TEAM PROJECT SmartHome

Course IT 538 SW-T

SmartHome FORDCS System Document

version v2022.1

By: Ömer Karacan

Supervisor: Ömer Karacan

Vienna, 17.03.2022

Table of Contents

[1. Ford Charging Station (FORDCS) 3](#_Toc98621650)

[1.1. Use Case “Report Factory Setting” 3](#_Toc98621651)

[1.1.1. Requirements 3](#_Toc98621652)

[1.1.2. System Interface 4](#_Toc98621653)

[1.1.3. Validation/Verification 4](#_Toc98621654)

[1.2. Use Case “Run-time Status Report” 5](#_Toc98621655)

[1.2.1. Requirements 5](#_Toc98621656)

[1.2.2. System Interface 5](#_Toc98621657)

[1.2.3. Validation/Verification 5](#_Toc98621658)

[1.3. Use Case “FSM Report” 6](#_Toc98621659)

[1.3.1. Requirements 6](#_Toc98621660)

[1.3.2. System Interface 6](#_Toc98621661)

[1.3.3. Implementation Hints 6](#_Toc98621662)

[1.3.4. Validation/Verification 7](#_Toc98621663)

# Ford Charging Station (FORDCS)

This section specifies the integration of the “Ford Charging Station” into the SmartHome Distributed Software System (DSS).

The FORDCS is defined by Use Cases.

# Use Case “Report Factory Setting”

# Requirements

The integration of the “Ford Charging Station” is a specialization of the **DSS\_UC3.1 “Report charging station”** and embedded in the same context. The communication between external entities and component and among components is described with request/response service calls, as for DSS\_UC3.1.

The integration of the new Charging Station has no impact on the subsystem/component architecture, and it can be seen as an extension of the domain model.

The most important agreement with the stakeholders is the definition of the unique identity of the Ford Charging Station.

Ford Charging Station Device ID: **CS\_FORD\_EVSE\_RS**

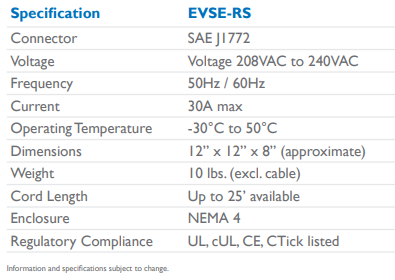


Figure 1 Ford Charging Station Report Data

The factory setting data structure of the Ford Charging Station “**FordCSSettingDataX”:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute name** | **Description** | **Data Type** | **Verification method** |
| DeviceID: | A unique device ID incorporates the specification EVSE-RS | string | = CS\_FORD\_EVSE\_RS |
| ConnectorType | Always SAEJ1772 | string | SAEJ1772 |
| Voltage | Charging voltage | integer | 208 <= [VAC] =< 240 |
| Frequency | Charging frequency | integer | = 50[Hz] OR 60[Hz] |
| Current | Charging current | floating point | 0 <= [A] =< 30 |
| OTemperature | Operating Temperature | floating point | -30 <= [C] =< +70 |
| Dimensions | Approximate physical dimension | string | 12x12x8 [inchVolume] |
| Weight | Approximate weight without cable | floating point | = 10 [lbs] |
| CordLength | Approximate cord length | floating point | <= 25 [inch] |
| Enclosure | Charging plug | string | = “NEMA 4” |
| Regulatory Compliance | Certificated international norms | string | “UL cUL CE CTick” |

Table 1 Ford CS Report Data

# System Interface

A webservices “**Report Factory Setting“** for reporting should be implemented:

***GET „/reportCsFactorySetting” // returns FordCSSettingDataX***

Side-note: refer for naming conventions to <https://hub.packtpub.com/best-practices-for-restful-web-services-naming-conventions-and-api-versioning-tutorial/>

# Validation/Verification

The Validation/Verification requirements are:

* An **Integration Test** over the “GET /reportCsFactorySetting” interface should test the “Voltage”, “OTemperature”, and “Cord length” attributes utilizing the black-box test techniques
  + “Equivalence Partitioning” and
  + “Boundary Value Analysis”
* with a system test tool, e.g., Postman!
* The minimum code coverage value is 80%!

Side-note:

* A **Component Test** should test the “Voltage”, “OTemperature”, and “Cord length” attributes utilizing the black-box test techniques
  + “Equivalence Partitioning” and
  + “Boundary Value Analysis”
* with a component test tool, e.g., JUnit!
* The minimum code coverage value is 80%!

# Use Case “Run-time Status Report”

# Requirements

The device user interface (see Figure 2) informs the driver about the current status of the Charging Station.

A status is defined by color and textual information!

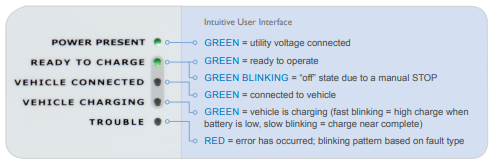


Figure 2 Ford Charging Station Run-time Status

# System Interface

A webservices “**Run-time Status Report“** should be implemented to determine the Run-time Status according to run-time status names and values that are given in Figure 2!

*GET „/runtimeStatusReport” // returns run-time status{status-name, status, status description}*

*// e.g. {“READY TO CHARGE”, GREEN, “ready to operate”}*

Following table defines the parameters of the return object “**RuntimeStatusReportDataX**”

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute name** | **Description** | **Data Type** | **Verification method** |
| StatusName: | Unique status name on the plug UX | string | One of the states  “POWER PRESENT”, “READY TO CHARGE”, “VEHICLE CONNECTED”, “VEHICLE CHARGING”, “TROUBLE” |
| Status | Current status | string | One of the corresponding status state “GREEN”, “GREEN BLINKING”, “RED” |
| StatusDescription | Human friendly status description | string | text |

Table 2 Run-time Status Report Data

Side-note: refer for naming conventions to <https://hub.packtpub.com/best-practices-for-restful-web-services-naming-conventions-and-api-versioning-tutorial/>

# Validation/Verification

The Validation/Verification requirement is:

* The minimum code coverage value is 100%!

# Use Case “FSM Status Report”

# Requirements

The system core behavior is defined by the FMS modelled in Figure 3.

*REQ\_FSM\_1: The last received event should be stored as the last received trigger event with a time stamp!*

For safety, maintenance, and test reasons this interface should be implemented and thoroughly tested!

# System Interface

The system interfaces provide access to the system core behavior “FORDCS Finite State Machine (FSM)”!

*GET provideVehicleChargingState // request the current FSM state*

*// responses with the current FSM state!*

*call synopsis:* [*http://localhost:8080/provideVehicleChargingState*](http://localhost:8080/provideVehicleChargingState)

# Validation/Verification

The Validation/Verification requirements are:

* A **System Test** should be designed to determine the Charging Station status
* A **Component Test** should be designed according to the Black-box Test Technique
  + “State Transition Testing”
* The code coverage value is 100% having a very high risk!

# Use Case “Software Component Report”

# Requirements

The information of the Software Components should be available at the system level.

The system core behavior is defined by the FMS modelled in Figure 3.

*REQ\_FSM\_2: The implemented FSM Model version should be available at system level!*

# System Interface

The system interface provides the implemented version of the “FORDCS Finite State Machine (FSM)”!

*GET provideSoftwareComponentReport/{SoftwareComponent}// request the software component*

*// responses with software component info!*

*call synopsis:* [*http://localhost:8080/provideSoftwareComponentReport/*FSMComponent](http://localhost:8080/provideSoftwareComponentReport/FSMComponent%20)

SoftwareComponentReport Data structure:

|  |  |  |
| --- | --- | --- |
| **SWCname** | **SWCDescription** | **SWCValue** |
| example |  |  |
| “FSMComponent” | “Implemented FSM Model version” | “2022.03” |

application.properties: For FSM model

**SWCnameFSM**=

**SWCDescriptionFSM** =

**SWCValueFSM** =

# Validation/Verification

The Validation/Verification requirements are:

* A **System Test** should be designed to determine the FSM model version, which should be equal to the version given in the application.properties file
* A **Component Test** should be designed to determine the FSM model version, which should be equal to the version given in the application.properties file!
* The code coverage value is 100%!

# Edu-Emulation - Implementation Hints

The run-time status of the Ford Charging Station should be implemented as a Finite State Machine (FSM), which can be performed utilizing different design patterns.

The pattern “Nested Switch Implementation” is recommended for implementation, see <http://www.cs.fsu.edu/~lacher/courses/COP5385/lectures/QP3/script.html>.

The following state diagram is modelled to visualize the event driven behaviour of the charging station.

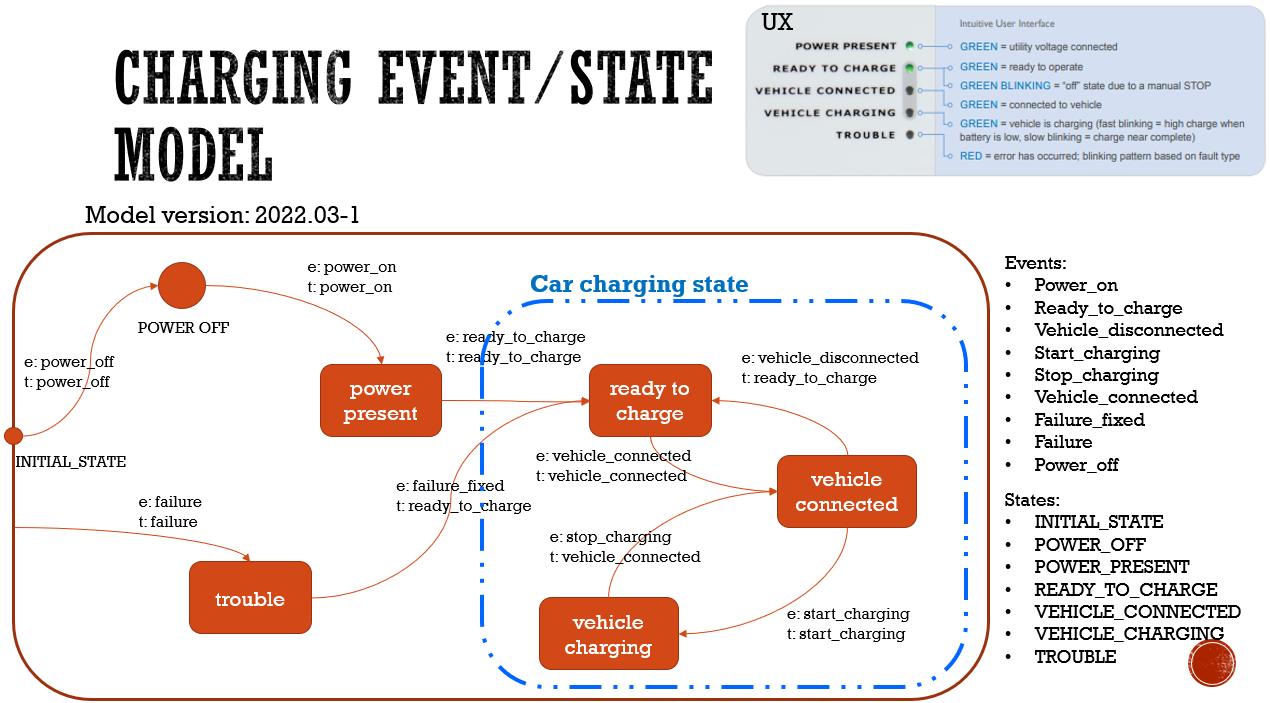


Figure 3 Ford Charging Station State Diagram

The following shows the state/event table of the “Car Charging State” comprising vehicle charging sub-states.

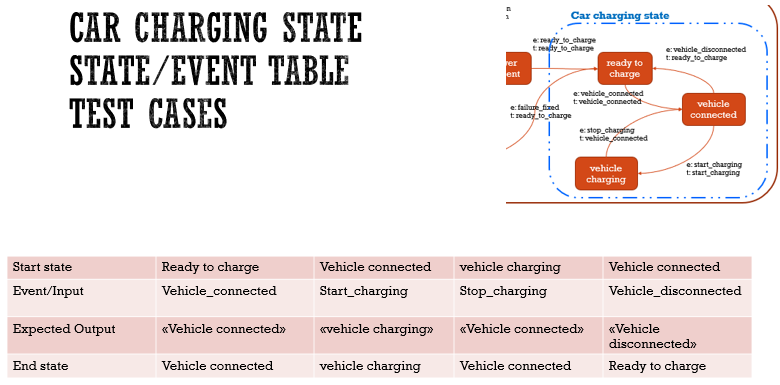


Figure 4 Vehicle Charging Sub-States