|  |
| --- |
| عنوان پروژه: Connected Cars Proposal |
| کد پروژه: P.EVR.28  تدوین: Connected Services |
| تاریخ ارسال گزارش: 07/02/1401 |
|  |

در پروژه روال شارژ، سرور واسطه بین اپلیکیشن و سرور ocpp است. بنابراین 4 دسته message passing وجود دارد:

* دریافت پیام از اپلیکیشن
* ارسال پیام به اپلیکیشن
* دریافت پیام از ocpp
* ارسال پیام به ocpp

پیام هایی که میان سرور و ocpp رد و بدل میشوند از طریق schema هایی که در پوشه schema تعریف شده اند درستی سنجی میشوند. هر صورتی که این درستی سنجی pass نشود، پیغام خطا به اپلیکیشن ارسال شده و روال شارژ قطع میشود.

**پیام هایی که اپلیکیشن به سرور ارسال میکند:**

در ابتدای کانکشن، شناسه ی شارژر یعنی cpID در انتهای url برای سرور ارسال میشود در طول message passing ها مورد استفاده قرار میگیرد. پس از دریافت cpID از url، لیستی از کانکتورهای available در آن شارژر به اپلیکیشن در قالب پیام connectors ارسال میشود تا کاربر انتخاب کند که با چه connector ای میخواهد شارژ انجام دهد. پیام ها به ترتیب ارسال از سمت اپلیکیشن برای آغاز یک روال شارژ است.

* **Userconnecting**: طی این پیام احراز هویت کاربر (از روی توکن ارسالی) تایید میشود، اطلاعات اولیه روال شارژ در اسکیمای profile به منظور تشخیص ادامه روال شارژ درصورت قطع اتصال اپلیکیشن ذخیره میشود (فیلد chargeInfo)، وضعیت فعال یا غیرفعال بودن شارژ بررسی میشود و یکسری اطلاعات در دیکشنری ای با نام clients با key=cpID ذخیره میشود تا درصورت وجود چندین روال شارژ همزمان، برای message passing میان سرور و ocpp مورد استفاده قرار گیرد. درصورت احراز هویت کاربر، به اپلیکیشن پیام authenticated با اطلاعات شارژر ارسال میشود. درصورتی که کاربر احراز هویت نشود یا شارژر در دسترسی نباشد، به اپلیکیشن پیام unauthorized ارسال میشود، و در صورت catch(err) پیام problemOccured ارسال میشود.

درصورتی که شارژر مشغول باشد و روال شارژر نیز مربوط به همین اپلیکیشن مجدد وصل شده نباشد، به اپلیکیشن اعلام میشود که شارژر درحال ارائه سرویس میباشد.

درصورت بروز هرگونه خطا، سوکت اپلیکیشن بسنه میشود و chargeInfo از دیتابیس خالی میشود.

* **Userapproved**: اپلیکیشن اعلاممیکندکه متوجه احراز هویت کاربر شده و میتواند روال شارژ را آغاز کند. بنابراین در این مرحله سرور بررسی میکند که آیا کاربر با rfid میخواهد شارژ کند یا phone. کاربر میتواند به تعداد rfid هایش میتواند شارژ همزمان داشته باشد، ولی اگر هیچ rfid ای نداشته باشد، فقط میتواند یک شارژ در لحظه داشته باشد. درصورتی که ظرفیت روال های شارژ همزمان کاربر پر شده باشد، به کاربر خطا ارسال میشود. همچنین درصورتیکه کاربر currency نداشته باشد، خطا برایش ارسال میشود. درصورتیکه همه چیز بدون مشکل باشد، پیام approved با اپلیکیشن ارسال میشود و سپس پیام RemoteStartTransaction به ocpp ارسال میشود تا مقدمات آغاز روال شارژ فراهم شود.
* **Userstop**: زمانی که کاربر از داخل اپلیکیشن بخواهد روال شارژ را خاتمه دهد، این پیام از اپلیکیشن به سرور ارسال میشود. سپس سرور پیام RemoteStopTransaction را به ocpp ارسال میکند تا درخواست خاتمه شارژ را اعلام کند.
* **Error**: اگر به هردلیلی این پیام از سمت اپلیکیشن بهسرور ارسال شود،سرور با ارسال پیام RemoteStopTransaction به ocpp ، درخواست خاتمه روال شارژ را میدهد.
* **Pingg**: هرزمانی که اپلیکیشن بخواهد کانکسن خود با سرور را بررسی کند، این پیام را ارسال میکند و در پاسخ، سرور پیام pong را به اپلیکیشن ارسال میکند.
* **getAllChargers**: طی این پیام اپلیکیشن درخواست میدهد که لیست همه شارژها با دسته بندی private و public را دریافت نماید. پاسخ در قالب پیام allChargers سمت اپلیکیشن ارسال میشود.

**پیام هایی که از ocpp دریافت میشوند:**

* **Tarrif**: پس از اینکه ocpp پیام RemoteStartTransaction را از سرور دریافت میکند، پیام tarrif را برای تعیین تعرفه های برق و ... ارسال میکند. این تعریفها برای محاسبه هزینه شارژ استفاده میشوند.
* **RemoteStartTransaction**: بلافاصله پس از پیام tarrif ، این پیام را ocpp ارسال میکند و فیلید ارسال میکند به نام status که یا accepted است یا Rejected. درصورت accepted بودن، ocpp اعلام امادگی میکند برای آغاز روال شارژ. درنتیجه سرور پیام startChargingRes را به اپلیکیشن ارسال میکند تا اپلیکیشن مطلع شود که روال شارژر درحال شروع شدن است. در صورت rejected بودن، سرور پیام problemOccured را سمت اپلیکیشن ارسال میکند و سوکت اپلیکیشن را Close میکند و chrgeInfo از دیتابیس پاک میشود، همچنین Clients[cpID] نیز delete میشود.
* **StartTransaction**: ocpp با ارسال این پیام اعلام میکند که روال شارژ آغاز شده است. از این پس پیام chargeData برای اپلیکیشن ارسال میشود تا اپلیکیشن از اطلاعات لحظه ای شارژ نظیر اینکه چند درصد شارژ انجام شده است و ... آگاه شود. در پیام StartTransaction فعلا مقادیر پیام chargeData صفر ارسال میشود برای اپلیکیشن.

همچنین اگر روال شارژ بر اساس یک schedule آغاز شده باشد، در پی این پیام status برای آن schedule مقدار charging میگیرد.

* **MeterValues**: در طی روال شارژ، ocpp فیلدهای مختلف شارژ نظیر درصد شارژ و ... را سمت سرور ارسال میکند، و سرور این پارامترها را در قالب پیام chargeData برای اپلیکیشن ارسال میکند. پس هربار که پیام meterValue دریافت شود، در لحظه پیام chargeData برای اپلیکیشن ارسال میشود.
* **RemoteStopTransaction**: ocpp این پیامرا به این منظور ارسال میکند که اعلام کند روال شارژ قرار است خاتمه یابد.
* **StopTransaction**: سپس ocpp شارژ را خاتمه میدهد و این پیام را به سرور ارسال میکند. در نتیجه این پیام، پیامstopCharging به اپلیکیشن ارسال میشود و کار اپلیکیشن خاتمه میابد.
* **Payment**: این پیام پس از خاتمه روال شارژ به سرور ارسال میشود و در نتیجه دریسافت آن، لاگ شارژ در دیتابیس ذخیره میشود.
* **Errors**: اگر به هر دلیلی خطایی سمت ocpp رخ دهد، این پیام سمت سرور ارسال شده و سرور نیز پیام error را به اپلیکیشن ارسال کرده و سوکت اپلیکیشن را میبندد.