);
    sMessageId : STRING(63) := '';
END_VAR
VAR_IN_OUT CONSTANT
    sData      : T_OCPP1_DataTransferData;
END_VAR
VAR_OUTPUT
    eStatus    : E_OCPP1_DataTransferStatus;
END_VAR
```

Return value

Name	Type	Description
SendDataTransfer	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sVendorId	STRING(255)	Identifier for the vendor, which identifies the vendor-specific implementation.
sMessageId	STRING(63)	Additional identification field for a single message.

Inputs/outputs

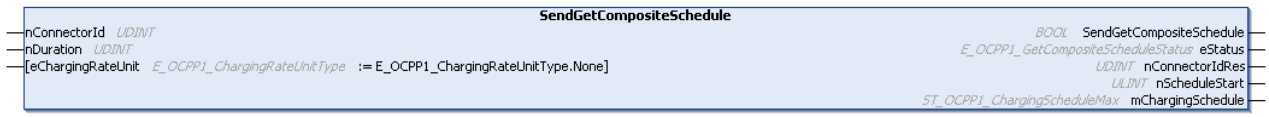
Name	Type	Description
sData	T_OCPP1_DataTransferData [▶ 208]	Text without specified length and format.

Outputs

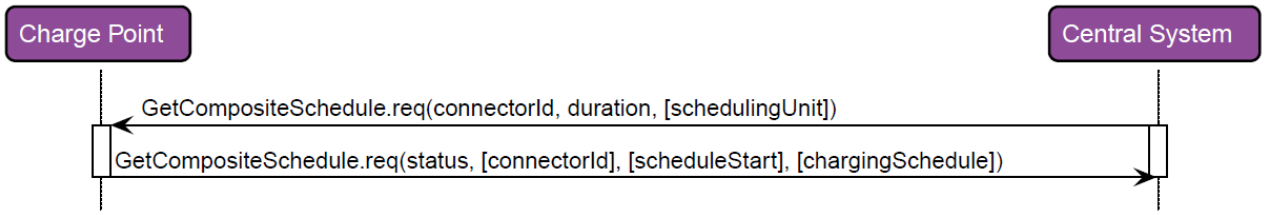
Name	Type	Description
eStatus	E_OCPP1_DataTransferStatus [▶ 193]	Status of the Data Transfer.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.36 SendGetCompositeSchedule



With this method, an OCPP server sends a Get Composite Schedule request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendGetCompositeSchedule : BOOL
VAR_INPUT
    nConnectorId      : UDINT;
    nDuration         : UDINT;
    eChargingRateUnit : E_OCPP1_ChargingRateUnitType := E_OCPP_ChargingRateUnitType.None;
END_VAR
VAR_OUTPUT
    eStatus           : E_OCPP1_GetCompositeScheduleStatus;
    nConnectorIdRes   : UDINT;
    nScheduleStart    : ULINT := 0;
    mChargingSchedule : ST_OCPP1_ChargingScheduleMax;
END_VAR
```

Return value

Name	Type	Description
SendGetCompositeSchedule	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nDuration	UDINT	Length of the requested schedule.
eChargingRateUnit	E_OCPP1_ChargingRateUnitType [▶ 192]	Used to specify the unit for the request.

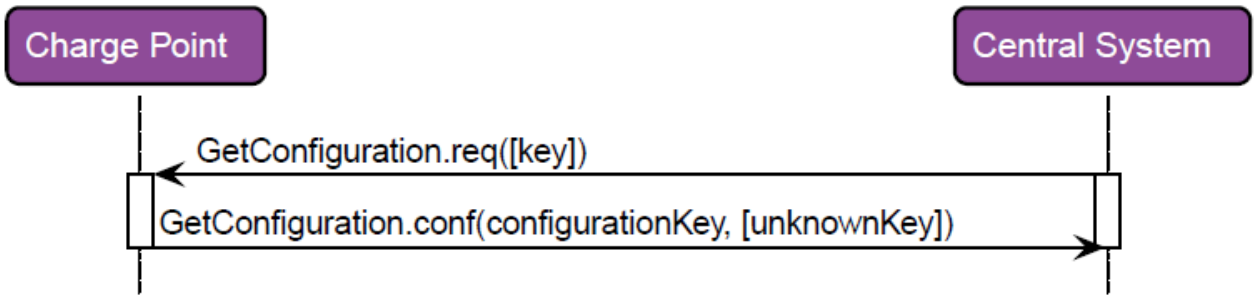
Outputs

Name	Type	Description
eStatus	E_OCPP1_GetCompositeScheduleStatus [▶ 194]	The status shows whether the schedule request could be answered by the Charge Point.
nConnectorIdRes	UDINT	Identifier of the Connector to which the schedule in the response refers.
nScheduleStart	ULINT	Start time of the schedule.
mChargingSchedule	ST_OCPP1_ChargingScheduleMax [▶ 204]	Requested schedule.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.37 SendGetConfiguration

With this method, an OCPP server sends a Get Configuration request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendGetConfiguration : BOOL
VAR_IN_OUT
    arrKeys       : ARRAY[*] OF ST_OCPP1_ConfigKeyValue;
END_VAR
VAR_OUTPUT
    nKeysCount    : UDINT := 0;
    nUnknownCount : UDINT := 0;
END_VAR
```

Return value

Name	Type	Description
SendGetConfiguration	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs/outputs

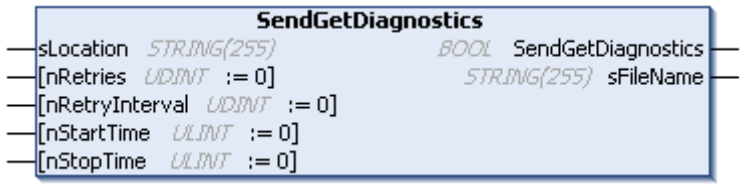
Name	Type	Description
arrKeys	ARRAY [*] OF ST_OCPP1_ConfigKeyValue [▶ 207]	List of requested or known configuration keys for which the configuration is requested.

Outputs

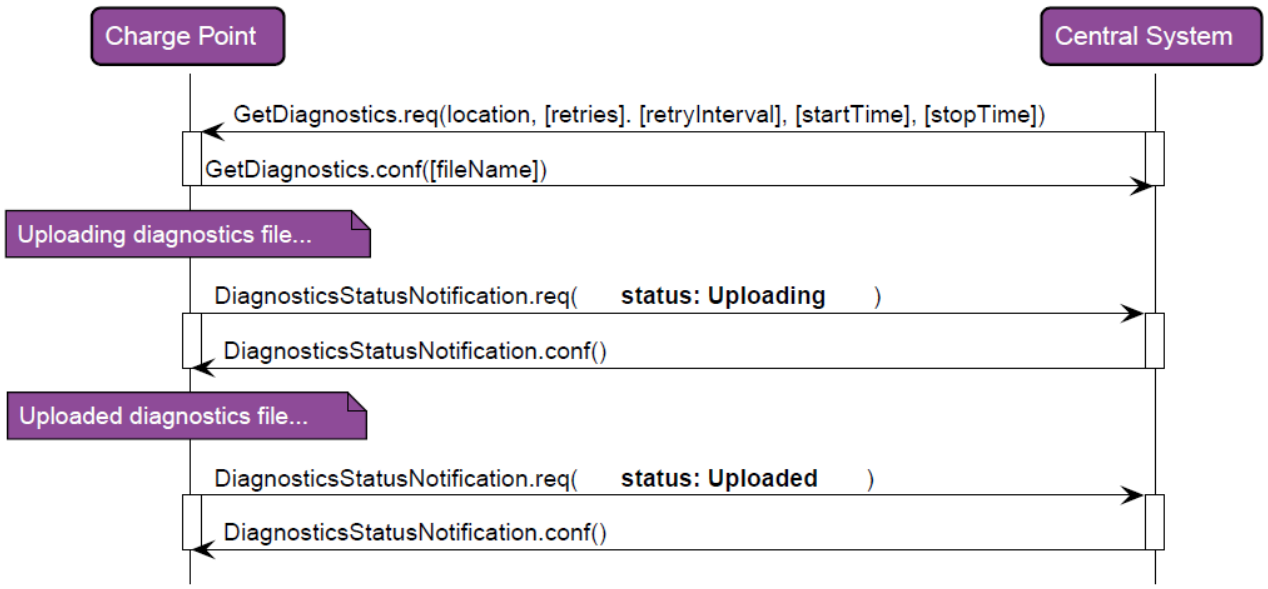
Name	Type	Description
nKeysCount	UDINT	Number of the following configuration keys.
nUnknownCount	UDINT	Number of the following unknown configuration keys.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.38 SendGetDiagnostics



With this method, an OCPP server sends a Get Diagnostics request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendGetDiagnostics : BOOL
VAR_INPUT
    sLocation      : STRING(255);
    nRetries       : UDINT := 0;
    nRetryInterval : UDINT := 0;
    nStartTime     : ULINT := 0;
    nStopTime      : ULINT := 0;
END_VAR
VAR_OUTPUT
    sFileName      : STRING(255);
END_VAR
```

Return value

Name	Type	Description
SendGetDiagnostics	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sLocation	STRING(255)	Directory to which the diagnostic file is to be uploaded.
nRetries	UDINT	Number of attempts made by the Charge Point if the upload fails.
nRetryInterval	UDINT	Interval after which a new upload is attempted.
nStartTime	ULINT	Start time of the logging information contained in the diagnostics.
nStopTime	ULINT	End time of the logging information contained in the diagnostics.

Outputs

Name	Type	Description
sFileName	STRING(255)	Contains the name of the diagnostic file that will be uploaded.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.39 SendGetLocalListVersion

With this method, an OCPP server sends a Get Local List Version request to the connected OCPP client. The response from the OCPP client is processed directly within the method.

Syntax

```
METHOD SendGetLocalListVersion : BOOL
VAR_OUTPUT
    nListVersion : UDINT;
END_VAR
```

Return value

Name	Type	Description
SendGetLocalListVersion	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

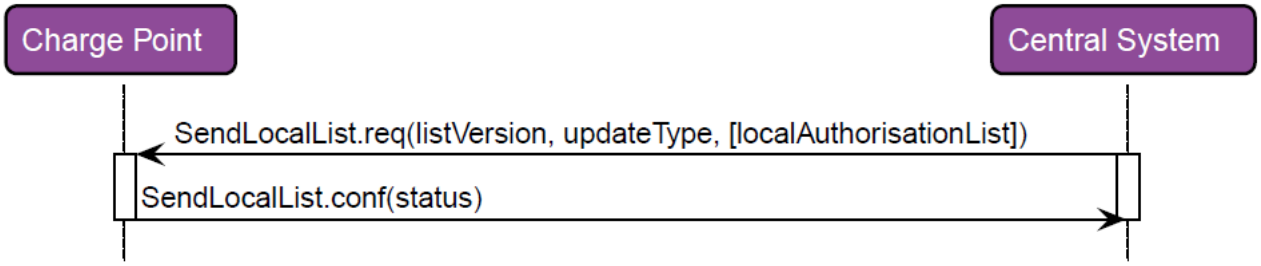
Outputs

Name	Type	Description
nListVersion	UDINT	Contains the version number of the Local Authorization List currently available in the Charge Point.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.40 SendLocalList

With this method, an OCPP server sends a Send Local List request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendLocalList : BOOL
VAR_INPUT
    nListVersion : UDINT;
    eUpdateType  : E_OCPP1_UpdateType;
    nAuthCount   : UDINT := 0;
END_VAR
VAR_IN_OUT
    arrAuthList  : ARRAY[*] OF ST_OCPP1_AuthorizationData;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_UpdateStatus;
END_VAR
```

Return value

Name	Type	Description
SendLocalList	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
nListVersion	UDINT	Version of the Local Authorization List provided as an update.
eUpdateType	E_OCPP1_UpdateType [▶ 197]	Definition of the update type, either as an update or as a complete replacement.
nAuthCount	UDINT	Number of entries in the Local Authorization List.

Inputs/outputs

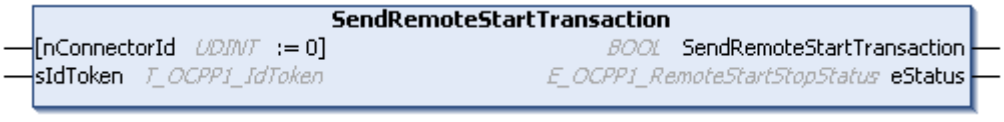
Name	Type	Description
arrAuthList	ARRAY [*] OF ST_OCPP1_AuthorizationData [▶ 202]	Either the values of the new Local Authorization List (complete replacement) or values to be added to the existing Local Authorization List (update).

Outputs

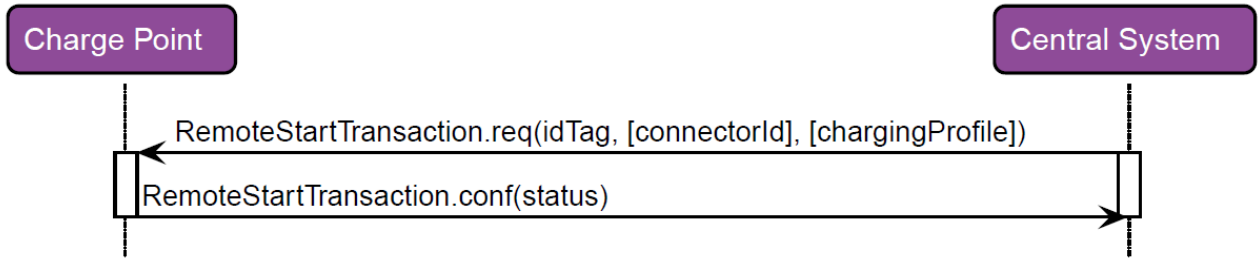
Name	Type	Description
eStatus	E_OCPP1_UpdateStatus [▶ 197]	The status indicates whether the Charge Point has received and carried out the Local Authorization List update.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.41 SendRemoteStartTransaction



With this method, an OCPP server sends a Remote Start Transaction request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendRemoteStartTransaction : BOOL
VAR_INPUT
    nConnectorId : UDINT := 0;
    sIdToken     : T_OCPP1_IdToken;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

Name	Type	Description
SendRemoteStartTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
sIdToken	T_OCPP1_IdToken [▶ 209]	ID token of the Charge Point in the Central System.

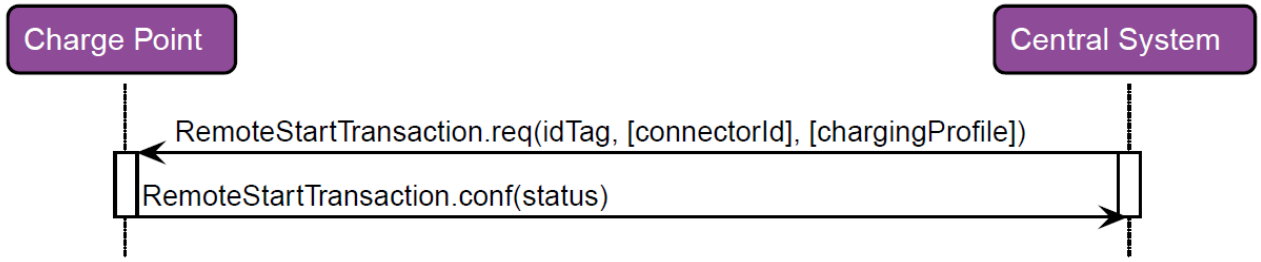
Outputs

Name	Type	Description
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status shows whether the Charge Point has accepted the request to start a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.42 SendRemoteStartTransactionProfile

With this method, an OCPP server sends a Remote Start Transaction request including a Charging Profile to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendRemoteStartTransactionProfile : BOOL
VAR_INPUT
    nConnectorId     : UDINT := 0;
    sIdToken         : T_OCPP1_IdToken;
    mChargingProfile : REFERENCE TO ST_OCPP1_ChargingProfile REF= 0;
END_VAR
VAR_OUTPUT
    eStatus          : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

Name	Type	Description
SendRemoteStartTransactionProfile	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

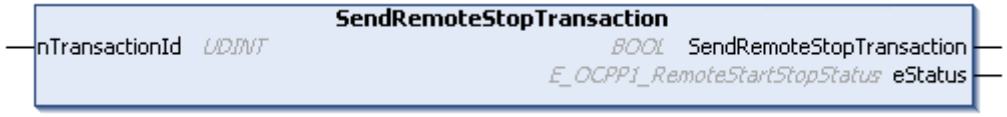
Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
sIdToken	T_OCPP1_IdToken [▶ 209]	ID token of the Charge Point in the Central System.
mChargingProfile	REFERENCE TO ST_OCPP1_ChargingProfileMax [▶ 204]	Charging Profile to be sent to the client.

Outputs

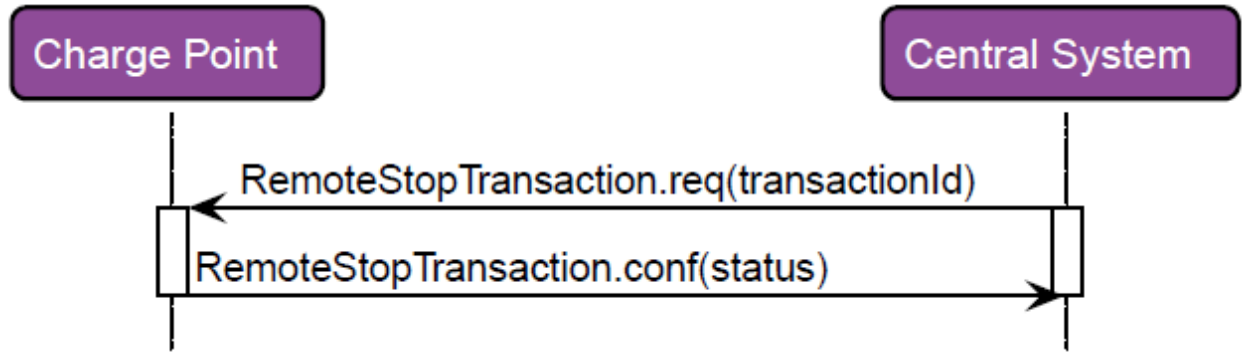
Name	Type	Description
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status shows whether the Charge Point has accepted the request to start a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.43 SendRemoteStopTransaction



With this method, an OCPP server sends a Remote Stop Transaction request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendRemoteStopTransaction : BOOL
VAR_INPUT
    nTransactionId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus        : E_OCPP1_RemoteStartStopStatus;
END_VAR
```

Return value

Name	Type	Description
SendRemoteStopTransaction	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
nTransactionId	UDINT	Identifier of the transaction to be stopped.

Outputs

Name	Type	Description
eStatus	E_OCPP1_RemoteStartStopStatus [▶ 196]	The status indicates whether the Charge Point has accepted the request to stop a transaction.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.44 SendReserveNow

With this method, an OCPP server sends a Reserve Now request to the connected OCPP client. The response from the OCPP client is processed directly within the method.

Syntax

```
METHOD SendReserveNow : BOOL
VAR_INPUT
    hStationId     : UDINT;
    nConnectorId   : UDINT;
    nExpiryDate    : ULINT;
    sIdTag         : T_OCPP1_IdToken;
    sParentIdTag   : T_OCPP1_IdToken := '';
    nReservationId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus        : E_OCPP1_ReservationStatus;
END_VAR
```

Return value

Name	Type	Description
SendReserveNow	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.
nExpiryDate	ULINT	Date and time of the end of the reservation.
sIdTag	T_OCPP1_IdToken [▶ 209]	Identifier for which the Charge Point should reserve a Connector.
sParentIdTag	T_OCPP1_IdToken [▶ 209]	The parent of the identifier for which the Charge Point is to reserve a Connector.
nReservationId	UDINT	Unique ID of the reservation.

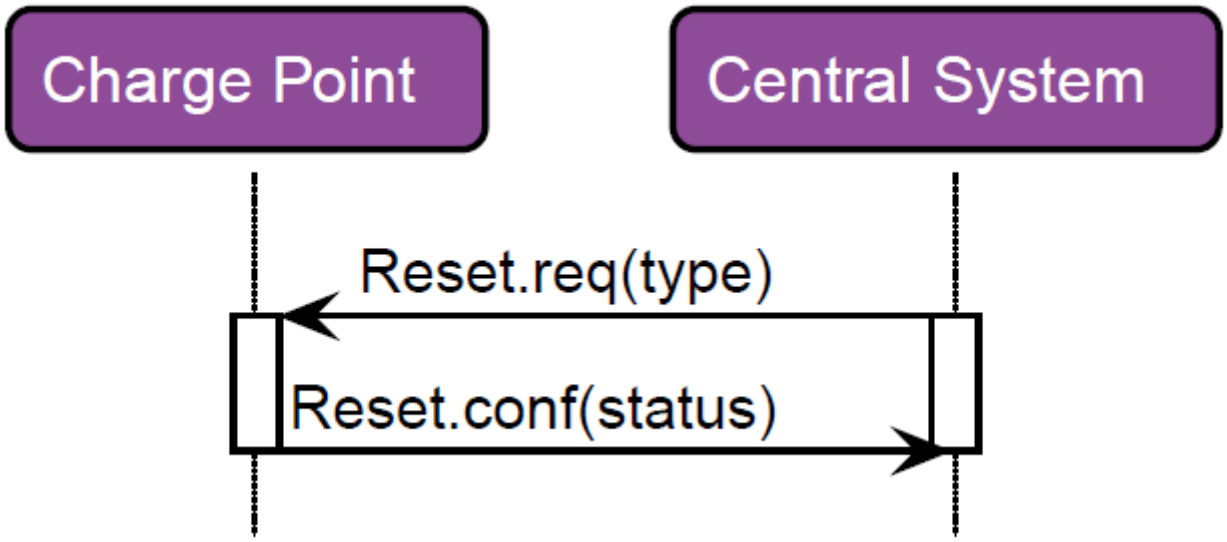
Outputs

Name	Type	Description
eStatus	E_OCPP1_ReservationStatus [▶ 196]	The status indicates whether the reservation was successful.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.45 SendReset

With this method, an OCPP server sends a Reset request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendReset : BOOL
VAR_INPUT
    eType       : E_OCPP1_ResetType;
END_VAR
VAR_OUTPUT
    eStatus     : E_OCPP1_ResetStatus;
END_VAR
```

Return value

Name	Type	Description
SendReset	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

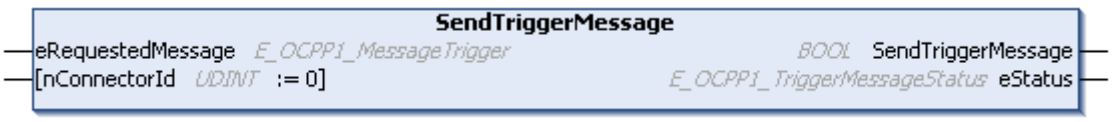
Name	Type	Description
eType	E_OCPP1_ResetType [▶ 196]	Type of reset that the Charge Point should perform.

Outputs

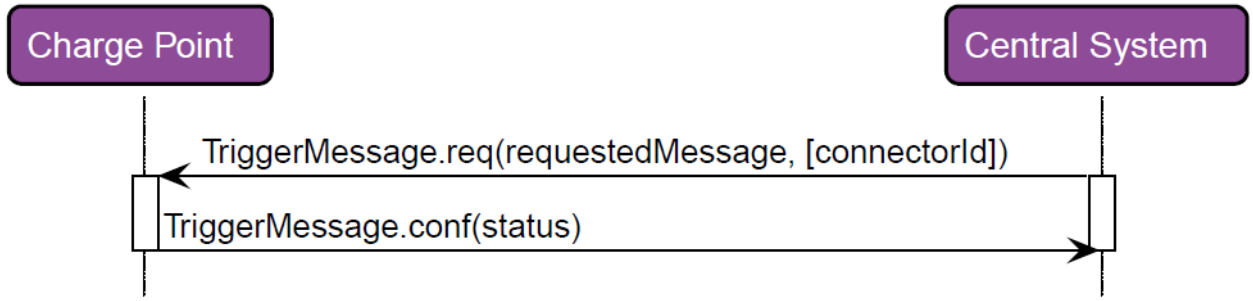
Name	Type	Description
eStatus	E_OCPP1_ResetStatus [▶ 196]	The status shows whether the Charge Point was able to accept the Reset request.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.46 SendTriggerMessage



With this method, an OCPP server sends a Trigger Message request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendTriggerMessage : BOOL
VAR_INPUT
    eRequestedMessage : E_OCPP1_MessageTrigger;
    nConnectorId      : UDINT := 0;
END_VAR
VAR_OUTPUT
    eStatus           : E_OCPP1_TriggerMessageStatus;
END_VAR
```

Return value

Name	Type	Description
SendTriggerMessage	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
eRequestedMessage	E_OCPP1_MessageTrigger [▶ 194]	Type of message to be triggered.
nConnectorId	UDINT	ID of the Connector of a Charge Point.

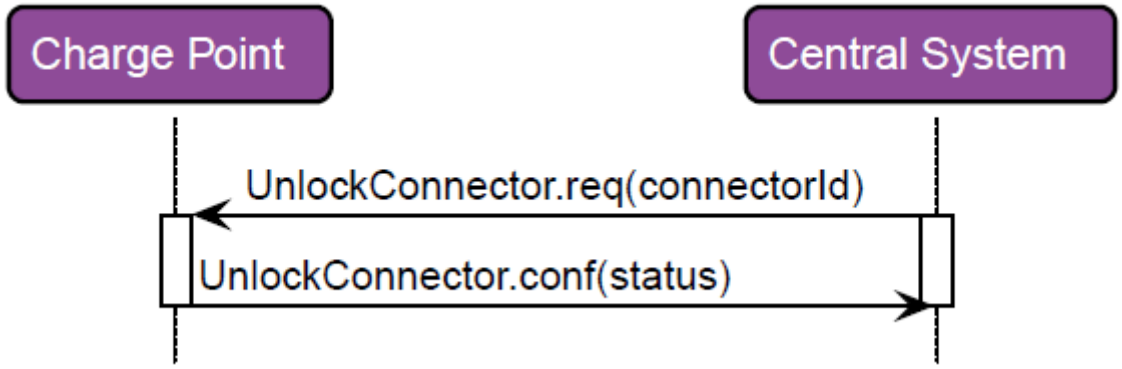
Outputs

Name	Type	Description
eStatus	E_OCPP1_TriggerMessageStatus [▶ 196]	The status indicates whether the Charge Point will send the requested message or not.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.47 SendUnlockConnector

With this method, an OCPP server sends an Unlock Connector request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendUnlockConnector : BOOL
VAR_INPUT
    nConnectorId : UDINT;
END_VAR
VAR_OUTPUT
    eStatus      : E_OCPP1_UnlockStatus;
END_VAR
```

Return value

Name	Type	Description
SendUnlockConnector	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

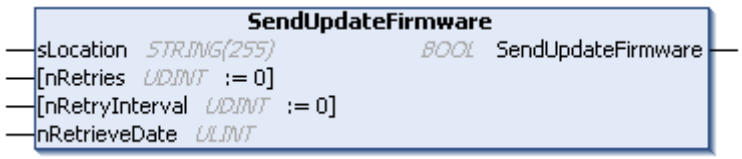
Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.

Outputs

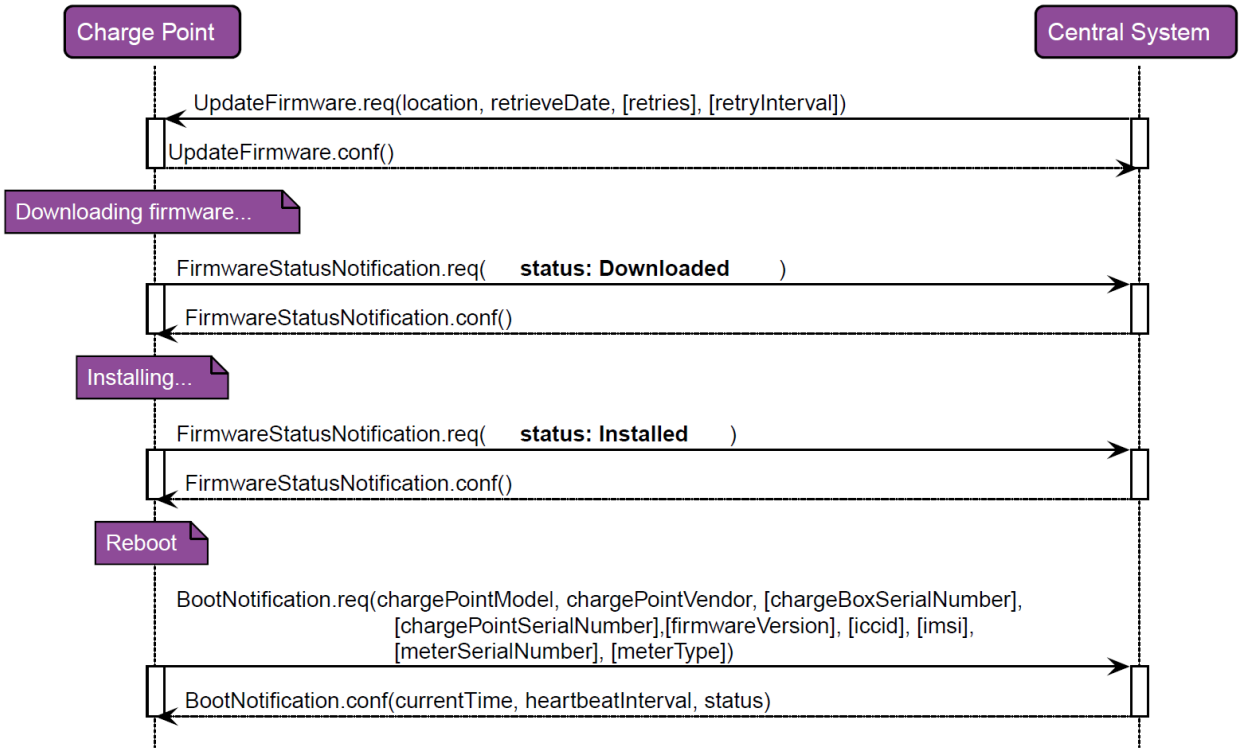
Name	Type	Description
eStatus	E_OCPP1_UnlockStatus [▶ 197]	The status shows whether the Connector has been unlocked.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.3.48 SendUpdateFirmware



With this method, an OCPP server sends an Update Firmware request to the connected OCPP client. The response from the OCPP client is processed directly within the method.



Syntax

```
METHOD SendUpdateFirmware : BOOL
VAR_INPUT
    sLocation      : STRING(255);
    nRetries       : UDINT := 0;
    nRetryInterval : UDINT := 0;
    nRetrieveDate  : ULINT;
END_VAR
```

Return value

Name	Type	Description
SendUpdateFirmware	BOOL	The method returns the return value TRUE if the call was successful. A method call is also considered successfully completed in the event of an error.

Inputs

Name	Type	Description
sLocation	STRING(255)	URI from which the firmware is to be retrieved.
nRetries	UDINT	Number of attempts to download the firmware again if the download fails.
nRetryInterval	UDINT	Time after which the download is attempted again.
nRetrieveDate	ULINT	Time and date from which the Charge Point may receive the new firmware.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

5.1.4 FB_OCPP1_ChargingProfile

This function block can be used to query the currently available Charging Profiles.

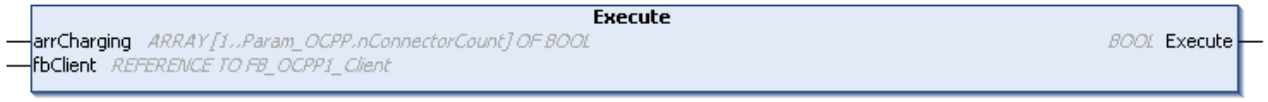
Syntax

```
FUNCTION BLOCK FB_OCPP1_ChargingProfile
VAR_OUTPUT
    nScheduleTime       : ULINT;
END_VAR
```

Outputs

Name	Type	Description
nScheduleTime	ULINT	The current timestamp used to schedule the Charging Profiles.

5.1.4.1 Execute



This method must be called cyclically if the function block should be used. The instance of FB_OCPP1_Client [▶ 21] must be transferred. In addition, information is required on which connector is currently charging. This allows the starting point of the Charging Schedule to be calculated.

Syntax

```
METHOD PUBLIC Execute : BOOL
VAR_INPUT
    arrCharging : ARRAY[1..Param_OCPP.nConnectorCount] OF BOOL;
    fbClient    : REFERENCE TO FB_OCPP1_Client;
END_VAR
```

Return value

Name	Type	Description
Execute	BOOL	The return value of the method is not currently assigned and always returns the value FALSE.

Inputs

Name	Type	Description
arrCharging	ARRAY [1..Param_OCPP [▶ 209].nConnectorCount] OF BOOL	Per connector within a Charge Point, the information as to whether charging is in progress or not.
fbClient	REFERENCE TO FB_OCPP1_Client [▶ 21]	Instance of the OCPP client.

5.1.4.2 GetCurrentPeriod

This method can be used to read the current Charging Schedule Period for a specific Connector of a Charge Point.

Syntax

```
METHOD PUBLIC GetCurrentPeriod : BOOL
VAR_INPUT
    nConnectorId : UDINT;
END_VAR
VAR_IN_OUT
    mPeriod      : ST_OCPP1_ChargingSchedulePeriod;
END_VAR
```

Return value

Name	Type	Description
GetCurrentPeriod	BOOL	The return value of the method is not currently assigned and always returns the value FALSE.

Inputs

Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.

Inputs/outputs

Name	Type	Description
mPeriod	ST_OCPP1_ChargingSchedulePeriod [▶ 206]	The current Charging Schedule Period at the Charge Point.

5.1.4.3 GetCurrentSchedule

This method can be used to read the current Charging Schedule for a specific Connector of a Charge Point.

Syntax

```
METHOD PUBLIC GetCurrentSchedule : BOOL
VAR_INPUT
    nConnectorId      : UDINT;
END_VAR
VAR_IN_OUT
    mChargingSchedule : ST_OCPP1_ChargingScheduleMax;
END_VAR
```

Return value

Name	Type	Description
GetCurrentSchedule	BOOL	The return value of the method is not currently assigned and always returns the value FALSE.

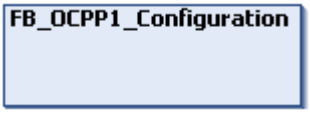
Inputs

Name	Type	Description
nConnectorId	UDINT	ID of the Connector of a Charge Point.

Inputs/outputs

Name	Type	Description
mChargingSchedule	ST_OCPP1_ChargingScheduleMax [▶ 205]	The current Charging Schedule at the Charge Point.

5.1.5 FB_OCPP1_Configuration



This function block can be used to manage the Configuration Keys of an instance of FB_OCPP1_Client [▶ 21].

Various methods are provided for the following purposes:

- Getting the last changes to Configuration Keys
- Initializing new Configuration Keys
- Initializing default values (for list see Configuration Keys [▶ 211])
- Setting new values for Configuration Keys
- Receiving Change Configuration and Get Configuration requests from an OCPP server with automatic response within the OCPP client

Syntax

```
FUNCTION BLOCK FB_OCPP1_Configuration
```

5.1.5.1 GetChanged

This method can be used to get the last change to the Config Keys that are part of the enumeration E_OCPP1_ConfigKey [▶ 192].

Syntax

```
METHOD GetChanged : BOOL
VAR_OUTPUT
    eKey   : E_OCPP1_ConfigKey;
    sValue : T_OCPP1_ConfigValue;
END_VAR
```

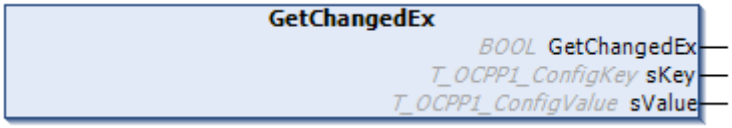
Return value

Name	Type	Description
GetChanged	BOOL	The method returns TRUE if the method call is successful.

Outputs

Name	Type	Description
eKey	E_OCPP1_ConfigKey [▶ 192]	Config Key for which the last change was made.
sValue	T_OCPP1_ConfigValue [▶ 208]	The corresponding value.

5.1.5.2 GetChangedEx



This method can be used to get the last change to the Config Keys that are not part of the enumeration E_OCPP1_ConfigKey [▶ 192].

Syntax

```
METHOD GetChangedEx : BOOL
VAR_OUTPUT
    sKey   : T_OCPP1_ConfigKey;
    sValue : T_OCPP1_ConfigValue;
END_VAR
```

Return value

Name	Type	Description
GetChanged	BOOL	The method returns TRUE if the method call is successful.

Outputs

Name	Type	Description
sKey	T_OCPP1_ConfigKey [▶ 208]	Config Key for which the last change was made.
sValue	T_OCPP1_ConfigValue [▶ 208]	The corresponding value.

5.1.5.3 GetValue

This method can be used to get the current value of a Config Key that is part of the enumeration E_OCPP1_ConfigKey [▶ 192].

Syntax

```
METHOD GetValue : T_OCPP1_ConfigValue
VAR_INPUT
    eKey : E_OCPP1_ConfigKey;
END_VAR
```

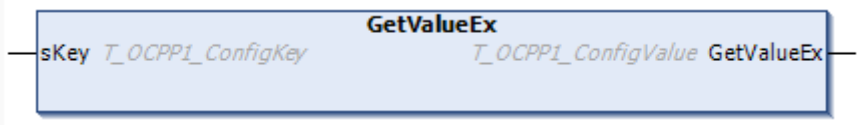
Return value

Name	Type	Description
GetValue	T_OCPP1_ConfigValue [▶ 208]	Current value of the queried Config Key.

Inputs

Name	Type	Description
eKey	E_OCPP1_ConfigKey [▶ 192]	Config Key for which the current value is to be queried.

5.1.5.4 GetValueEx



This method can be used to query the current value of a Config Key.

Syntax

```
METHOD GetValueEx : T_OCPP1_ConfigValue
VAR_INPUT
    sKey : T_OCPP1_ConfigKey;
END_VAR
```

Return value

Name	Type	Description
GetValue	T_OCPP1_ConfigValue [▶ 208]	Current value of the queried Config Key.

Inputs

Name	Type	Description
sKey	T_OCPP1_ConfigKey [▶ 208]	Config Key for which the current value is to be queried.

5.1.5.5 InitKeyValue

This method can be used to add new Key/Value pairs for Configuration Keys that are part of the enumeration E_OCPP1_ConfigKey [▶ 192]. If a Key is transferred that is already known in the OCPP client, the value is overwritten. The method InitKeyValueEx [▶ 185] can be used to add user-defined Keys.

Syntax

```
METHOD InitKeyValue : UDINT
VAR_INPUT
    eKey      : E_OCPP1_ConfigKey;
    sValue    : T_OCPP1_ConfigValue;
    bReadonly : BOOL;
END_VAR
```

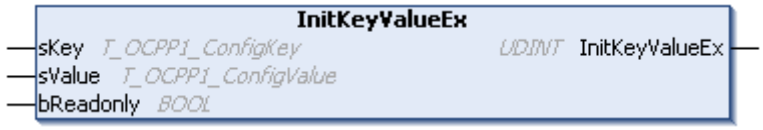
Return value

Name	Type	Description
InitKeyValue	UDINT	The assigned index of the Config Key. At present, however, this index is only used internally and does not need to be taken into account during programming.

Inputs

Name	Type	Description
eKey	E_OCPP1_ConfigKey [▶ 192]	Configuration Key to be overwritten or added.
sValue	T_OCPP1_ConfigValue [▶ 208]	Value of the associated Configuration Key.
bReadonly	BOOL	Determines the access right of the Central System, either the value may only be read (TRUE) or also written (FALSE).

5.1.5.6 InitKeyValueEx



This method can be used to add new Key/Value pairs for Configuration Keys. If a Key is transferred that is already known in the OCPP client, the value is overwritten. The method can be used to add user-defined Keys that are not part of the enumeration E_OCPP1_ConfigKey [▶ 192].

Syntax

```
METHOD InitKeyValueEx : UDINT
VAR_INPUT
    sKey      : T_OCPP1_ConfigKey;
    sValue    : T_OCPP1_ConfigValue;
    bReadonly : BOOL;
END_VAR
```

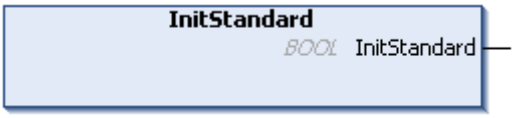
Return value

Name	Type	Description
InitKeyValueEx	UDINT	The assigned index of the Config Key. At present, however, this index is only used internally and does not need to be taken into account during programming.

Inputs

Name	Type	Description
sKey	T_OCPP1_ConfigKey [▶ 208]	Configuration Key to be overwritten or added.
sValue	T_OCPP1_ConfigValue [▶ 208]	Value of the associated Configuration Key.
bReadonly	BOOL	Determines the access right of the Central System, either the value may only be read (TRUE) or also written (FALSE).

5.1.5.7 InitStandard



This method is used to set default values for some of the Configuration Keys described in the OCPP specification. The list of Configuration Keys and their default assignment can be found in the appendix [▶ 211].

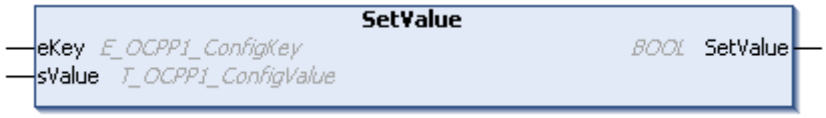
Syntax

```
METHOD InitStandard : BOOL
```

Return value

Name	Type	Description
InitStandard	BOOL	The return value of the method is not currently assigned and always returns the value FALSE.

5.1.5.8 SetValue



This method can be used to write a new value for a Config Key from the enumeration E_OCPP1_ConfigKey [▶ 192].

Syntax

```
METHOD SetValue : BOOL
VAR_INPUT
    eKey : E_OCPP1_ConfigKey;
END_VAR
VAR_IN_OUT CONSTANT
    sValue : T_OCPP1_ConfigValue;
END_VAR
```

Return value

Name	Type	Description
SetValue	BOOL	The method returns TRUE if the method call is successful.

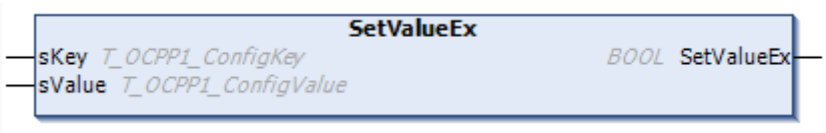
Inputs

Name	Type	Description
eKey	E_OCPP1_ConfigKey [▶ 192]	Config Key that is to be rewritten.

Inputs/outputs

Name	Type	Description
sValue	T_OCPP1_ConfigValue [▶ 208]	New value for the Config Key to be written.

5.1.5.9 SetValueEx



This method can be used to set a new value for a Config Key.

Syntax

```
METHOD SetValueEx : BOOL
VAR_INPUT
    sKey : T_OCPP1_ConfigKey;
END_VAR
VAR_IN_OUT CONSTANT
    sValue : T_OCPP1_ConfigValue;
END_VAR
```

Return value

Name	Type	Description
SetValueEx	BOOL	The method returns TRUE if the method call is successful.

Inputs

Name	Type	Description
sKey	T_OCPP1_ConfigKey [▶ 208]	Config Key that is to be rewritten.

Inputs/outputs

Name	Type	Description
sValue	T_OCPP1_ConfigValue [▶ 208]	New value for the Config Key to be written.

Also see about this

- E_OCPP1_ConfigKey [▶ 192]

5.1.5.10 RecvChangeConfiguration

This method is an alternative way of receiving and responding to a Change Configuration Request from the corresponding OCPP server in the OCPP client. In contrast to the RecvChangeConfiguration [▶ 29] method at FB_OCPP1_Client [▶ 21], the response is sent automatically to the OCPP server.

If the Configuration Keys to be written are read-only, they are not written. All writable Configuration Keys are saved by the method.

Syntax

```
METHOD RecvChangeConfiguration : BOOL
VAR_INPUT
    fbClient : REFERENCE TO FB_OCPP1_Client;
END_VAR
```

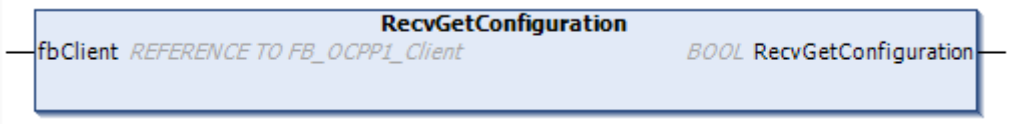
Return value

Name	Type	Description
RecvChangeConfiguration	BOOL	Not currently implemented. In future versions: if the method call is successful, the method returns TRUE.

Inputs/outputs

Name	Type	Description
fbClient	REFERENCE TO FB_OCPP1_Client [▶ 21]	Instance of the client function block used.

5.1.5.11 RecvGetConfiguration



This method is an alternative way of receiving and responding to a Get Configuration Request from the corresponding OCPP server in the OCPP client. In contrast to the RecvGetConfiguration [▶ 34] method at FB_OCPP1_Client [▶ 21], the response is sent automatically to the OCPP server. Depending on the request, it contains all or specifically requested Configuration Keys.

Syntax

```
METHOD RecvGetConfiguration : BOOL
VAR_INPUT
    fbClient : REFERENCE TO FB_OCPP1_Client;
END_VAR
```

Return value

Name	Type	Description
RecvGetConfiguration	BOOL	Not currently implemented. In future versions: if the method call is successful, the method returns TRUE.

Inputs/outputs

Name	Type	Description
fbClient	REFERENCE TO FB_OCPP1_Client [▶ 21]	Instance of the client function block used.

5.2 DUTs

5.2.1 Enumerations

5.2.1.1 E_OCPP1_Action

```
TYPE E_OCPP1_Action :
(
    None                            := 0,
    // Core
    BootNotification                := 10001,
    Heartbeat                       := 10002,
    MeterValues                     := 10003,
    StatusNotification              := 10004,
    Authorize                       := 10005,
    StartTransaction                := 10006,
    StopTransaction                 := 10007,
    RemoteStartTransaction          := 10011,
    RemoteStopTransaction           := 10012,
    ChangeAvailability              := 10013,
    ChangeConfiguration             := 10014,
    GetConfiguration                := 10015,
    UnlockConnector                 := 10016,
    ClearCache                      := 10017,
    Reset                           := 10018,
    // Remote Trigger
    TriggerMessage                  := 10021,
    // Reservation
    ReserveNow                      := 10031,
    CancelReservation               := 10032,
    // Smart Charging
    SetChargingProfile              := 10041,
    GetCompositeSchedule            := 10042,
    ClearChargingProfile            := 10043,
    // Authentication
    GetLocalListVersion             := 10051,
    SendLocalList                   := 10052,
    // Firmware
    DiagnosticsStatusNotification   := 10061,
    FirmwareStatusNotification      := 10062,
    GetDiagnostics                  := 10063,
    UpdateFirmware                  := 10064,
    // Vendor
    DataTransfer                    := 10091,
    // Security
    SecurityEventNotification       := 11001,
    ExtendedTriggerMessage          := 11002,
    LogStatusNotification           := 11003,
    GetLog                          := 11004,
    GetInstalledCertificateIds      := 11005,
    CertificateSigned               := 11006,
    SignCertificate                 := 11007,
    InstallCertificate              := 11008,
    DeleteCertificate               := 11009,
```

```
    SignedFirmwareStatusNotification := 11010,
    SignedUpdateFirmware             := 11011
) UDINT;
END_TYPE
```

5.2.1.2 E_OCPP1_AuthenticationMode

```
TYPE E_OCPP1_AuthenticationMode :
(
    None  := 0, // No Authentication
    Ident := 1, // Ident Authentication
    Basic := 2  // Basic Authentication
) UDINT;
END_TYPE
```

5.2.1.3 E_OCPP1_AuthorizationStatus

```
TYPE E_OCPP1_AuthorizationStatus :
(
    None := 0,       // Any status not covered by this implementation
    Accepted,        // Identifier is allowed for charging.
    Blocked,         // Identifier has been blocked. Not allowed for charging.
    Expired,         // Identifier has expired. Not allowed for charging.
    Invalid,         // Identifier is unknown. Not allowed for charging.
    ConcurrentTx     // Identifier is already involved in another transaction and multiple transacti
ons are not allowed.
) UDINT;
END_TYPE
```

5.2.1.4 E_OCPP1_AvailabilityStatus

```
TYPE E_OCPP1_AvailabilityStatus :
(
    None     := 0, // Any status not covered by this implementation
    Accepted,      // Request has been accepted and will be executed.
    Rejected,      // Request has not been accepted and will not be executed.
    Scheduled      // Request has been accepted and will be executed when transaction(s) in progress
 have finished.
) UDINT;
END_TYPE
```

5.2.1.5 E_OCPP1_AvailabilityType

```
TYPE E_OCPP1_AvailabilityType :
(
    None := 0,        // Any status not covered by this implementation
    Inoperative,      // Charge point is not available for charging.
    Operative         // Charge point is available for charging.
) UDINT;
END_TYPE
```

5.2.1.6 E_OCPP1_CancelReservationStatus

```
TYPE E_OCPP1_CancelReservationStatus :
(
    None := 0,     // Any status not covered by this implementation
    Accepted,      // Reservation for the identifier has been cancelled.
    Rejected       // Reservation could not be cancelled, because there is no reservation active for
 the identifier.
) UDINT;
END_TYPE
```

5.2.1.7 E_OCPP1_ChargePointError

```
TYPE E_OCPP1_ChargePointError :
(
    None := 0,            // Any status not covered by this implementation
    NoError,              // No error to report.
    ConnectorLockFailure, // Failure to lock or unlock connector.
    EVCommunicationError, // Communication failure with the vehicle, might be Mode 3 or other commun
ication protocol problem.
    GroundFailure,        // Ground fault circuit interrupter has been activated.
    HighTemperature,      // Temperature inside Charge Point is too high.
```

```
    InternalError,        // Error in internal hard - or software component.
    LocalListConflict,    // The authorization information received from the Central System is in co
nflict with the LocalAuthorizationList.
    OverCurrentFailure,   // Over current protection device has tripped.
    OverVoltage,          // Voltage has risen above an acceptable level.
    PowerMeterFailure,    // Failure to read electrical / energy / power meter.
    PowerSwitchFailure,   // Failure to control power switch.
    ReaderFailure,        // Failure with idTag reader.
    ResetFailure,         // Unable to perform a reset.
    UnderVoltage,         // Voltage has dropped below an acceptable level.
    WeakSignal,           // Wireless communication device reports a weak signal.
    OtherError            // Other type of error.More information in vendorErrorCode.
) UDINT;
END_TYPE
```

5.2.1.8 E_OCPP1_ChargePointStatus

```
TYPE E_OCPP1_ChargePointStatus :
(
    None := 0,     // Any status not covered by this implementation
    Available,     // When a Connector becomes available for a new user. (Operative)
    Preparing,     // When a Connector becomes no longer available for a new user but there is no on
going Transaction. (Operative)
    Charging,      // When the contactor of a Connector closes, allowing the vehicle to charge. (Ope
rative)
    SuspendedEVSE, // When the EV is connected to the EVSE but the EVSE is not offering energy to th
e EV. (Operative)
    SuspendedEV,   // When the EV is connected to the EVSE and the EVSE is offering energy but the E
V is not taking any energy. (Operative)
    Finishing,     // When a Transaction has stopped at a Connector, but the Connector is not yet av
ailable for a new user. (Operative)
    Reserved,      // When a Connector becomes reserved as a result of a Reserve Now command. (Opera
tive)
    Unavailable,   // When a Connector becomes unavailable as the result of a Change Availability co
mmand or an event upon which the Charge Point transitions to unavailable at its discretion. (Inopera
tive)
    Faulted        // When a Charge Point or connector has reported an error and is not available fo
r energy delivery . (Inoperative)
) UDINT;
END_TYPE
```

5.2.1.9 E_OCPP1_ChargingProfileKindType

```
TYPE E_OCPP1_ChargingProfileKindType :
(
    None := 0, // Any status not covered by this implementation
    Absolute,  // Schedule periods are relative to a fixed point in time defined in the schedule.
    Recurring, // The schedule restarts periodically at the first schedule period.
    Relative   // Schedule periods are relative to a situation-
specific start point that is determined by the charge point.
) UDINT;
END_TYPE
```

5.2.1.10 E_OCPP1_ChargingProfilePurposeType

```
TYPE E_OCPP1_ChargingProfilePurposeType :
(
    None := 0,             // Any status not covered by this implementation
    ChargePointMaxProfile, // Configuration for the maximum power or current available for an entire
 Charge Point.
    TxDefaultProfile,      // Default profile that can be configured in the Charge Point.
    TxProfile              // Profile with constraints to be imposed by the Charge Point on the curr
ent transaction.
) UDINT;
END_TYPE
```

5.2.1.11 E_OCPP1_ChargingProfileStatus

```
TYPE E_OCPP1_ChargingProfileStatus :
(
    None := 0,   // Any status not covered by this implementation
    Accepted,    // Request has been acceptedand will be executed.
    Rejected,    // Request has not been accepted and will not be executed.
    NotSupported // Charge Point indicates that the request is not supported.
) UDINT;
END_TYPE
```

5.2.1.12 E_OCPP1_ChargingRateUnitType

```
TYPE E_OCPP1_ChargingRateUnitType :
(
    None := 0, // Any status not covered by this implementation
    W,         // Watts (power).
    A          // Amperes (current).
) UDINT;
END_TYPE
```

5.2.1.13 E_OCPP1_ClearCacheStatus

```
TYPE E_OCPP1_ClearCacheStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Command has been executed.
    Rejected   // Command has not been executed.
) UDINT;
END_TYPE
```

5.2.1.14 E_OCPP1_ClearChargingProfileStatus

```
TYPE E_OCPP1_ClearChargingProfileStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Request has been accepted and will be executed.
    Unknown    // No Charging Profile(s) were found matching the request.
) UDINT;
END_TYPE
```

5.2.1.15 E_OCPP1_ConfigKey

```
TYPE E_OCPP1_ConfigKey :
(
    None := 0,
    AllowOfflineTxForUnknownId, // optional
    AuthorizationCacheEnabled, // optional
    AuthorizeRemoteTxRequests,
    BlinkRepeat,     // optional
    ClockAlignedDataInterval,
    ConnectionTimeOut,
    ConnectorPhaseRotation,
    ConnectorPhaseRotationMaxLength, // optional readonly
    GetConfigurationMaxKeys,// readonly
    HeartbeatInterval,
    LightIntensity, // optional
    LocalAuthorizeOffline,
    LocalPreAuthorize,
    MeterValuesAlignedData,
    MeterValuesAlignedDataMaxLength, // optional readonly
    MeterValuesSampledData,
    MeterValuesSampledDataMaxLength, // optional readonly
    MeterValueSampleInterval,
    MinimumStatusDuration, // optional
    NumberOfConnectors, // readonly
    ResetRetries,
    StopTransactionOnEVSideDisconnect,
    StopTransactionOnInvalidId,
    StopTxnAlignedData,
    StopTxnAlignedDataMaxLength, // optional readonly
    StopTxnSampledData,
    StopTxnSampledDataMaxLength, // optional readonly
    SupportedFeatureProfiles, // readonly
    SupportedFeatureProfilesMaxLength, // optional readonly
    TransactionMessageAttempts,
    TransactionMessageRetryInterval,
    UnlockConnectorOnEVSideDisconnect,
    WebSocketPingInterval, // optional
    LocalAuthListEnabled,
    LocalAuthListMaxLength, // readonly
    SendLocalListMaxLength, // readonly
    ReserveConnectorZeroSupported, // optional readonly
    ChargeProfileMaxStackLevel, // Readonly
    ChargingScheduleAllowedChargingRateUnit, // readonly
    ChargingScheduleMaxPeriods, // readonly
    ConnectorSwitch3to1PhaseSupported, // optional readonly
    MaxChargingProfilesInstalled, // readonly
```

```
    MaxKeys // amount of keys +1, not a Configkey!
) UDINT;
END_TYPE
```

5.2.1.16 E_OCPP1_ConfigurationStatus

```
TYPE E_OCPP1_ConfigurationStatus :
(
    None := 0,      // Any status not covered by this implementation
    Accepted,       // Configuration key is supported and setting has been changed.
    Rejected,       // Configuration key is supported, but setting could not be changed.
    RebootRequired, // Configuration key is supported and setting has been changed, but change will
be available after reboot.
    NotSupported    // Configuration key is not supported.
) UDINT;
END_TYPE
```

5.2.1.17 E_OCPP1_DataTransferStatus

```
TYPE E_OCPP1_DataTransferStatus :
(
    None := 0,        // Any status not covered by this implementation
    Accepted,         // Message has been accepted and the contained request is accepted.
    Rejected,         // Message has been accepted but the contained request is rejected.
    UnknownMessageId, // Message could not be interpreted due to unknown messageId string.
    UnknownVendorId   // Message could not be interpreted due to unknown vendorId string.
) UDINT;
END_TYPE
```

5.2.1.18 E_OCPP1_DiagnosticsStatus

```
TYPE E_OCPP1_DiagnosticsStatus :
(
    None := 0,    // Any status not covered by this implementation
    Idle,         // Charge Point is not performing diagnostics related tasks.
    Uploaded,     // Diagnostics information has been uploaded.
    UploadFailed, // Uploading of diagnostics failed.
    Uploading     // File is being uploaded.
) UDINT;
END_TYPE
```

5.2.1.19 E_OCPP1_Error

```
TYPE E_OCPP1_Error :
(
    None := 0,                    // Any error not covered by this implementation
    GenericError,                 // Any other error not covered by the previous ones
    NotImplemented,               // Requested Action is not known by receiver
    NotSupported,                 // Requested Action is recognized but not supported by the receive
r
    InternalError,                // An internal error occurred and the receiver was not able to pro
cess the requested Action successfully
    ProtocolError,                // Payload for Action is incomplete
    SecurityError,                // During the processing of Action a security issue occurred preve
nting receiver from completing the Action successfully
    FormationViolation,           // Payload for Action is syntactically incorrect or not conform th
e PDU structure for Action
    PropertyConstraintViolation,  // Payload is syntactically correct but at least one field contain
s an invalid value
    OccurenceConstraintViolation, // Payload for Action is syntactically correct but at least one of
 the fields violates occurence constraints
    TypeConstraintViolation       // Payload for Action is syntactically correct but at least one of
 the fields violates data type constraints
) UDINT;
END_TYPE
```

5.2.1.20 E_OCPP1_FirmwareStatus

```
TYPE E_OCPP1_FirmwareStatus :
(
    None := 0,          // Any status not covered by this implementation
    Downloaded,         // New firmware has been downloaded by Charge Point.
    DownloadFailed,     // Charge point failed to download firmware.
    Downloading,        // Firmware is being downloaded.
    Idle,               // Charge Point is not performing firmware update related tasks.
```

```
    InstallationFailed, // Installation of new firmware has failed.
    Installing,         // Firmware is being installed.
    Installed           // New firmware has successfully been installed in charge point.
) UDINT;
END_TYPE
```

5.2.1.21 E_OCPP1_GetCompositeScheduleStatus

```
TYPE E_OCPP1_GetCompositeScheduleStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Request has been accepted and will be executed.
    Rejected   // Request has not been accepted and will not be executed.
) UDINT;
END_TYPE
```

5.2.1.22 E_OCPP1_Location

```
TYPE E_OCPP1_Location :
(
    None := 0, // Any status not covered by this implementation
    Body,      // Measurement inside body of Charge Point(e.g.Temperature)
    Cable,     // Measurement taken from cable between EV and Charge Point
    EV,        // Measurement taken by EV
    Inlet,     // Measurement at network("grid") inlet connection
    Outlet     // Measurement at a Connector.Default value
) UDINT;
END_TYPE
```

5.2.1.23 E_OCPP1_Measurand

```
TYPE E_OCPP1_Measurand :
(
    None := 0,                    // Any status not covered by this implementation
    CurrentExport,                // Instantaneous current flow from EV
    CurrentImport,                // Instantaneous current flow to EV
    CurrentOffered,               // Maximum current offered to EV
    EnergyActiveExportRegister,   // Numerical value read from the "active electrical energy" (Wh or
 kWh) register of the (most authoritative) electrical meter measuring energy exported (to the grid).
    EnergyActiveImportRegister,   // Numerical value read from the "active electrical energy" (Wh or
 kWh) register of the (most authoritative) electrical meter measuring energy imported (from the grid
 supply).
    EnergyReactiveExportRegister, // Numerical value read from the "reactive electrical energy" (VAR
h or kVARh) register of the (most authoritative) electrical meter measuring energy exported (to the
grid).
    EnergyReactiveImportRegister, // Numerical value read from the "reactive electrical energy" (VAR
h or kVARh) register of the (most authoritative) electrical meter measuring energy imported (from th
e grid supply).
    EnergyActiveExportInterval,   // Absolute amount of "active electrical energy" (Wh or kWh) expor
ted (to the grid) during an associated time "interval".
    EnergyActiveImportInterval,   // Absolute amount of "active electrical energy" (Wh or kWh) impor
ted(from the grid supply) during an associated time "interval".
    EnergyReactiveExportInterval, // Absolute amount of "reactive electrical energy" (VARh or kVARh)
 exported (to the grid) during an associated time "interval".
    EnergyReactiveImportInterval, // Absolute amount of "reactive electrical energy" (VARh or kVARh)
 imported (from the grid supply) during an associated time "interval".
    Frequency,                    // Instantaneous reading of powerline frequency.
    PowerActiveExport,            // Instantaneous active power exported by EV. (W or kW)
    PowerActiveImport,            // Instantaneous active power imported by EV. (W or kW)
    PowerFactor,                  // Instantaneous power factor of total energy flow
    PowerOffered,                 // Maximum power offered to EV
    PowerReactiveExport,          // Instantaneous reactive power exported by EV. (var or kvar)
    PowerReactiveImport,          // Instantaneous reactive power imported by EV. (var or kvar)
    RPM,                          // Fan speed in RPM
    SoC,                          // State of charge of charging vehicle in percentage
    Temperature,                  // Temperature reading inside Charge Point.
    Voltage                       // Instantaneous AC RMS supply voltage
) UDINT;
END_TYPE
```

5.2.1.24 E_OCPP1_MessageTrigger

```
TYPE E_OCPP1_MessageTrigger :
(
    None := 0,                     // Any status not covered by this implementation
    BootNotification,              // To trigger a BootNotification request
```

```
    DiagnosticsStatusNotification, // To trigger a DiagnosticsStatusNotification request
    FirmwareStatusNotification,    // To trigger a FirmwareStatusNotification request
    Heartbeat,                     // To trigger a Heartbeat request
    MeterValues,                   // To trigger a MeterValues request
    StatusNotification             // To trigger a StatusNotification request
) UDINT;
END_TYPE
```

5.2.1.25 E_OCPP1_Phase

```
TYPE E_OCPP1_Phase :
(
    None := 0, // Any status not covered by this implementation
    L1,        // Measured on L1
    L2,        // Measured on L2
    L3,        // Measured on L3
    N,         // Measured on Neutral
    L1N,       // Measured on L1 with respect to Neutral conductor
    L2N,       // Measured on L2 with respect to Neutral conductor
    L3N,       // Measured on L3 with respect to Neutral conductor
    L12,       // Measured between L1 and L2
    L23,       // Measured between L2 and L3
    L31        // Measured between L3 and L1
) UDINT;
END_TYPE
```

5.2.1.26 E_OCPP1_ReadingContext

```
TYPE E_OCPP1_ReadingContext :
(
    None := 0,         // Any status not covered by this implementation
    InterruptionBegin, // Value taken at start of interruption.
    InterruptionEnd,   // Value taken when resuming after interruption.
    Other,             // Value for any other situations.
    SampleClock,       // Value taken at clock aligned interval.
    SamplePeriodic,    // Value taken as periodic sample relative to start time of transaction.
    TransactionBegin,  // Value taken at start of transaction.
    TransactionEnd,    // Value taken at end of transaction.
    Trigger            // Value taken in response to a TriggerMessage
) UDINT;
END_TYPE
```

5.2.1.27 E_OCPP1_Reason

```
TYPE E_OCPP1_Reason :
(
    None := 0,      // Any status not covered by this implementation
    DeAuthorized,   // The transaction was stopped because of the authorization status in a StartTra
nsaction response
    EmergencyStop,  // Emergency stop button was used.
    EVDisconnected, // Disconnecting of cable, vehicle moved away from inductive charge unit.
    HardReset,      // A hard reset command was received.
    Local,          // Stopped locally on request of the user at the Charge Point.
    Other,          // Any other reason.
    PowerLoss,      // Complete loss of power.
    Reboot,         // A locally initiated reset / reboot occurred. (for instance watchdog kicked in
)
    Remote,         // Stopped remotely on request of the user. This is a regular termination of a t
ransaction.
    SoftReset,      // A soft reset command was received.
    UnlockCommand   // Central System sent an Unlock Connector command.
) UDINT;
END_TYPE
```

5.2.1.28 E_OCPP1_RecurrencyKindType

```
TYPE E_OCPP1_RecurrencyKindType :
(
    None := 0, // Any status not covered by this implementation
    Daily,     // The schedule restarts every 24 hours.
    Weekly     // The schedule restarts every 7 days.
) UDINT;
END_TYPE
```

5.2.1.29 E_OCPP1_RegistrationStatus

```
TYPE E_OCPP1_RegistrationStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Charge point is accepted by Central System.
    Pending,   // Central System is not yet ready to accept the Charge Point.
    Rejected   // Charge point is not accepted by Central System. This may happen when the Charge Po
int id is not known by Central System.
) UDINT;
END_TYPE
```

5.2.1.30 E_OCPP1_RemoteStartStopStatus

```
TYPE E_OCPP1_RemoteStartStopStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Command will be executed.
    Rejected   // Command will not be executed.
) UDINT;
END_TYPE
```

5.2.1.31 E_OCPP1_ReservationStatus

```
TYPE E_OCPP1_ReservationStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Reservation has been made.
    Faulted,   // Reservation has not been made, because connectors or specified connector are in a
faulted state.
    Occupied,  // Reservation has not been made.All connectors or the specified connector are occupi
ed.
    Rejected   // Reservation has not been made.Charge Point is not configured to accept reservation
s.
) UDINT;
END_TYPE
```

5.2.1.32 E_OCPP1_ResetStatus

```
TYPE E_OCPP1_ResetStatus :
(
    None := 0, // Any status not covered by this implementation
    Accepted,  // Configuration key is supported and setting has been changed.
    Rejected   // Configuration key is supported, but setting could not be changed.
) UDINT;
END_TYPE
```

5.2.1.33 E_OCPP1_ResetType

```
TYPE E_OCPP1_ResetType :
(
    None := 0, // Any status not covered by this implementation
    Hard,      // Restart(all) the hardware, the Charge Point is not required to gracefully stop ong
oing transaction.
    Soft       // Stop ongoing transactions gracefully and sending StopTransaction for every ongoing
 transaction.
) UDINT;
END_TYPE
```

5.2.1.34 E_OCPP1_TriggerMessageStatus

```
TYPE E_OCPP1_TriggerMessageStatus :
(
    None := 0,     // Any status not covered by this implementation
    Accepted,      // Requested notification will be sent.
    Rejected,      // Requested notification will not be sent.
    NotImplemented // Requested notification cannot be sent because it is either not implemented or
unknown.
) UDINT;
END_TYPE
```

5.2.1.35 E_OCPP1_Unit

```
TYPE E_OCPP1_Unit :
(
    None := 0,  // Any status not covered by this implementation
    Wh,         // watt-hours (energy).Default.
    kWh,        // kilo-watt-hours (energy).
    VARh,       // var-hours (reactive energy).
    kVARh,      // kilo-var-hours (reactive energy).
    W,          // watts (power).
    kW,         // kilo-watts (power).
    VA,         // volt-ampere (apparent power).
    kVA,        // kilo-volt-ampere (apparent power).
    VARs,       // vars (reactive power).
    kVAr,       // kilo-vars (reactive power).
    A,          // amperes (current).
    V,          // voltage (RMS AC).
    Celsius,    // degrees (temperature).
    Fahrenheit, // degrees (temperature).
    K,          // degrees kelvin (temperature).
    Percent     // percentage.
) UDINT;
END_TYPE
```

5.2.1.36 E_OCPP1_UnlockStatus

```
TYPE E_OCPP1_UnlockStatus :
(
    None := 0,    // Any status not covered by this implementation
    Unlocked,     // Connector has successfully been unlocked.
    UnlockFailed, // Failed to unlock the connector.
    NotSupported  // Charge Point has no connector lock, or ConnectorId is unknown.
) UDINT;
END_TYPE
```

5.2.1.37 E_OCPP1_UpdateStatus

```
TYPE E_OCPP1_UpdateStatus :
(
    None := 0,      // Any status not covered by this implementation
    Accepted,       // Local Authorization List successfully updated.
    Failed,         // Failed to update the Local Authorization List.
    NotSupported,   // Update of Local Authorization List is not supported by Charge Point.
    VersionMismatch // Version number in the request for a differential update is less or equal then
 version number of current list.
) UDINT;
END_TYPE
```

5.2.1.38 E_OCPP1_UpdateType

```
TYPE E_OCPP1_UpdateType :
(
    None := 0,    // Any status not covered by this implementation
    Differential, // Indicates that the current Local Authorization List must be updated with the va
lues in this message.
    Full          // Indicates that the current Local Authorization List must be replaced by the val
ues in this message.
) UDINT;
END_TYPE
```

5.2.1.39 E_OCPP1_ValueFormat

```
TYPE E_OCPP1_ValueFormat :
(
    None := 0, // Any status not covered by this implementation
    Raw,       // Data is to be interpreted as integer / decimal numeric data.
    SignedData // Data is represented as a signed binary data block, encoded as hex data.
) UDINT;
END_TYPE
```

5.2.1.40 E_OCPP_DebugLevel

```
TYPE E_OCPP_DebugLevel :
(
    None              := 0, // No Encryption
```

```
    MessageLogFile    := 1  // Write a log file of all messages to the boot directory.
) UDINT;
END_TYPE
```

5.2.1.41 E_OCPP_EncryptionMode

```
TYPE E_OCPP_EncryptionMode :
(
    None              := 0, // No Encryption
    Enable            := 1, // TLS enable, but without certificate validation (insecure)
    ServerCertificate := 2, // TLS enable, with server certificate validation
    ClientCertificate := 3  // TLS enable, with client certificate authentication
) UDINT;
END_TYPE
```

5.2.1.42 E_OCPP_EncryptionProtocol

```
TYPE E_OCPP_EncryptionProtocol :
(
    None    := 0,
    TLSv1p2 := 2, // TLS version 1.2
    TLSv1p3 := 3  // TLS version 1.3
) UDINT;
END_TYPE
```

5.2.2 Properties

5.2.2.1 ST_OCPP1_Client_BootInfo

The parameters sent in the Boot Notification are summarized in this structure.

Syntax

```
TYPE ST_OCPP_Client1_BootInfo :
STRUCT
    sChargePointModel        : STRING(23) := 'TcIotOcpp';
    sChargePointVendor       : STRING(23) := 'Beckhoff Automation';
    sChargeBoxSerialNumber   : STRING(23);
    sChargePointSerialNumber : STRING(23);
    sFirmwareVersion         : STRING(23);
    sMeterSerialNumber       : STRING(23);
    sMeterType               : STRING(23);
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional (OCPP)	Description
sChargePointModel	STRING(23)	No	Model of the Charge Point.
sChargePointVendor	STRING(23)	No	Vendor of the Charge Point.
sChargeBoxSerialNumber	STRING(23)	Yes	Serial number of the Charge Box within the Charge Point.
sChargePointSerialNumber	STRING(23)	Yes	Serial number of the Charge Point.
sFirmwareVersion	STRING(23)	Yes	Firmware version of the Charge Point.
sMeterSerialNumber	STRING(23)	Yes	Serial number of the main electricity meter of the Charge Point.
sMeterType	STRING(23)	Yes	Type of main electricity meter of the Charge Point.

5.2.2.2 ST_OCPP1_Client_Options

This structure can be used to switch optional functions in the OCPP client on and off.

Syntax

```
TYPE ST_OCPP_Client1_Options :
STRUCT
    bAllowOfflineTxForUnknownId : BOOL := FALSE;
    bAuthListEnabled            : BOOL := TRUE;
    bAuthCacheEnabled           : BOOL := TRUE;
    bLocalAuthOffline           : BOOL := TRUE;
    bLocalPreAuth               : BOOL := TRUE;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional (OCPP)	Description
bAllowOfflineTxForUnknownId	BOOL	No	If TRUE, Unknown Offline Authorization is enabled.
bAuthListEnabled	BOOL	No	If TRUE, the Local Authorization List is enabled.
bAuthCacheEnabled	BOOL	Yes	If TRUE, the Authorization Cache is enabled.
bLocalAuthOffline	BOOL	No	If TRUE, the Charge Points start a Transaction for locally authorized identifiers.
bLocalPreAuth	BOOL	No	If TRUE, the Charge Points start a Transaction for locally authorized identifiers without waiting for the Central System.

5.2.2.3 ST_OCPP1_Client_Param

This structure is used to set the parameters of the WebSockets connection of the OCPP client.

Syntax

```
TYPE ST_OCPP_Client1_Param :
STRUCT
    nOID            : OTCID := 0;
    nTaskOID        : OTCID := 0;
    bConnect        : BOOL := TRUE;
    sHost           : T_MaxString := '';
    nPort           : UINT := 443;
    sPath           : T_MaxString;
    sIdentity       : T_MaxString;
    eAuthMode       : E_OCPP1_AuthenticationMode;
    sAuthKey        : T_MaxString;
    eEncryptionMode : E_OCPP_EncryptionMode;
    eEncryptionProt : E_OCPP_EncryptionProtocol;
    sCaFile         : T_MaxString := '%TC_TARGETPATH%\Certificates\CA.crt';
    sCrtFile        : T_MaxString := '%TC_TARGETPATH%\Certificates\OCPP.crt';
    sKeyFile        : T_MaxString := '%TC_TARGETPATH%\Certificates\OCPP.key';
    eDebugLevel     : E_OCPP_DebugLevel := E_OCPP_DebugLevel.None;
    eTraceLevel     : TcTraceLevel := TcTraceLevel.tlWarning;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
nOID	OTCID	ObjectID of the TcIotOcppClient object. If this value is left at 0 (default), a new instance is created.
nTaskOID	OTCID	ObjectID of the task used. If this value is left at 0 (default), the I/O Idle Task is used.
bConnect	BOOL	If TRUE, the OCPP client automatically connects to the OCPP server. If FALSE, the connection must be established using the method Execute [▶ 22].
sHost	T_MaxString	Host name or IP address of the OCPP server.
nPort	UINT	Port of the OCPP server.
sPath	T_MaxString	Optionally specifies a URI of the OCPP server.
sIdentity	T_MaxString	Specifies the Identity of the Client.
eAuthMode	E_OCPP1_AuthenticationMode [▶ 190]	Optional Authentication Mode.
sAuthKey	T_MaxString	Optional Authentication Key.
eEncryptionMode	E_OCPP_EncryptionMode [▶ 198]	Optional Encryption Mode.
eEncryptionProt	E_OCPP_EncryptionProtocol [▶ 198]	Defines the Encryption Protocol.
sCaFile	T_MaxString	Certificate of the Certificate Authority (CA) as a file path (PEM or DER format).
sCrtFile	T_MaxString	Client certificate (Public Key) as file path (PEM or DER format).
sKeyFile	T_MaxString	Private Key of the client as a file path (PEM or DER format).
eDebugLevel	E_OCPP_DebugLevel [▶ 197]	The Debug Level (None or MessageLogFile). MessageLogFile ensures that a logfile with all OCPP messages is written to the TwinCAT boot directory.
eTraceLevel	TcTraceLevel	The maximum Trace Level from ADS logging.

5.2.2.4 ST_OCPP1_Client_Settings

This structure can be used to make further settings in the OCPP client.

Syntax

```
TYPE ST_OCPP_Client1_Settings :
STRUCT
    tConnectTimeout:                 : TIME := T#10S;
    tReconnectTimeout:               : TIME := T#10S;
    tBootTimeout:                    : TIME := T#5S;
    tMessageTimeout:                 : TIME := T#5S;
    nTransactionMessageAttempts      : UDINT := 5;
    tTransactionMessageRetryInterval : TIME := T#5S;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
tConnectTimeout	TIME	Specifies the time after which a connection attempt is aborted.
tReconnectTimeout	TIME	Specifies the time after a Disconnect before the client attempts to reconnect.
tBootTimeout	TIME	Specifies the time to wait for a response to the Boot Notification.
tMessageTimeout	TIME	Specifies how long to wait for a response to a message.
nTransactionMessageAttempts	UDINT	Specifies how often the Charge Point attempts to resend messages relating to transactions if the Central System has not processed them.
tTransactionMessageRetryInterval	TIME	Specifies the time in seconds after which another attempt to send is made.

5.2.2.5 ST_OCPP1_Server_Param

This structure is used to set the parameters of the WebSockets connection of the OCPP server.

Syntax

```
TYPE ST_OCPP1_ServerParam :
STRUCT
    nOID             : OTCID := 0;
    nTaskOID         : OTCID := 0;
    sHost            : T_MaxString := '0.0.0.0';
    nPort            : UINT := 443;
    sPath            : T_MaxString := 'ocpp';
    eAuthMode        : E_OCPP1_AuthenticationMode := E_OCPP1_AuthenticationMode.Basic;
    eEncryptionMode  : E_OCPP_EncryptionMode := E_OCPP_EncryptionMode.Enable;
    eEncryptionProt  : E_OCPP_EncryptionProtocol;
    sCaFile          : T_MaxString := '%TC_TARGETPATH%\Certificates\CA.crt';
    sCrtFile         : T_MaxString := '%TC_TARGETPATH%\Certificates\OCPP.crt';
    sKeyFile         : T_MaxString := '%TC_TARGETPATH%\Certificates\OCPP.key';
    eDebugLevel      : E_OCPP_DebugLevel := E_OCPP_DebugLevel.None;
    eTraceLevel      : TcTraceLevel := TcTraceLevel.tlWarning;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
nOID	OTCID	ObjectID of the TcIotOcppServer object. If this value is left at 0 (default), a new instance is created.
nTaskOID	OTCID	ObjectID of the task used. If this value is left at 0 (default), the I/O Idle Task is used.
sHost	T_MaxString	Defines the server address as a host name, domain name or IP address.
nPort	UINT	Defines the server port.
sPath	T_MaxString	Defines the server path.
eAuthMode	E_OCPP1_AuthenticationMode [▶ 190]	Optional Authentication Mode.
eEncryptionMode	E_OCPP_EncryptionMode [▶ 198]	Optional Encryption Mode.
eEncryptionProt	E_OCPP_EncryptionProtocol [▶ 198]	Defines the Encryption Protocol.
sCaFile	T_MaxString	Certificate of the Certificate Authority (CA) as a file path (PEM or DER format).
sCrtFile	T_MaxString	Server certificate (Public Key) as file path (PEM or DER format).
sKeyFile	T_MaxString	Private Key of the server as a file path (PEM or DER format).
eDebugLevel	E_OCPP_DebugLevel [▶ 197]	The Debug Level (None or MessageLogFile). MessageLogFile ensures that a logfile with all OCPP messages is written to the TwinCAT boot directory.
eTraceLevel	TcTraceLevel	The maximum Trace Level from ADS logging.

5.2.3 Types

5.2.3.1 ST_OCPP1_AuthorizationData

Syntax

```
TYPE ST_OCPP1_AuthorizationData :
STRUCT
    sIdTag      : T_OCPP1_IdToken;
    stIdTagInfo : ST_OCPP1_IdTagInfo;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
sIdTag	T_OCPP1_IdToken [▶ 209]	No	ID token to which this authorization information belongs.
stIdTagInfo	ST_OCPP1_IdTagInfo [▶ 207]	Yes	Contains information on Authorization Status, expiration date and parent ID.

5.2.3.2 ST_OCPP1_ChargingProfile

Syntax

```
TYPE ST_OCPP1_ChargingProfile :
STRUCT
    nProfileId      : UDINT;
    nTransactionId  : UDINT;
    nStackLevel     : UDINT;
    eProfilePurpose : E_OCPP1_ChargingProfilePurposeType;
    eProfileKind    : E_OCPP1_ChargingProfileKindType;
    eRecurrencyKind : E_OCPP1_RecurrencyKindType;
    nValidFrom      : ULINT;
    nValidTo        : ULINT;
    mSchedule       : ST_OCPP1_ChargingSchedule;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
nProfileId	UDINT	No	Unique identifier for this Charging Profile.
nTransactionId	UDINT	Yes	Only valid if eProfilePurpose is set to TxProfile. The Transaction ID can be used to assign the profile to a specific transaction.
nStackLevel	UDINT	No	Value that defines the priority of a Charging Profile. The value 0 defines the lowest priority.
eProfilePurpose	E_OCPP1_ChargingProfilePurposeType [▶ 191]	No	Purpose of the transmitted Charging Profile.
eProfileChild	E_OCPP1_ChargingProfileKindType [▶ 191]	No	Determines the Charging Profile type.
eRecurrencyChild	E_OCPP1_RecurrencyKindType [▶ 195]	Yes	Specifies the starting point of a repetition.
nValidFrom	ULINT	Yes	Time from which the Charging Profile is valid. If no value is specified, the Charging Profile is valid as soon as it reaches the Charge Point.
nValidTo	ULINT	Yes	Time from which the Charging Profile is no longer valid. If no value is specified, the Charging Profile is valid until the Charge Point receives a new Charging Profile.
mSchedule	ST_OCPP1_ChargingSchedule [▶ 205]	No	Contains the Charging Schedule and thus the limit values for the available power or current over time.

5.2.3.3 ST_OCPP1_ChargingProfileMax

Syntax

```
TYPE ST_OCPP1_ChargingProfileMax :
STRUCT
    nProfileId      : UDINT;
    nTransactionId  : UDINT;
    nStackLevel     : UDINT;
    eProfilePurpose : E_OCPP1_ChargingProfilePurposeType;
    eProfileKind    : E_OCPP1_ChargingProfileKindType;
    eRecurrencyKind : E_OCPP1_RecurrencyKindType;
    nValidFrom      : ULINT;
    nValidTo        : ULINT;
    mSchedule       : ST_OCPP1_ChargingScheduleMax;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
nProfileId	UDINT	No	Unique identifier for this Charging Profile.
nTransactionId	UDINT	Yes	Only valid if eProfilePurpose is set to TxProfile. The Transaction ID can be used to assign the profile to a specific transaction.
nStackLevel	UDINT	No	Value that defines the priority of a Charging Profile. The value 0 defines the lowest priority.
eProfilePurpose	E_OCPP1_ChargingProfilePurposeType [▶ 191]	No	Purpose of the transmitted Charging Profile.
eProfileChild	E_OCPP1_ChargingProfileKindType [▶ 191]	No	Determines the Charging Profile type.
eRecurrencyChild	E_OCPP1_RecurrencyKindType [▶ 195]	Yes	Specifies the starting point of a repetition.
nValidFrom	ULINT	Yes	Time from which the Charging Profile is valid. If no value is specified, the Charging Profile is valid as soon as it reaches the Charge Point.
nValidTo	ULINT	Yes	Time from which the Charging Profile is no longer valid. If no value is specified, the Charging Profile is valid until the Charge Point receives a new Charging Profile.
mSchedule	ST_OCPP1_ChargingScheduleMax [▶ 205]	No	Contains the Charging Schedule and thus the limit values for the available power or current over time.

5.2.3.4 ST_OCPP1_ChargingSchedule

Syntax

```
TYPE ST_OCPP1_ChargingSchedule :
STRUCT
    nStart       : ULINT;
    nDuration    : UDINT;
    eRateUnit    : E_OCPP1_ChargingRateUnitType;
    fRateMin     : REAL;
    nPeriodCount : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
nStart	ULINT	Yes	Starting point of an absolute Charging Schedule. If the value is not specified, the schedule is relative to the start of the charging process.
nDuration	UDINT	Yes	Duration of the Charging Schedule in seconds. If the value is not specified, the last Charging Schedule Period is continued indefinitely or until the end of the transaction.
eRateUnit	E_OCPP1_ChargingRateUnitType [▶ 192]	No	The unit of the limit, expected in watts (power) or amperes (current).
fRateMin	REAL	Yes	Minimum charging rate supported by the electric vehicle. The unit is defined by eRateUnit.
nPeriodCount	UDINT	No	Number of subsequent Charging Schedule Periods.

5.2.3.5 ST_OCPP1_ChargingScheduleMax

Syntax

```
TYPE ST_OCPP1_ChargingScheduleMax :
STRUCT
    nStart       : ULINT;
    nDuration    : UDINT;
    eRateUnit    : E_OCPP1_ChargingRateUnitType;
    fRateMin     : REAL;
    nPeriodCount : UDINT;
    arrPeriods   : ARRAY[1..*] OF ST_OCPP1_ChargingSchedulePeriod;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
nStart	ULINT	Yes	Starting point of an absolute Charging Schedule. If the value is not specified, the schedule is relative to the start of the charging process.
nDuration	UDINT	Yes	Duration of the Charging Schedule in seconds. If the value is not specified, the last Charging Schedule Period is continued indefinitely or until the end of the transaction.
eRateUnit	E_OCPP1_ChargingRateUnitType [▶ 192]	No	The unit of the limit, expected in watts (power) or amperes (current).
fRateMin	REAL	Yes	Minimum charging rate supported by the electric vehicle. The unit is defined by eRateUnit.
nPeriodCount	UDINT	No	Number of subsequent Charging Schedule Periods.
arrPeriods	ARRAY[1..Param_OCPP.nSchedulePeriodsMax] OF ST_OCPP1_ChargingSchedulePeriod [▶ 206]	No	List of Charging Schedule Period elements.

5.2.3.6 ST_OCPP1_ChargingSchedulePeriod

Syntax

```
TYPE ST_OCPP1_ChargingSchedulePeriod :
STRUCT
    nStart     : UDINT;
    fLimit     : REAL;
    nPhases    : UDINT := 0;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
nStart	UDINT	No	Start of the Charging Schedule Period, in seconds from the start of the Charging Schedule.
fLimit	REAL	No	Charging Rate Limit during the respective Charging Schedule Period.
nPhases	UDINT	Yes	The number of phases that can be used for charging.

5.2.3.7 ST_OCPP1_ConfigKeyValue

Syntax

```
TYPE ST_OCPP1_ConfigKeyValue :
STRUCT
    sKey         : T_OCPP1_ConfigKey;
    sValue       : T_OCPP1_ConfigValue;
    bReadonly    : BOOL;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
sKey	T_OCPP1_ConfigKey [▶ 208]	No	Configuration Key to be overwritten or added.
sValue	T_OCPP1_ConfigValue [▶ 208]	Yes	Value of the associated Configuration Key. Can be omitted if the Configuration Key is known and should not be reset.
bReadonly	BOOL	No	Determines the access right of the Central System, either the value may only be read (TRUE) or also written (FALSE).

5.2.3.8 ST_OCPP1_IdTagInfo

Syntax

```
TYPE ST_OCPP1_IdTagInfo :
STRUCT
    eStatus      : E_OCPP1_AuthorizationStatus;
    nExpiryDate  : ULINT;
    sParentIdTag : T_OCPP1_IdToken;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
eStatus	E_OCPP1_AuthorizationStatus [▶ 190]	No	Contains information on whether the ID tag has been accepted by the Central System.
nExpiryDate	ULINT	Yes	Optionally contains the date on which the ID tag is to be removed from the Authorization Cache.
sParentIdTag	T_OCPP1_IdToken [▶ 209]	Yes	Optionally contains the parent ID tag

5.2.3.9 ST_OCPP1_SampledValue

From the point of view of the OCPP specification, a Meter Value is a Sampled Value with a timestamp. In this PLC library, the timestamp is also managed in the Sampled Value in order to have one structure level less.

Syntax

```
TYPE ST_OCPP1_SampledValue :
STRUCT
    fValue     : LREAL;
    nTimestamp : ULINT := 0;
    eContext   : E_OCPP1_ReadingContext := E_OCPP_ReadingContext.SamplePeriodic;
    eFormat    : E_OCPP1_ValueFormat := E_OCPP_ValueFormat.Raw;
    eMeasurand : E_OCPP1_Measurand;
    ePhase     : E_OCPP1_Phase;
    eLocation  : E_OCPP1_Location;
    eUnit      : E_OCPP1_Unit;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Optional	Description
fValue	LREAL	No	Sampled Value as a decimal number ("Raw"). "SignedData" is not supported.
nTimestamp	ULINT	Yes	Timestamp of the Sampled Value.
eContext	E_OCPP1_ReadingContext [▶ 195]	Yes	Context in which the Sampled Value was recorded (start, end or sample).
eFormat	E_OCPP1_ValueFormat [▶ 197]	Yes	In this implementation, it is always the "Raw" format.
eMeasurand	E_OCPP1_Measurand [▶ 194]	Yes	Type of the Sampled Value.
ePhase	E_OCPP1_Phase [▶ 195]	Yes	Information on the interpretation of the Sampled Value.
eLocation	E_OCPP1_Location [▶ 194]	Yes	Location of the Sampled Value.
eUnit	E_OCPP1_Unit [▶ 197]	Yes	Unit of the Sampled Value.

5.2.3.10 T_OCPP1_ConfigKey

Syntax

```
TYPE T_OCPP1_ConfigKey : STRING(63); END_TYPE
```

5.2.3.11 T_OCPP1_ConfigValue

Syntax

```
TYPE T_OCPP1_ConfigValue : STRING(183); END_TYPE
```

5.2.3.12 T_OCPP1_DataTransferData

Syntax

```
TYPE T_OCPP1_DataTransferData : STRING(Param_OCPP.nDataTransferMax); END_TYPE
```

5.2.3.13 T_OCPP1_IdToken

Syntax

```
TYPE T_OCPP1_IdToken : STRING(23); END_TYPE
```

5.2.3.14 T_OCPP_Hash

Syntax

```
TYPE T_OCPP_Hash : STRING(95); END_TYPE
```

5.2.3.15 T_OCPP_Identity

Syntax

```
TYPE T_OCPP_Identity : STRING(31); END_TYPE
```

5.2.3.16 T_OCPP_MessageId

Syntax

```
TYPE T_OCPP_MessageId : STRING(63); END_TYPE
```

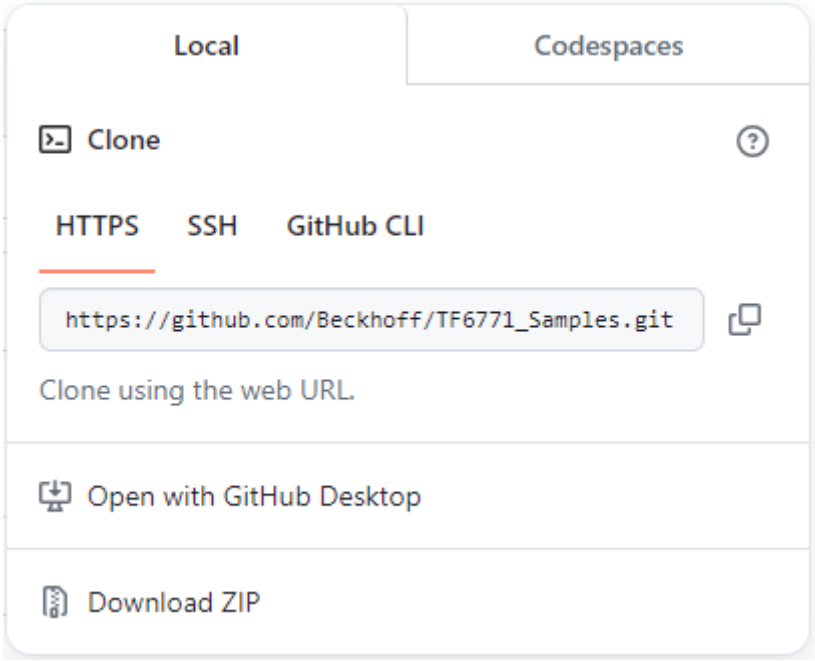
5.3 GVLs

5.3.1 Param_OCPP

Name	Type	Default value	Description
nConnectorCount	UDINT	2	Maximum number of Connectors per Charge Point.
nSampledValuesMax	UDINT	64	Maximum number of SampledValues in a MeterValue.
nChargingProfileMax	UDINT	10	Maximum number of Charging Profiles.
nSchedulePeriodsMax	UDINT	24	Maximum number of Charging Schedule Periods in a Charging Schedule.
nConfigKeysMax	UDINT	100	Maximum number of Configuration Keys in a GetConfiguration call.
nAuthListMax	UDINT	10	Maximum number of entries in the Local Authorization List.
fChargingLimitMax	REAL	40	Standard charging power limit for schedules.
nDataTransferMax	UDINT	1023	Maximum number of bytes in the payload for the SendDataTransfer method.

6 Samples

Sample code and configurations for this product can be obtained from the corresponding repository on GitHub: https://github.com/Beckhoff/TF6771_Samples. There you have the option to clone the repository or download a ZIP file containing the sample.



The following table also provides an overview of the samples currently available in the documentation:

Sample	Short description
TF6771_IotOcppGettingStarted	This sample provides a simple Getting Started and is described in more detail in the chapter Quick Start [▶ 15].
TF6771_IotOcppClient1	This sample shows the basic use of the OCPP client.
TF6771_IotOcppServer1	This sample shows the basic use of the OCPP server.

7 Appendix

7.1 Hash calculation

The Hash consists of the identity (username) and password. These are separated by a colon.

%username%:%password%

The SHA256 algorithm is used for Hash calculation.

Example:

Identity (username): BeckhoffOCPP

Password: 1234abc$

Calculated Hash: 30206a1e174ac04ea140234f2c1edd854d8dbca1d664b92d763cb5ef78c8e8dc

Any SHA256 calculator can be used at this point. These can be found online or as functions in various libraries, for example.

7.2 Configuration Keys

Here you will find all Configuration Keys that are initialized with the method InitStandard [▶ 186].

Name	Profile	Value	Description
GetConfigurationMaxKeys	Core (9.1)	Param_OCPP [▶ 209].nConfigKeysMax	Maximum number of Configuration Keys in a GetConfiguration call.
NumberOfConnectors	Core (9.1)	Param_OCPP [▶ 209].nConnectorCount	Maximum number of Connectors per Charge Point.
SupportedFeatureProfiles	Core (9.1)	Core, Reservation, SmartCharging, RemoteTrigger	List of all supported Feature Profiles.
LocalAuthListMaxLength	Local Auth List Management (9.2)	Param_OCPP [▶ 209].nAuthListMax	Maximum number of entries in the Local Authorization List.
SendLocalListMaxLength	Local Auth List Management (9.2)	Param_OCPP [▶ 209].nAuthListMax	Maximum number of entries that can be sent to the Charge Point in a SendLocalList [▶ 120].
ChargingScheduleAllowedChargingRateUnit	Smart Charging (9.4)	Current	List of supported quantities that may be used in a ChargingSchedule. Permitted values are "Current" and "Power".
ChargingScheduleMaxPeriods	Smart Charging (9.4)	Param_OCPP [▶ 209].nSchedulePeriodsMax	Maximum number of Charging Schedule Periods in a Charging Schedule.
MaxChargingProfilesInstalled	Smart Charging (9.4)	Param_OCPP [▶ 209].nChargingProfileMax	Maximum number of Charging Profiles that can be installed in parallel.

7.3 Troubleshooting

Behavior	Explanation
The Heartbeat interval in the client has been changed. However, a different value is set once the connection to the server has been established.	In the BootNotification.conf, the OCPP server sends a specified Heartbeat interval for the client. This is also set internally in the client. However, it is then possible to change the internal HeartbeatInterval during runtime.

More Information:
www.beckhoff.com/tf6771

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com