1 Currently implemented energy management mechanisms

1.1 Overview of the energy management architecture

The charge controller features a software architecture that allows the co-existence of multiple energy management mechanisms. Each of these mechanisms is registered in a central registry.

Each mechanism can at any point in time provide a maximum current setting that is permissible to offer to the connected vehicle. Also each mechanism can require for the charging to be paused. At any point in time each mechanism can inform the registry about a change forcing a full update of the charging current that is signaled to the vehicle.

The central registry upon start of a charging transaction queries all mechanisms for the maximum current. If any mechanism requires pausing of the transaction, the switching on of the contactor is delayed. The transaction however does start and charging time is counted.

If all mechanisms agree that charging is permissible, the registry computes the minimum of all the signalled current settings and provides this setting to the mode 3 charging logic.

If at any point in time any mechanism identifies a change of its strategy (e.g. due to the change of some input or just based on time being elapsed) it triggers a reevaluation. The registry again performs the same algorithm, queries all mechanisms and provides the end result to the mode 3 charging logic.

The registry itself therefore operates on a real-time basis and does not keep a charging plan for the future. If any mechanism requires a plan for the future, e.g. a charging schedule based on time, it needs to keep this state internally.

1.2 Backend based mechanisms

1.2.1 Real-time setting via Backend

The charge controller maintains an OCPP configuration parameter named OperatorCurrentLimit. This parameter can be changed at any time while the charge controller is connected to a backend system. The parameter contains a single value expressing the maximum charging current that is permissible. A change is possible locally using the configuration interface or remotely using a OCPP ChangeConfiguration message. A change takes effect immediately and triggers a reevaluation of the charging current by the central registry.

Using this parameter and given a sufficiently low latency and high reliability backend connection any energy management algorithm can be implemented on the backend side and not inside the controller.

This feature is an example for the most simple energy management mechanism within the architecture described above as it always reports a fixed value and only reports a change when a message arrives from the backend.

1.2.2 OCPP 1.6 Smart Charging

The ebee charge controller has implemented the OCPP 1.6 smart charging profile that is used to communicate schedules from the backend to the charge controller. All types of schedules including transaction schedules are reported. Similarly to the real time mechanism above it provides a means for fine grained control to the backend system. The actual computation of sensible schedules however is done on the backend side.
The backend side can use meter values reported by the charge controller or even by other charge controllers as well as arbitrary other input, e.g from a grid operator, another colocated device or even the weather report as input for deriving schedules.

The mechanism on the charge controller is the simply executing those schedules by reporting the correct current to the mode 3 charging logic at the right point in time and by triggering reevaluations at every change in its schedule.

1.3 Local mechanisms

1.3.1 Peer group

The peer group mechanism allows pairing a specific number (currently up to 4) charge controllers into a peer group. This peer group is then assigned a joint maximum current that must not be exceeded.

After starting a charging transaction and before offering a specific current to a vehicle, the mechanism queries all group members for the charging current they are signalling at this point in time. The controller then starts offering the remaining left over current to the vehicle, however not more than its configured maximum current.
At the same time the controller starts signalling the current it desires to charge which is its built in hardware maximum.
While charging each controller cyclically queries the other peer members for their status and learns their desired current and also the current they are currently charging with.
Each controller seeks to offer its fair share of current which is the relationship of the total current available to the group and its share of the total desired current.

In result two controllers that have the same maximum current setting, will each charge with half of the available current or their maximum.

If two controllers where one has twice the amount of maximum current (e.g. 32A) than the other controller (e.g. 16A) will charge with 2/3 of the available current while the other will charge with 1/3.

The peer group mechanism is designed to protect a fuse from tripping at all costs and will therefore not charge if a peer member is not available for communication as it is unclear how much of the available current is currently used by that member.

The peer group mechanism will charge with the amount of current that is available even if all other peer members are offering their hardware maximum if communication was possible with all peer members in the past.

Based on a configuration parameter it is possible to make the algorithm more risk taking but also more reliable in case of communication loss. Based on this parameter the algorithm

assumes that a controller that cannot be reached for communication does not charge at all. In this mode the algorithm is taking a minor risk to trip a fuse it is designed to protect as it is possible that a set of peers is partitioned into two communicating sets of peers which then each assume they can offer the maximum available current.

The peer group mechanism currently does not take into account

- the actual current drawn by the vehicle,
- the phases used by the vehicle
- the possibility to sequence vehicles with higher current as compared to offering less current in parallel

1.3.2 meter monitoring

The meter monitoring algorithm serves the purpose to protect a fuse that is colocated with a meter that is connected to the charge controller. The most typical setting is a S0 or pulse based meter that is connected to the charge controller in addition to the built in eHZ or modbus meter. However, the reverse scenario is also possible.

The controller will cyclically monitor the power that the meter signals, either by reading the power directly from the meter or by computing the power based on subsequent energy (kwh) values. In case of S0 meters the power is computed based on the time between two subsequent ticks.

If the meter signals a power that is above a certain set threshold and the charge controller is currently charging the charge controller will decrease the current that is offered to the connected vehicle sufficiently to decrease the total power consumption below the threshold. If the reduction is not sufficient the reduction is increased.

The algorithm assumes that a connected car is always consuming the maximum amount of current that it is signalled.

1.3.3 Dry contact based

The dry contact based algorithm allows reducing the current signalled to a vehicle to a fixed value based on an opto-coupler input to the controller.

Like the OperatorCurrentLimit real time backend algorithm, this provides a simple real time means for local current reduction. The switching of the opto coupler can be triggered by an external load that is drawing current and requires the reduction of the charging current, such as a co-located street light to a street light charger or an air conditioning system colocated to a residential charger.

The algorithm reduces the signalled current to a preconfigured value when the input goes high and increases it again, when the input goes low.