Electric Era Charging Network Uptime Report Display Program Documentation Set

Table of Contents

Electric Era Charging Network
Uptime Report Display Program

Electric Era Charging Network Uptime Report Display Program

Table of Contents

Chapter 1. Electric Era Coding Challenge Specification

Hasitha Dharmasiri

1. Overview

1. This is a simple coding challenge to test your abilities.
2. To join the software program at Electric Era, you must complete this challenge.

2. Challenge

1. You must write a program that calculates uptime for stations in a charging network.
2. It will take in a formatted input file that indicates individual charger uptime status for a given time period and write output to standard-output (stdout).
3. Station Uptime is defined as the percentage of time that any charger at a station was available, out of the entire time period that any charger at that station was reporting in.

2.1. Input File Format

1. The input file will be a simple ASCII text file.
2. The first section will be a list of station IDs that indicate the Charger IDs present at each station.
3. The second section will be a report of each Charger ID's availability reports.
4. An availability report will contain the Charger ID, the start time, the end time, and if the charger was "up" (i.e. available) or not.
5. The following preconditons will apply:
6. Station ID will be guaranteed to be a unsigned 32-bit integer and guaranteed to be unique to any other Station ID.
7. Charger ID will be guaranteed to be a unsigned 32-bit integer and guaranteed to be unique across all Station IDs.
8. start time nanos and end time nanos are guaranteed to fit within a unsigned 64-bit integer.
9. up will always be true or false
10. Each Charger ID may have multiple availability report entries.
11. Report entries need not be contiguous in time for a given Charger ID.
12. A gap in time in a given Charger ID's availability report should count as downtime.
13.

```
a.  [Stations]
```

b. <Station ID 1> <Charger ID 1> <Charger ID 2> ... <Charger ID n>

c. ...

d. <Station ID n> ...

e.

f. [Charger Availability Reports]

g. <Charger ID 1> <start time nanos> <end time nanos> <up (true/false)>

h. <Charger ID 1> <start time nanos> <end time nanos> <up (true/false)>

i. ...

j. <Charger ID 2> <start time nanos> <end time nanos> <up (true/false)>

k. <Charger ID 2> <start time nanos> <end time nanos> <up (true/false)>

l. ...

m. <Charger ID n> <start time nanos> <end time nanos> <up (true/false)>

2.2. Program Parameters and Runtime Conditions

1. Your program will be executed in a Linux environment running on an amd64 architecture.
2. If your chosen language of submission is compiled, ensure it compiles in that environment.
3. Please avoid use of non-standard dependencies.
4. The program should accept a single argument, the path to the input file. The input file may not necessarily be co-located in the same folder as the program.
5. Example CLI execution:
6. ./your_submission relative/path/to/input/file

2.3. Output Format

1. The output shall be written to stdout. If the input is invalid, please simply print ERROR and exit.
2. stderr may contain detailed error information but is not mandatory.
3. If there is no error, please write stdout as follows, and then exit gracefully.
4. <Station ID 1> <Station ID 1 uptime>
5. <Station ID 2> <Station ID 2 uptime>
6. ...
7. <Station ID n> <Station ID n uptime>
8. Station ID n uptime should be an integer in the range [0-100] representing the given station's uptime percentage. The value should be rounded down to the nearest percent.

9. Please output Station IDs in *ascending order*.

3. Testing and Submission

1. This repository contains a few example input files, along with the expected stdout output (this expected stdout is encoded in a separate paired file).
2. Please submit the following in a zip file to coding-challenge-submissions@electricera.tech for consideration:

4. Considerations

1. All aspects of your solution will be considered. Be mindful of:
2. Correctness for both normal and edge cases
3. Error-handling for improper inputs or unmet preconditions
4. Maintainability and readability of your solution
5. Scalability of the solution with increasingly large datasets

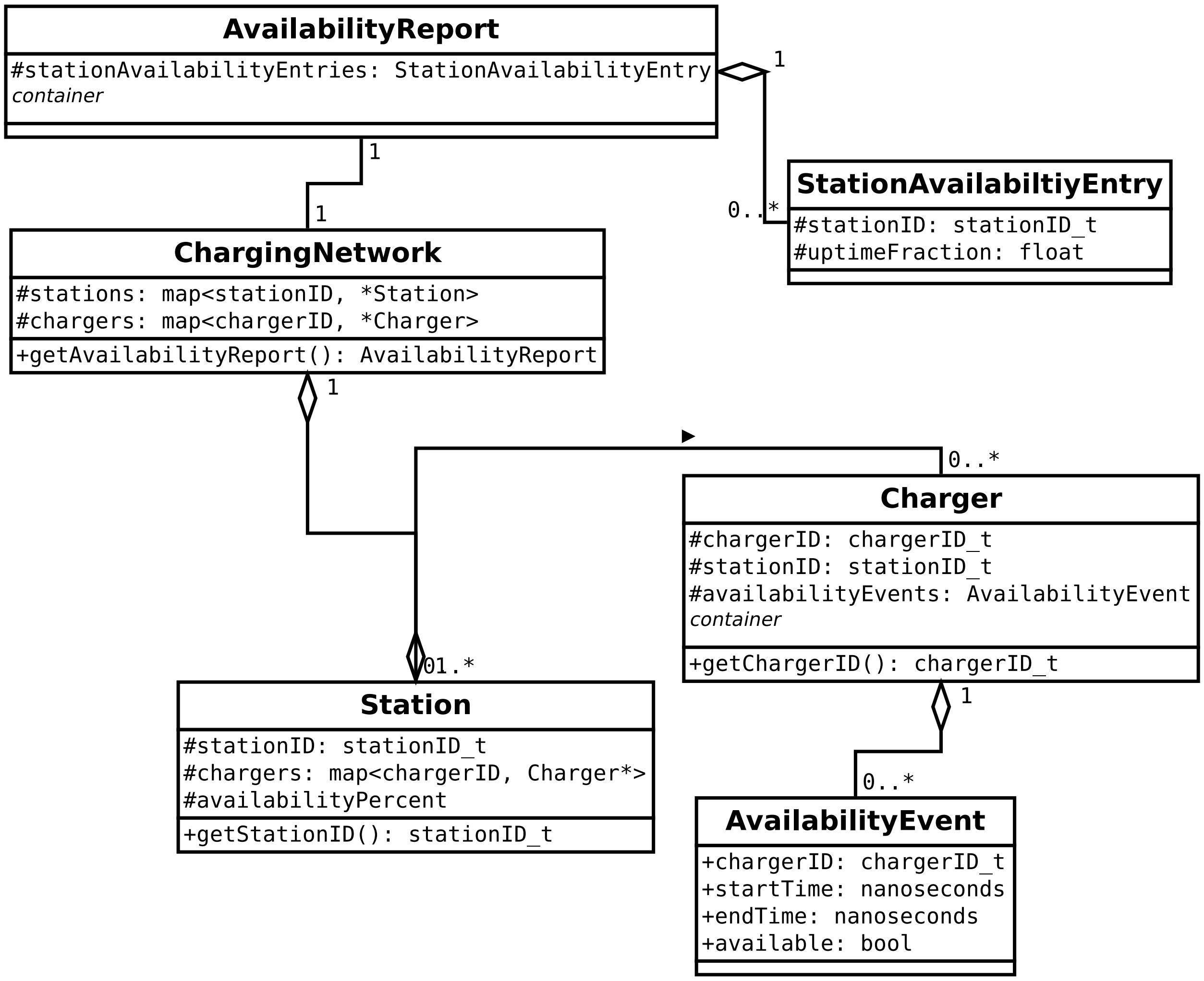
Chapter 2. Using the Program

Jaspreet Dha

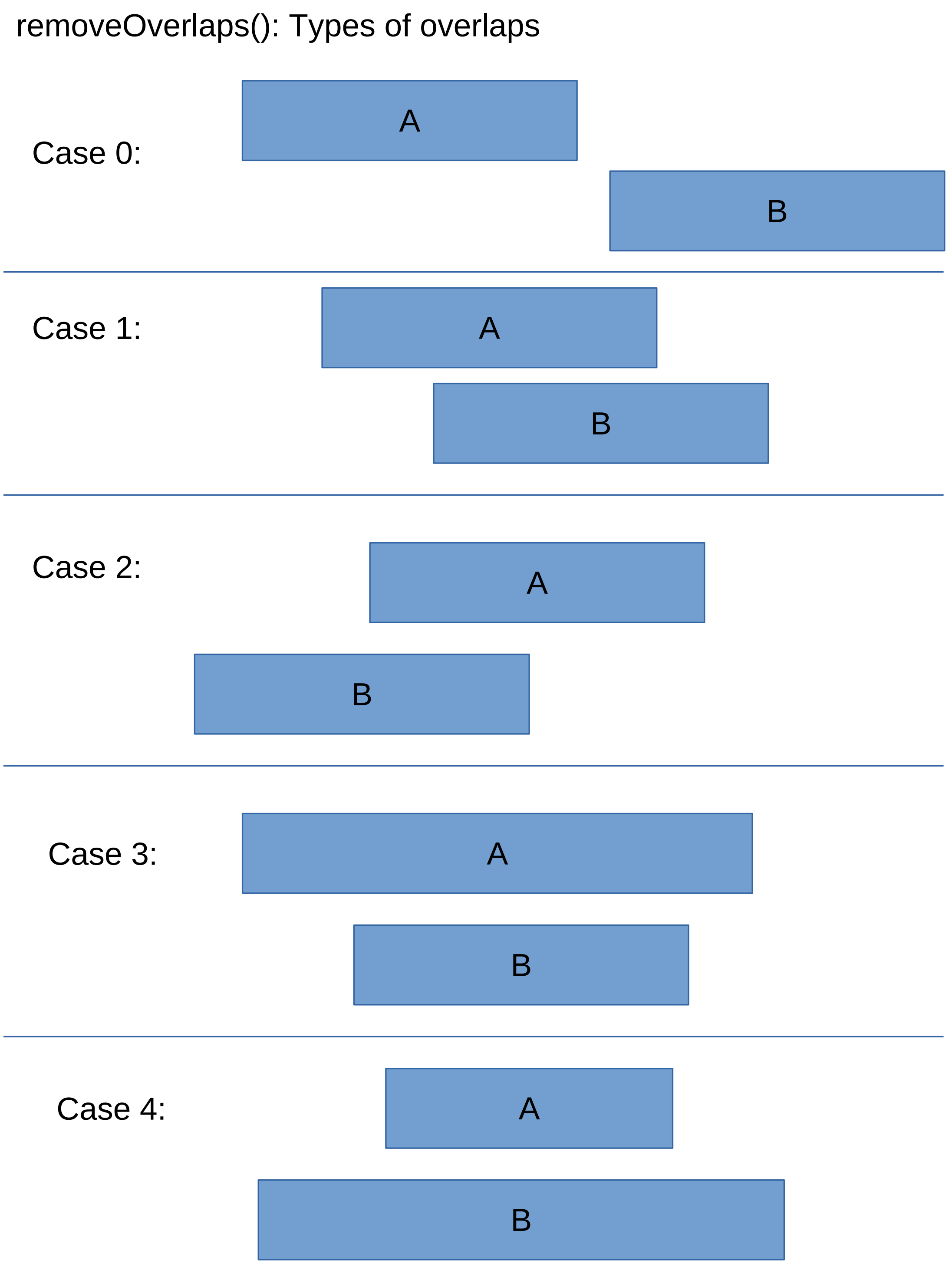
As mentioned in Section 2.2, “Program Parameters and Runtime Conditions”, the program is meant to be run on the AMD64 platform. For your information, this does not mean that the program will only run only AMD CPUs. AMD licences the AMD64 instruction set to Intel and any modern Intel processor will be able to run programs compiled for the AMD64 instruction set.

1. Diagrams

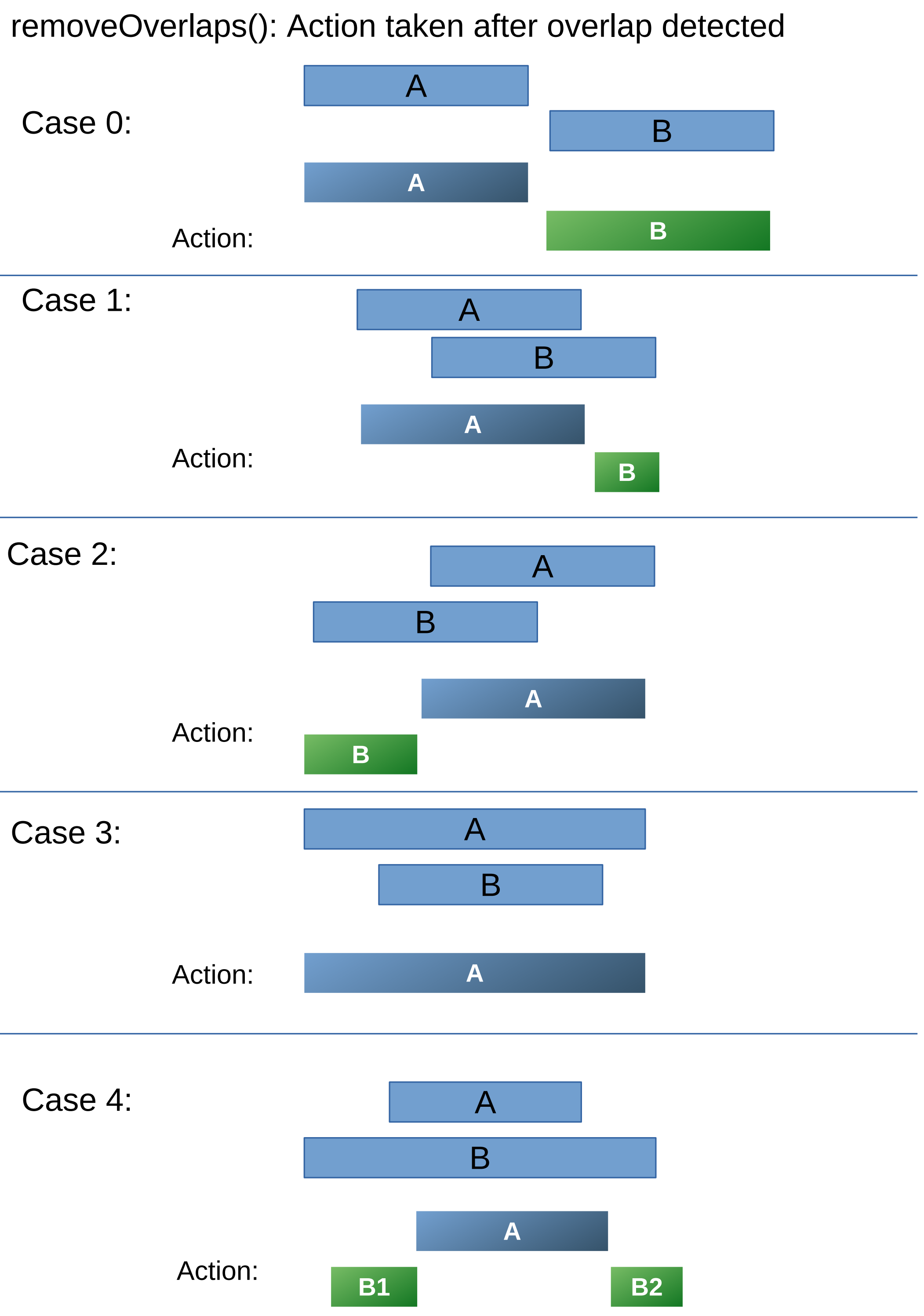
1.1. Class Diagram



1.2. Removing Overlaps



1.3. Removing Overlaps: Action Taken


removeOverlaps(): Action taken after overlap detected
Case 0:
A
B
Action:
A
B
Case 1:
A
B
Action:
A
B
Case 2:
A
B
Action:
A
B
Case 3:
A
B
Action:
A
Case 4:
A
B
Action:
A
B1
B2

Electric Era Charging Network Uptime Report Display Class Documentation

Electric Era Charging Network Uptime Report Display Class Documentation

Table of Contents

List of Tables

Chapter 1. Todo List

Class ChargingNodes::Charger	write additional usage info.

Chapter 2. Class Documentation

1. Availability::AvailabilityEvent Class Reference

Encapsulates a single line of the Charger Availability Reports section of the input data file.

```
#include <AvailabilityEvent.h>
```

Public Member Functions

- AvailabilityEvent ()

  Default constructor.
- AvailabilityEvent (const AvailabilityEvent &other)

  Copy constructor.
- AvailabilityEvent (nanoseconds_t startTime, nanoseconds_t endTime, bool available)

  Constructor.
- ~AvailabilityEvent ()

  Destructor.
- AvailabilityEvent & operator= (const AvailabilityEvent &other)

  Assignment operator.
- AvailabilityEvent (AvailabilityEvent &&)

  Move constructor.
- AvailabilityEvent & operator= (AvailabilityEvent &&other)

  Move assignment operator.
- bool operator== (const AvailabilityEvent &other) const

  Equality operator.
- bool operator!= (const AvailabilityEvent &other) const

  Inequality operator.
- void **insertAvailabilityEvent** (shared_ptr< AvailabilityEvent > ae)
- std::partial_ordering operator<=> (const AvailabilityEvent &other) const noexcept

  Spaceship comparison operator.

Public Attributes

- nanoseconds_t startTime {0}

  Start time of the availability event in nanoseconds.

- nanoseconds_t endTime {0}

  End time of the availability event in nanoseconds.

- bool available {false}

  Whether the charger was available or not during this time period.

1.1. Detailed Description

Encapsulates a single line of the Charger Availability Reports section of the input data file.

See Spec Section 2.1.4, 2.1.10 to 2.1.13, 2.1,13.f to 2.1.13.m, and 2.1 generally.

1.2. Constructor & Destructor Documentation

1.2.1. AvailabilityEvent()`[1/2]`

```
Availability::AvailabilityEvent::AvailabilityEvent (const Availabili-
tyEvent & other)[default]
```

Copy constructor.

Parameters .

Table 2.1.

other	object being copied

1.2.2. AvailabilityEvent()`[2/2]`

```
Availability::AvailabilityEvent::AvailabilityEvent  (AvailabilityEvent
&& )[default]
```

Move constructor.

Parameters .

Table 2.2.

The	object to be moved from

1.3. Member Function Documentation

1.3.1. operator!=()

```
bool Availability::AvailabilityEvent::operator!= (const Availabili-
tyEvent & other) const
```

Inequality operator.

Parameters .

Table 2.3.

other	the object being compared with

Returns. false if a single member is not equal

1.3.2. operator<=>()

```
std::partial_ordering     Availability::AvailabilityEvent::operator<=>
(const AvailabilityEvent & other) const[noexcept]
```

Spaceship comparison operator.

Sorts on startTime, then endTime, then available

Parameters .

Table 2.4.

other	object to compare

Returns. std::partial_ordering

1.3.3. operator=()`[1/2]`

```
AvailabilityEvent & Availability::AvailabilityEvent::operator= (Avail-
abilityEvent && other)[default]
```

Move assignment operator.

Parameters .

Table 2.5.

other	The object to be moved from

Returns. AvailabilityEvent&

1.3.4. operator=()`[2/2]`

```
AvailabilityEvent & Availability::AvailabilityEvent::operator= (const
AvailabilityEvent & other)[default]
```

Assignment operator.

Parameters .

Table 2.6.

other	the object being assigned from

Returns. object reference

1.3.5. operator==()

```
bool  Availability::AvailabilityEvent::operator==  (const  Availabili-
tyEvent & other) const
```

Equality operator.

Parameters .

Table 2.7.

other	the object being compared with

Returns. true if all members are equal

1.4. Member Data Documentation

1.4.1. available

```
bool Availability::AvailabilityEvent::available {false}
```

Whether the charger was available or not during this time period.

true or false. See Spec Section 2.1.9 and 2.1.13.g

1.4.2. endTime

```
nanoseconds_t Availability::AvailabilityEvent::endTime {0}
```

End time of the availability event in nanoseconds.

See Spec Section 2.1.8 and 2.1.13.g

1.4.3. startTime

```
nanoseconds_t Availability::AvailabilityEvent::startTime {0}
```

Start time of the availability event in nanoseconds.

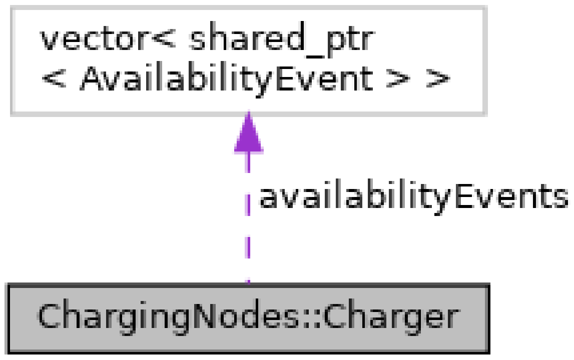
See Spec Section 2.1.8 and 2.1.13.g

The documentation for this class was generated from the following files:
AvailabilityEvent.hAvailabilityEvent.cpp

2. ChargingNodes::Charger Class Reference

Encapsulates a single charger.

`#include <Charger.h>`
Collaboration diagram for ChargingNodes::Charger:



Public Member Functions

- Charger ()

  Default constructor.

- Charger (const Charger &other)

  Copy constructor.

- **Charger** (chargerID_t chargerID)
- ~Charger ()

  Destructor.

- Charger & operator= (const Charger &other)

  Assignment operator.

- Charger (Charger &&)

  Move constructor.

- Charger & operator= (Charger &&other)

  Move assignment operator.

- bool operator== (const Charger &other) const

  Equality operator.

- bool operator!= (const Charger &other) const

  Inequality operator.
- chargerID_t getChargerID () const

  Returns chargerID for this Charger.
- void insertAvailabilityEvent (shared_ptr< AvailabilityEvent > ae)

  Insert an AvailabilityEvent.
- vector< shared_ptr< AvailabilityEvent > > getAvailabilityEvents () const

  Returns a vector of shared_ptr's to AvailabilityEvent's for this Charger.

Protected Attributes

- chargerID_t chargerID

  The ID for this charger.
- vector< shared_ptr< AvailabilityEvent > > availabilityEvents

  A container of AvailabilityEvent's for this Charger.

Friends

- class **Availability::StationAvailabilityReportFactory**

2.1. Detailed Description

Encapsulates a single charger.

Its identity lies in the ChargerID of the charger.

Usual usage:

```
chargerID_t chargerID = 1003;
Charger c( chargerID );
cout << c;
```

Todo

write additional usage info.

2.2. Constructor & Destructor Documentation

2.2.1. Charger()`[1/3]`

```
ChargingNodes::Charger::Charger ( )[default]
```

Default constructor.

Using Rule of Zero, but we specify C++ default.

2.2.2. Charger()[2/3]

```
ChargingNodes::Charger::Charger (const Charger & other)[default]
```

Copy constructor.

C++ default.

Parameters .

Table 2.8.

other	the object being copied from

2.2.3. ~Charger()

```
ChargingNodes::Charger::~Charger ( )
```

Destructor.

C++ default.

2.2.4. Charger()[3/3]

```
ChargingNodes::Charger::Charger (Charger && )[default]
```

Move constructor.

C++ default.

Parameters .

Table 2.9.

The	object to be moved from

2.3. Member Function Documentation

2.3.1. getAvailabilityEvents()

```
vector<shared_ptr<AvailabilityEvent>      >      ChargingNodes::Charg-
er::getAvailabilityEvents ( ) const[inline]
```

Returns a vector of shared_ptr's to AvailabilityEvent's for this Charger.

Returns. std::vector< std::shared_ptr< AvailabilityEvent > >

2.3.2. getChargerID()

```
chargerID_t ChargingNodes::Charger::getChargerID ( ) const
```

Returns chargerID for this Charger.

Returns. chargerID_t

2.3.3. insertAvailabilityEvent()

```
void ChargingNodes::Charger::insertAvailabilityEvent (shared_ptr<
AvailabilityEvent > ae)
```

Insert an AvailabilityEvent.

Pass a shared_ptr to an AvailabilityEvent and this object will cache a copy of the pointer to use later.

Parameters .

Table 2.10.

ae	shared_ptr to an AvailabilityEvent.

2.3.4. operator!=()

```
bool ChargingNodes::Charger::operator!= (const Charger & other) const
```

Inequality operator.

Parameters .

Table 2.11.

other	the object being compared with

Returns. bool

2.3.5. operator=()[1/2]

```
Charger & ChargingNodes::Charger::operator= (Charger && other)[default]
```

Move assignment operator.

C++ default.

Parameters .

Table 2.12.

other	The object to be moved from

Returns. Charger&

2.3.6. operator=()[2/2]

```
Charger & ChargingNodes::Charger::operator= (const Charger & other)[de-
fault]
```

Assignment operator.

C++ default.

Parameters .

Table 2.13.

other	the object being copied from

Returns. Charger&

2.3.7. operator==()

```
bool ChargingNodes::Charger::operator== (const Charger & other) const
```

Equality operator.

Parameters .

Table 2.14.

other	the object being compared with

Returns. bool

2.4. Member Data Documentation

2.4.1. availabilityEvents

```
vector<shared_ptr<AvailabilityEvent> > ChargingNodes::Charger::avail-
abilityEvents[protected]
```

A container of AvailabilityEvent's for this Charger.

An AvailabilityEvent encapsulates a single line of the [Charger Availability Reports] section of the input data file. I.e., it has a charger ID, start time, end time, and availability.

2.4.2. chargerID

```
chargerID_t ChargingNodes::Charger::chargerID[protected]
```

The ID for this charger.

This is a 32-bit unsigned integer.

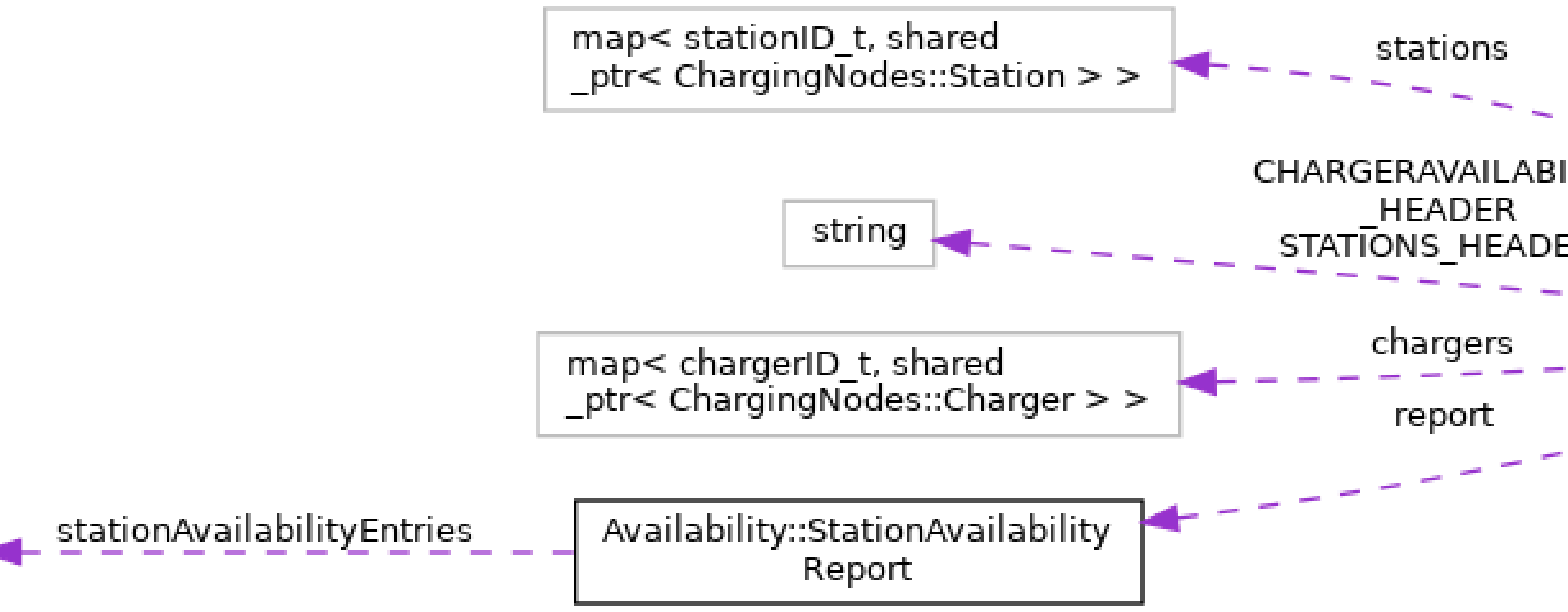
The documentation for this class was generated from the following files:

Charger.hCharger.cpp

3. Charging::ChargingNetwork Class Reference

Encapsulates a charging network.

`#include <ChargingNetwork.h>`
Collaboration diagram for Charging::ChargingNetwork:


map< stationID_t, shared _ptr< ChargingNodes::Station > >
stations
CHARGERAVAILABI
_HEADER
STATIONS_HEADE
string
map< chargerID_t, shared _ptr< ChargingNodes::Charger > >
chargers
report
stationAvailabilityEntries
Availability::StationAvailability Report

Public Member Functions

- ChargingNetwork ()

  Default constructor.

- ChargingNetwork (const ChargingNetwork &other)

  Copy constructor.

- ChargingNetwork (const ::std::filesystem::path &inputFile)

  Constructor with dependency passed in.

- ~ChargingNetwork ()

  Destructor.

- ChargingNetwork & operator= (const ChargingNetwork &other)

  Assignment operator.

- ChargingNetwork (ChargingNetwork &&)

Move constructor.

- ChargingNetwork & operator= (ChargingNetwork &&other)

  Move assignment operator.

- StationAvailabilityReport getStationAvailabilityReport () const

  Get the station availabilty report.

Static Public Attributes

- static const ::std::string_view ERROR_TEXT {"ERROR"}

  Text to print when there is an error.

Protected Attributes

- map< stationID_t, shared_ptr< Station > > **stations**
- map< chargerID_t, shared_ptr< Charger > > **chargers**
- StationAvailabilityReport **report**

Static Protected Attributes

- static const string STATIONS_HEADER {"[Stations]"}

  The heading in the data file preceding the info on stations.

- static const string CHARGERAVAILABILITY_HEADER {"[Charger Availability Reports]"}

  The heading in the data file preceding the info on uptime.

3.1. Detailed Description

Encapsulates a charging network.

Usual usage is to pass the input data file location via parameter to the constructor (Constructor DI):

```
const std::filesystem::path  chargingNetworkDataFile {"/path/to/data/file"};
ChargingNetwork cn {chargingNetworkDataFile};
```

3.2. Constructor & Destructor Documentation

3.2.1. ChargingNetwork() [1/4]

```
Charging::ChargingNetwork::ChargingNetwork ( )[default]
```

Default constructor.

Using Rule of Zero for five basic methods.

3.2.2. ChargingNetwork() [2/4]

`Charging::ChargingNetwork::ChargingNetwork (const ChargingNetwork & other)[default]`

Copy constructor.

Default.

Parameters .

Table 2.15.

other	the object being copied from

3.2.3. ChargingNetwork() [3/4]

`Charging::ChargingNetwork::ChargingNetwork (const ::std::filesystem::path & inputFile)`

Constructor with dependency passed in.

Pass the location of the input data file. See Spec Section 2.1 and 3.4 Throws std::filesystem::filesystem_error if the specified input data file is not found.

```
const std::filesystem::path  chargingNetworkDataFile {"/path/to/data/file"};
ChargingNetwork cn {chargingNetworkDataFile};
```

Parameters .

Table 2.16.

inputFile	The path to the input data file

3.2.4. ~ChargingNetwork()

`Charging::ChargingNetwork::~ChargingNetwork ( )[default]`

Destructor.

Default.

3.2.5. ChargingNetwork() [4/4]

`Charging::ChargingNetwork::ChargingNetwork (ChargingNetwork && )[default]`

Move constructor.

Parameters .

Table 2.17.

The	object to be moved from

3.3. Member Function Documentation

3.3.1. getStationAvailabilityReport()

`StationAvailabilityReport Charging::ChargingNetwork::getStationAvailabilityReport ( ) const`

Get the station availabilty report.

```
ChargingNetwork cn {chargingNetworkDataFile};
StationAvailabilityReport report = cn.getStationAvailabilityReport();
cout << report;
```

Returns. StationAvailabilityReport

3.3.2. operator=()[1/2]

`ChargingNetwork & Charging::ChargingNetwork::operator= (ChargingNetwork && other)[default]`

Move assignment operator.

Parameters .

Table 2.18.

other	The object to be moved from

Returns. ChargingNetwork&

3.3.3. operator=()[2/2]

`ChargingNetwork & Charging::ChargingNetwork::operator= (const ChargingNetwork & other)[default]`

Assignment operator.

Default.

Parameters .

Table 2.19.

other	the object being assigned from

Returns. object reference

3.4. Member Data Documentation

3.4.1. CHARGERAVAILABILITY_HEADER

```
const string Charging::ChargingNetwork::CHARGERAVAILABILITY_HEADER
{"[Charger Availability Reports]"}[inline], [static], [protected]
```

The heading in the data file preceding the info on uptime.

After this heading is a list of chargers and their uptimes.

3.4.2. ERROR_TEXT

```
const ::std::string_view Charging::ChargingNetwork::ERROR_TEXT {"ER-
ROR"}[inline], [static]
```

Text to print when there is an error.

See Spec Section 2.3.1.

3.4.3. STATIONS_HEADER

```
const string Charging::ChargingNetwork::STATIONS_HEADER {"[Station-
s]"}[inline], [static], [protected]
```

The heading in the data file preceding the info on stations.

After this heading is the list of stations and their associated chargers.

The documentation for this class was generated from the following files:
ChargingNetwork.hChargingNetwork.cpp

4. std::less< ChargingNodes::Charger > Struct Reference

Comparator to allow Charger to be added to a map.

```
#include <Charger.h>
```

Public Member Functions

- bool **operator()** (const ChargingNodes::Charger &a, const ChargingNodes::Charger &b) const

4.1. Detailed Description

Comparator to allow Charger to be added to a map.

The documentation for this struct was generated from the following file:
Charger.h

5. std::less< ChargingNodes::Station > Struct Reference

Comparator to allow Station to be added to a map.

`#include <Station.h>`

Public Member Functions

- bool **operator()** (const ChargingNodes::Station &a, const ChargingNodes::Station &b) const

5.1. Detailed Description

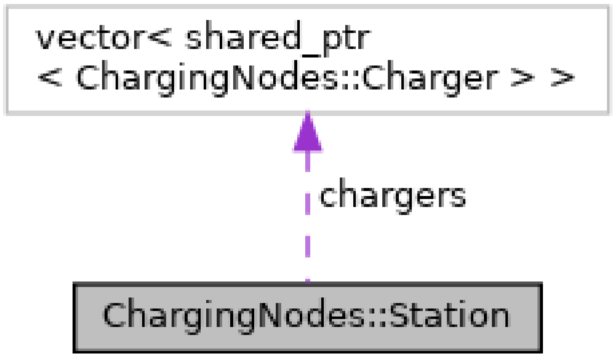
Comparator to allow Station to be added to a map.

The documentation for this struct was generated from the following file:
Station.h

6. ChargingNodes::Station Class Reference

Encapsulates a single charging station.

`#include <Station.h>`
Collaboration diagram for ChargingNodes::Station:



Public Member Functions

- Station ()

  Default constructor.

- Station (const Station &other)

Copy constructor.

- Station (const stationID_t stationID)

  constructor from stationID

- ~Station ()

  Destructor.

- Station & operator= (const Station &other)

  Assignment operator.

- Station (Station &&)

  Move constructor.

- Station & operator= (Station &&other)

  Move assignment operator.

- bool operator== (const Station &other) const

  Equality operator.

- bool operator!= (const Station &other) const

  Inequality operator.

- stationID_t **getStationID** () const
- void **insertCharger** (shared_ptr< ChargingNodes::Charger > charger)

Protected Attributes

- stationID_t stationID

  Station ID.

- vector< shared_ptr< ChargingNodes::Charger > > chargers

  A vector of Charger's.

Friends

- class **Availability::StationAvailabilityReportFactory**
- std::ostream & **operator<<** (std::ostream &os, const Station &s)

6.1. Detailed Description

Encapsulates a single charging station.

Its identity lies in the station ID. See Spec Section 2.1.2, 2.1.6

Create the Station and add Charger pointers to it:

```
stationID_t stationID {3};
Station station{ stationID };

chargerID_t chargerID{ 1003 };
auto c = std::make_shared<Charger>(chargerID);
station.insertCharger( c );
```

6.2. Constructor & Destructor Documentation

6.2.1. Station()`[1/3]`

`ChargingNodes::Station::Station (const Station & other)[default]`

Copy constructor.

Parameters .

Table 2.20.

other	the object being copied from

6.2.2. Station()`[2/3]`

`ChargingNodes::Station::Station (const stationID_t stationID)`

constructor from stationID

Parameters .

Table 2.21.

other	the station ID from which to construct this object

6.2.3. Station()`[3/3]`

`ChargingNodes::Station::Station (Station && )[default]`

Move constructor.

Parameters .

Table 2.22.

The	object to be moved from

6.3. Member Function Documentation

6.3.1. operator!=()

```
bool ChargingNodes::Station::operator!= (const Station & other) const
```

Inequality operator.

Parameters .

Table 2.23.

other	the object being compared with

Returns. object reference

6.3.2. operator=()[1/2]

```
Station & ChargingNodes::Station::operator= (const Station & other)[de-
fault]
```

Assignment operator.

Parameters .

Table 2.24.

other	the object being copied from

Returns. object reference

6.3.3. operator=()[2/2]

```
Station & ChargingNodes::Station::operator= (Station && other)[default]
```

Move assignment operator.

Parameters .

Table 2.25.

other	The object to be moved from

Returns. Station&

6.3.4. operator==()

```
bool ChargingNodes::Station::operator== (const Station & other) const
```

Equality operator.

Parameters .

Table 2.26.

other	the object being compared with

Returns. object reference

6.4. Member Data Documentation

6.4.1. chargers

```
vector<shared_ptr<ChargingNodes::Charger>    >    ChargingNodes::Sta-
tion::chargers[protected]
```

A vector of Charger's.

Mulitiplicity: Each Station can be associated from zero to many Chargers.

6.4.2. stationID

```
stationID_t ChargingNodes::Station::stationID[protected]
```

Station ID.

Identity of a Station lies within the stationID. 32 bit unsigned int. See Spec Section 2.1.6,

The documentation for this class was generated from the following files:
Station.hStation.cpp

7. Availability::StationAvailabilityEntry Class Reference

Encapsulates a single line of the station availability report.

```
#include <StationAvailabilityEntry.h>
```

Public Member Functions

- StationAvailabilityEntry ()

  Default constructor.

- StationAvailabilityEntry (ChargingNodes::stationID_t stationID, float uptimeFraction)

  Constructor.

- StationAvailabilityEntry (const StationAvailabilityEntry &other)

  Copy constructor.

- ~StationAvailabilityEntry ()

Destructor.

- StationAvailabilityEntry & operator= (const StationAvailabilityEntry &other)

  Assignment operator.

- bool operator< (const StationAvailabilityEntry &other) const noexcept

  Comparison operator less than.

- constexpr std::partial_ordering operator<=> (const StationAvailabilityEntry &other) const noexcept

  Spaceship comparison operator.

Protected Attributes

- ChargingNodes::stationID_t stationID

  The station ID for this station.

- float uptimeFraction

  The fraction of time that any charger at this station was available.

Friends

- class **StationAvailabilityReportFactory**

- std::ostream & operator<< (std::ostream &os, const StationAvailabilityEntry &sae)

  Write to output stream.

7.1. Detailed Description

Encapsulates a single line of the station availability report.

Consists of a station ID and uptime percentage.

```
StationAvailabilityReport report;
report += StationAvailabilityEntry({0, 12.5f/100});
cout << report;
```

7.2. Constructor & Destructor Documentation

7.2.1. StationAvailabilityEntry() [1/3]

```
Availability::StationAvailabilityEntry::StationAvailabilityEntry
( )[default]
```

Default constructor.

C++ default.

7.2.2. StationAvailabilityEntry() [2/3]

```
Availability::StationAvailabilityEntry::StationAvailabilityEntry
(ChargingNodes::stationID_t stationID, float uptimeFraction)
```

Constructor.

Parameters .

Table 2.27.

stationID	The ID of the station which this enty is regarding.
uptimeFraction	The fraction of time that any charger at this station was available. See Spec Section 2.3. We store as a decimal fraction and display as a percentage. For example, if a station was available 12.5% of the time, uptimeFraction would be 0.125.

7.2.3. StationAvailabilityEntry() [3/3]

```
Availability::StationAvailabilityEntry::StationAvailabilityEntry (con-
st StationAvailabilityEntry & other)[default]
```

Copy constructor.

C++ default.

Parameters .

Table 2.28.

other	object to copy

7.2.4. ~StationAvailabilityEntry()

```
Availability::StationAvailabilityEntry::~StationAvailabilityEntry
( )[default]
```

Destructor.

C++ default.

7.3. Member Function Documentation

7.3.1. operator<()

```
bool Availability::StationAvailabilityEntry::operator< (const Station-
AvailabilityEntry & other) const[noexcept]
```

Comparison operator less than.

Sorts primarily on stationID, then on uptimeFraction.

Parameters .

Table 2.29.

other	object to compare

Returns. bool

7.3.2. operator<=>()

```
constexpr  std::partial_ordering  Availability::StationAvailabilityEn-
try::operator<=> (const StationAvailabilityEntry & other) const[const-
expr], [noexcept]
```

Spaceship comparison operator.

Sorts primarily on stationID, then on uptimeFraction.

Parameters .

Table 2.30.

other	object to compare

Returns. std::partial_ordering

7.3.3. operator=()

```
StationAvailabilityEntry & Availability::StationAvailabilityEntry::op-
erator= (const StationAvailabilityEntry & other)[default]
```

Assignment operator.

C++ default.

Parameters .

Table 2.31.

other	object to copy from

Returns. object reference

7.4. Friends And Related Function Documentation

7.4.1. operator<<

```
std::ostream& operator<< (std::ostream & os, const StationAvailabili-
tyEntry & sae)[friend]
```

Write to output stream.

Writes the station ID, a space, and then the uptime percentage. See Spec Section 2.3.7 (Converts the uptimeFraction to percentage, truncates and converts to integer.) So, 12.5% would print as 12. 12.79% would also print as 12. See Spec Section 2.3.8 Doesn't emit a newline.

Parameters .

Table 2.32.

os	output stream
sae	StationAvailabilityEntry to write

Returns. std::ostream&

7.5. Member Data Documentation

7.5.1. stationID

```
ChargingNodes::stationID_t  Availability::StationAvailabilityEntry::s-
tationID[protected]
```

The station ID for this station.

Each entry in the report pertains to one station. Each station has one entry. See Spec Section 2.3.7.

7.5.2. uptimeFraction

```
float  Availability::StationAvailabilityEntry::uptimeFraction[protect-
ed]
```

The fraction of time that any charger at this station was available.

See Spec Section 2.3. We store as a decimal fraction. For example, if a station was available 12.5% of the time, uptimeFraction would be 0.125. We multiple by 100, truncate, and convert to integer to print. So, 12.5% would print as 12.

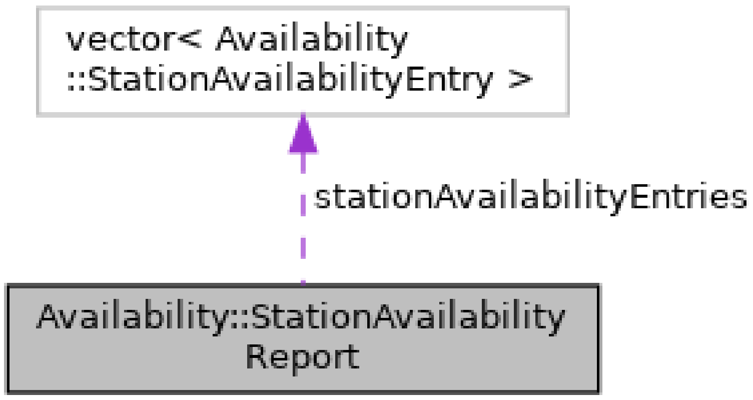
See Spec Section 2.3.7 and 2.3.8

The documentation for this class was generated from the following files:
StationAvailabilityEntry.hStationAvailabilityEntry.cpp

8. Availability::StationAvailabilityReport Class Reference

Encapsulates a station availabilty report.

```
#include <StationAvailabilityReport.h>
```
Collaboration diagram for Availability::StationAvailabilityReport:


vector< Availability ::StationAvailabilityEntry >
stationAvailabilityEntries
Availability::StationAvailability Report

Public Member Functions

- StationAvailabilityReport ()

  Default constructor.
- StationAvailabilityReport (const StationAvailabilityReport &other)

  Copy constructor.
- ~StationAvailabilityReport ()

  Destructor.
- StationAvailabilityReport & operator= (const StationAvailabilityReport &other)

  Assignment operator.
- **StationAvailabilityReport** (StationAvailabilityReport &&other)
- StationAvailabilityReport & **operator=** (StationAvailabilityReport &&other)
- StationAvailabilityReport & operator+= (const StationAvailabilityEntry &sae)

  Insert a StationAvailabilityEntry.
- void sort ()

  Sort the report entries by stationID.

Protected Attributes

- vector< StationAvailabilityEntry > stationAvailabilityEntries

  A vector of StationAvailabilityEntry's.

Friends

- std::ostream & operator<< (std::ostream &os, const StationAvailabilityReport &sar)

  Write to output stream.

8.1. Detailed Description

Encapsulates a station availabilty report.

See Spec Section 2.3.4 to 2.3.9

```
StationAvailabilityReport report;
report += StationAvailabilityEntry({0, 12.5f/100});
cout << report;
```

Normally, you will not create a report manually, but rather will get one from the StationAvailabilityReportFactory .

8.2. Constructor & Destructor Documentation

8.2.1. StationAvailabilityReport()[1/2]

```
Availability::StationAvailabilityReport::StationAvailabilityReport
( )[default]
```

Default constructor.

C++ default.

8.2.2. StationAvailabilityReport()[2/2]

```
Availability::StationAvailabilityReport::StationAvailabilityReport
(const StationAvailabilityReport & other)[default]
```

Copy constructor.

C++ default.

Parameters .

Table 2.33.

other	object to be copied

8.3. Member Function Documentation

8.3.1. operator+=()

```
StationAvailabilityReport & Availability::StationAvailabilityReport::operator+= (const StationAvailabilityEntry & sae)
```

Insert a StationAvailabilityEntry.

The StationAvailabilityReport consists of a series of StationAvailabilityEntry's.

Parameters .

Table 2.34.

sae	StationAvailabilityEntry to insert

Returns. StationAvailabilityReport&

8.3.2. operator=()

```
StationAvailabilityReport & Availability::StationAvailabilityRe-
port::operator= (const StationAvailabilityReport & other)[default]
```

Assignment operator.

C++ default.

Parameters .

Table 2.35.

other	object to be copied from

Returns. object reference.

8.4. Friends And Related Function Documentation

8.4.1. operator<<

```
std::ostream& operator<< (std::ostream & os, const StationAvailabili-
tyReport & sar)[friend]
```

Write to output stream.

Only emits a newline after entries *other than* the first one. This means there is no newline after the last entry. There is no newline if there are no entries. See Spec output sample files.

Parameters .

Table 2.36.

os	Output stream to write to.

sar	StationAvailabilityReport to write
	`StationAvailabilityReport report;`
	`report += StationAvailabilityEntry({0, 12.5f/100});`
	`cout << report;`

Returns. std::ostream&

8.5. Member Data Documentation

8.5.1. stationAvailabilityEntries

`vector<StationAvailabilityEntry> Availability::StationAvailabilityReport::stationAvailabilityEntries[protected]`

A vector of StationAvailabilityEntry's.

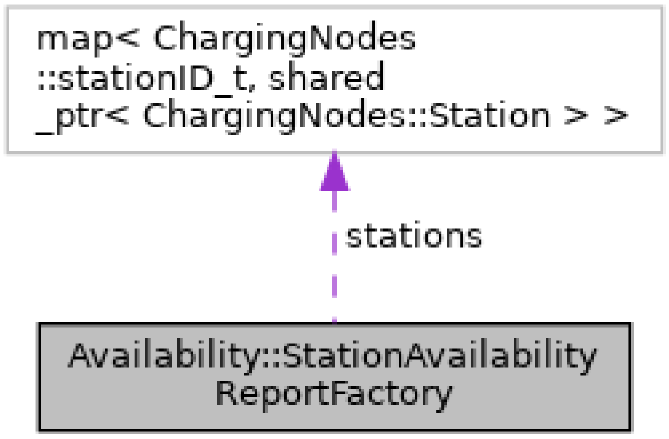
They include the stationID and the uptime fraction (the fraction of time that any charger at a given station was availabile.

The documentation for this class was generated from the following files:
StationAvailabilityReport.hStationAvailabilityReport.cpp

9. Availability::StationAvailabilityReportFactory Class Reference

Factory for creating StationAvailabilityReport.

`#include <StationAvailabilityReportFactory.h>`
Collaboration diagram for Availability::StationAvailabilityReportFactory:


map< ChargingNodes ::stationID_t, shared _ptr< ChargingNodes::Station > >
stations
Availability::StationAvailability ReportFactory

Public Member Functions

- StationAvailabilityReportFactory ()

  Default constructor.

- StationAvailabilityReportFactory (map< ChargingNodes::stationID_t, shared_ptr< ChargingNodes::Station >> stations)

  Constructor.

- StationAvailabilityReportFactory (const StationAvailabilityReportFactory &other)

  Copy constructor.

- ~StationAvailabilityReportFactory ()

  Destructor.

- StationAvailabilityReportFactory & operator= (const StationAvailabilityReportFactory &other)

  Assignment operator.

- StationAvailabilityReport getReport ()

  Get the station availability report.

Protected Attributes

- map< ChargingNodes::stationID_t, shared_ptr< ChargingNodes::Station > > stations

  A map of Station's, keyed by station ID.

9.1. Detailed Description

Factory for creating StationAvailabilityReport.

It depends on a container of Station's. Inject via constructor:

```
map<stationID_t, shared_ptr<Station>> stations;
... add a bunch of Station's to the container with insert()
auto factory = StationAvailabilityReportFactory(this->stations);
StationAvailabilityReport report =  factory.getReport();
```

Why not just pass the stations map to the StationAvailabilityReport object? Well, the report doesn't need to concern itself with Stations and all of its attendant info (such as each station's chargers and each charger's availability). The report only contains entries for each station and its availability.

9.2. Constructor & Destructor Documentation

9.2.1. StationAvailabilityReportFactory()`[1/3]`

```
Availability::StationAvailabilityReportFactory::StationAvailabilityRe-
portFactory ( )[default]
```

Default constructor.

C++ default.

9.2.2. StationAvailabilityReportFactory()`[2/3]`

```
Availability::StationAvailabilityReportFactory::StationAvailabilityRe-
portFactory (map< ChargingNodes::stationID_t, shared_ptr< ChargingN-
odes::Station >> stations)
```

Constructor.

Parameters .

Table 2.37.

stations	a container of Station's on which to report.

9.2.3. StationAvailabilityReportFactory()[3/3]

```
Availability::StationAvailabilityReportFactory::StationAvailabilityRe-
portFactory (const StationAvailabilityReportFactory & other)[default]
```

Copy constructor.

C++ default.

Parameters .

Table 2.38.

other	object to copy.

9.2.4. ~StationAvailabilityReportFactory()

```
Availability::StationAvailabilityReportFactory::~StationAvailabili-
tyReportFactory ( )[default]
```

Destructor.

C++ default.

9.3. Member Function Documentation

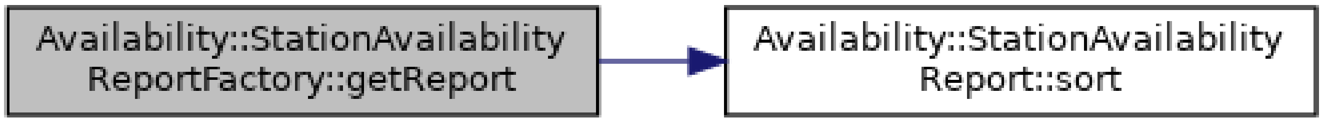
9.3.1. getReport()

```
StationAvailabilityReport  Availability::StationAvailabilityReportFac-
tory::getReport ( )
```

Get the station availability report.

Returns an object which contains StationAvailabilityEntry's, which are lines of the station availability report.

Returns. StationAvailabilityReport
Here is the call graph for this function:


Availability::StationAvailability ReportFactory::getReport
Availability::StationAvailability Report::sort

9.3.2. operator=()

`StationAvailabilityReportFactory & Availability::StationAvailabilityReportFactory::operator= (const StationAvailabilityReportFactory & other)[default]`

Assignment operator.

C++ default.

Parameters .

Table 2.39.

other	object to copy from.

Returns. object reference

9.4. Member Data Documentation

9.4.1. stations

`map<ChargingNodes::stationID_t, shared_ptr<ChargingNodes::Station> > Availability::StationAvailabilityReportFactory::stations[protected]`

A map of Station's, keyed by station ID.

The need for this is that this object will iterate over its Station's, generating a report of availability by station. `stations` contains all the info needed for creating the report.

The documentation for this class was generated from the following files:
StationAvailabilityReportFactory.hStationAvailabilityReportFactory.cpp

Index