codibly | part of spyrosoft

A Developer Perspective
Coping with a variety of standards - OpenADR, OCPP, etc.

codibly.com

Łukasz Kulczyński
Executive Vice President | Head of eMobility

+48 538 050 158
lukasz.kulczynski@codibly.com
Linkedin

codibly.com

Agenda

1. Codibly
2. Standards and their Role
3. Architecture design of different standards
4. Implementation models
5. Case Study - QUIZ
6. Which stack should I use?
7. Most common Questions from our Clients
8. Case Study EVSE x DEMAND RESPONSE

codibly | part of spyrosoft

openADR ALLIANCE

we integrate global energy industry through software solutions

codibly.com

codibly | part of spyrosoft

openADR ALLIANCE

ABOUT

13 Years in the Industry

150+ Energy Projects

20 Offices Globally

1,600 Experts

codibly.com

SOLUTIONS

- CHARGING MANAGEMENT SYSTEMS
- DEMAND RESPONSE
- eFLEET CHARGING
- DYNAMIC LOAD MANAGEMENT
- eMSP
- DRIVER APPS
- SMART CHARGING
- INTEGRATIONS (OCPP, OCPI, OICP, OSCP)
- V2X

- DEMAND RESPONSE/VPP INTEGRATIONS
- PROTOCOLS (OPENADR, IEEE 2030.5, ETC.)
- ENERGY MANAGEMENT SYSTEMS
- MICROGRIDS MANAGEMENT & CONTROLS
- ASSET MANAGEMENT PLATFORMS
- ENERGY TRADING

- VEHICLE INFOTAINMENT
- SAFETY and CYBERSECURITY
- AUTONOMOUS DRIVING
- HMI

HOW?

READY TO IMPLEMENT INTEGRATIONS

Open-Source Components

TEAM AUGMENTATION

From Design to Deployment

CUSTOM SOFTWARE DEVELOPMENT

codibly | part of spyrosoft

Selected Case Studies & Customers

flo EV Charging, FlexCharging, wallbox, eo, HUBJECT, ev.energy, SIEMENS energy, CIRCLE K, go eve, KiWiPOWER demand management, true a Landis+Gyr company, GENERAC, Trilliant, IoTecha

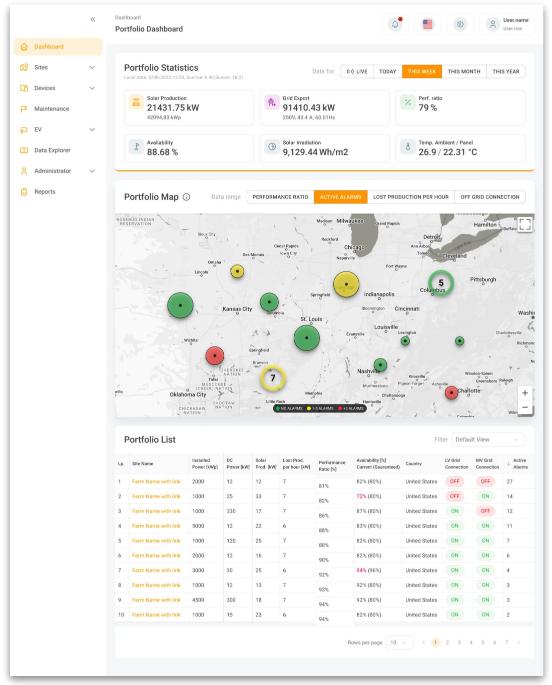
codibly.com

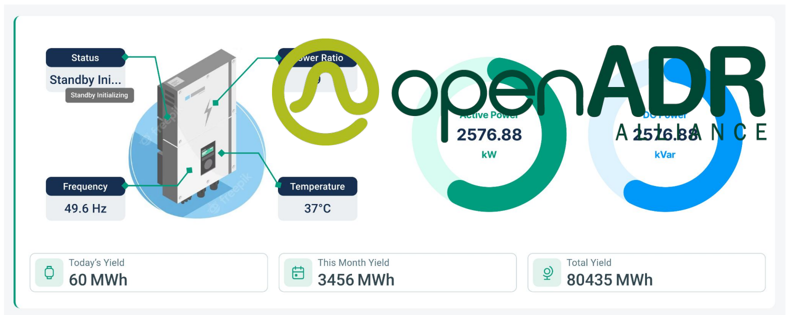
Client: Large North American Utility

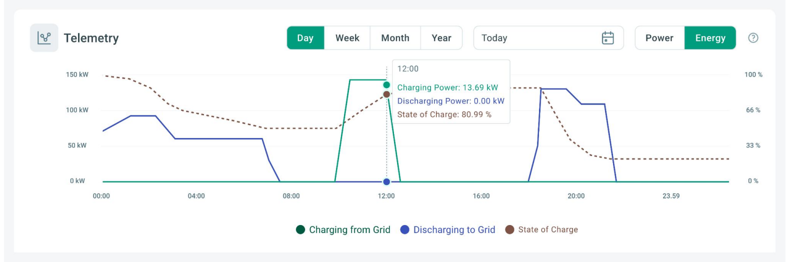
All-in-one solution for efficient management of utility scale storage systems for prosumers, installers and distributors.

The whole ecosystem enables energy costs and usage optimization, battery and charger management, communication with grid services, smart charging and storage of generated energy based on utility tariffs to enable demand response.

- Grid and Device Level Integrations
- Dynamic Load Management





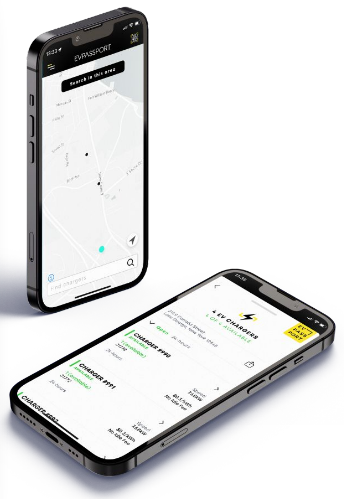


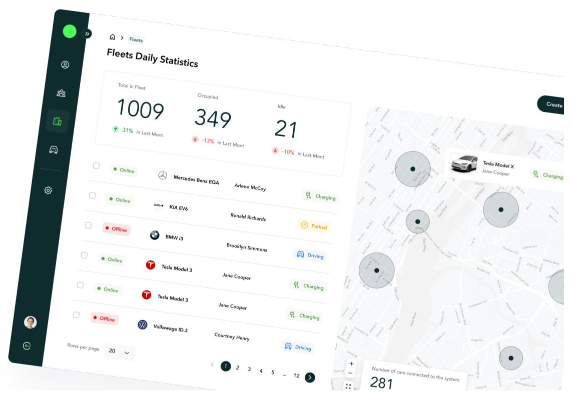
Client: Leading German Vehicle Manufacturer

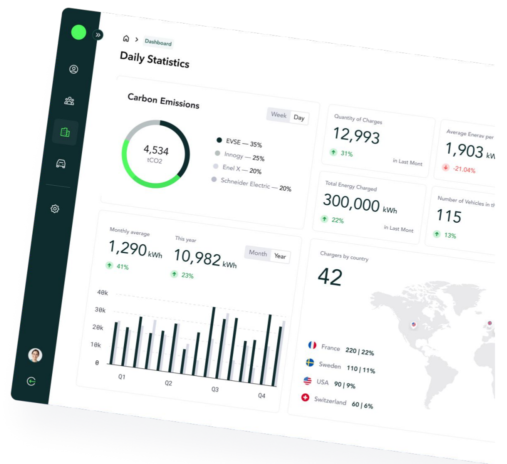
The aim was to manage and monitor EV charging stations from any manufacturer in one digital platform. This includes overseeing operations, diagnosing and resolving remote issues, and managing tariffs.

The outcome was a platform and mobile app that allowed for remote setup and monitoring, full dynamic load balancing, flexible scheduling, real-time data insights, and payment system

- Real-time Insights for Customers
- Vendor Agnostic Charge Point Operator Platform





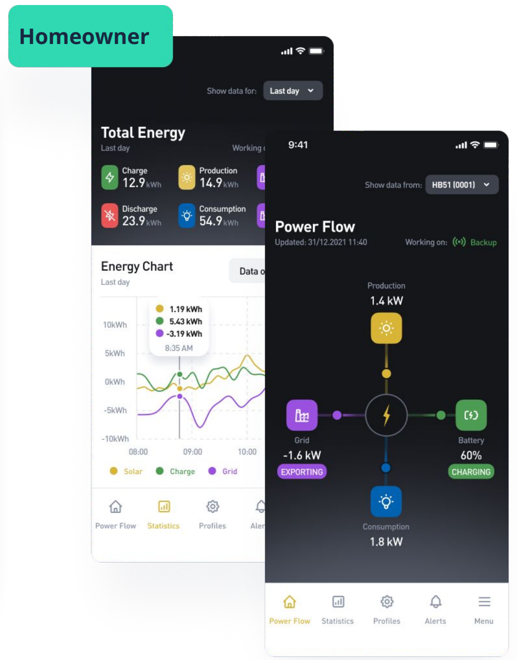


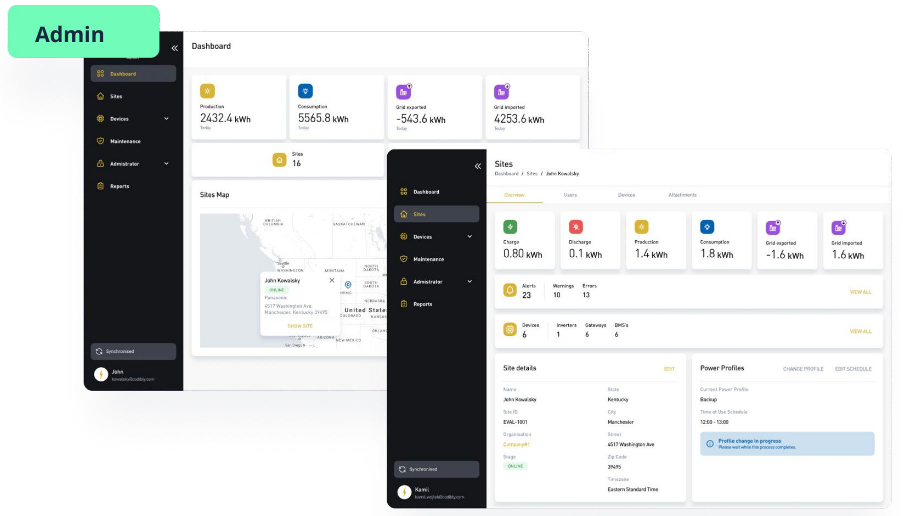
Client: Leading Japanese Electronics & Device Manufacturer

Platform to control remote charging stations and maintain a specific load level set by the utility. Manages energy demand response (DR) programs and efficiently adjusts loads across multiple charging points. Supports various types of EVSEs and provides a scalable, secure platform that can handle different devices with customizations to meet system requirements.

Implemented OCPP 2.01, MODBUS, and ChargePoint API.

- Multi-tenancy
- Interoperability & Scalability





codibly | part of spyrosoft

openADR

We use wide range of technologies

- We are experienced **technology experts**
- We use over **100 technologies and tools,** always choosing what's best for the client
- We constantly improve **our competencies**
- We do not limit ourselves, and we are eager to **explore new technology areas**
- We follow **technological trends** and apply **best practices**

spring · node · mongoDB · SQL Server · MySQL · ANGULAR · Flutter · Kotlin · Swift · Xamarin · Azure · Looker · Google Cloud · Google Big Query

JS · TS · LabVIEW · MATLAB · LINUX · Windows · mac OS · iOS · android · Java · Jira Service Management · k6 · Gatling

Selenium · cypress · DEFENSIVE SECURITY · Ansys · VECTOR · AUTOSAR · ENTERPRISE ARCHITECT · ROS · 2 · JMeter · hadoop · NVIDIA · Magento · shopware

QNX SOFTWARE SYSTEMS · WiFi · 5G · RFID · LTE · aws · slint · Qt · unity · ZABBIX · ITIL · Prometheus · Grafana · appian · outsystems · docker · kubernetes

UNREAL ENGINE · androidauto · specflow · new relic · tableau · Jenkins · PC · TC · circleci · Terraform · ANSIBLE · Opsgenie · Bitbucket · snowflake

salesforce · cloudcraze · AWS CodeCommit · Amazon Athena · amazon REDSHIFT · Azure Functions · GITLAB CI/CD · GitLab · Digital Ocean · databricks · neo4j

Nagios · SALTSTACK · PostgreSQL · elastic stack · ORACLE · Apache Airflow · Spark · Qlik · redis · Google Analytics · QuickSight · elasticsearch · openstack

kafka · ArcGIS · PostGIS · mapbox · GeoServer · MapServer · QGIS · Leaflet · OPENSHIFT · Scala · DVC · plotly · dask · pandas

Standards and their role

codibly | part of spyrosoft

Standards and their role

i. **Interoperability in the energy and EV sectors**
   1. Enable seamless communication between diverse hardware and software platforms.
   2. Foster collaboration between different manufacturers, service providers, and grid operators.

ii. **Ensure seamless communication between different devices, systems, and platforms.**
   1. Allow systems to expand with ease, supporting more devices, users, or services without requiring complete redesigns.
   2. Facilitate large-scale deployments, such as smart grids and national EV charging networks.

iii. **Reduce vendor lock-in and foster competition.**

iv. **Enable scalability in energy sector**
   1. Promote innovation by providing a common foundation for developers and stakeholders.
   2. Enable startups and smaller companies to participate without needing proprietary infrastructure.

v. **Cost Efficiency:**
   1. Reduce development and integration costs by avoiding proprietary solutions.
   2. Streamline the deployment of new technologies by leveraging pre-defined frameworks.

vi. **Faster Time-to-Market:**
   1. Standardized protocols simplify development and testing, accelerating product releases.
   2. Reduce friction in regulatory approval processes by adhering to industry norms.

vii. **Security:**
   1. Many standards include robust security protocols, protecting systems against data breaches and unauthorized access.

codibly | part of spyrosoft

Architecture design of different standards

codibly | part of spyrosoft

Comparison of Standards

Standard	Primary Use Case	Architecture Model	Communication Protocols	Complexity	Scalability	Integration Flexibility	Security Features	Developer Considerations
OCPP	EV charger management	Client-Server	WebSocket, HTTP(S)	Moderate	High	Medium	(none, Password,mTLS)	Requires middleware for non-OCPP chargers. Open source SDKs, cloud base accelerators available
OCPI	Roaming for EV networks	Decentralized	REST, JSON	Low-Moderate	High	Medium	(token-based)	Lightweight API-based integration. Ideal for roaming networks.
OpenADR	Demand response programs	Hub-and-Spoke	HTTP, XML, JSON	Moderate	High	High	(mTLS, OAuth2)	Optimized for demand response. Requires Business Logic for demand response programs.
OSCP	Power load of a network of charging stations	Client-Server	REST, JSON	Low-Moderate	High	High	(TLS)	Simple to implement. Needs grid operator cooperation.
ISO 15118	Plug-and-charge functionality	Peer-to-Peer	TLS, XML, V2G-TP	High	Moderate	Low	(mTLS with PKI)	Complex PKI management. Essential for future EV ecosystems.
IEEE 2030.5	DER and IoT integration	Hub-and-Spoke	HTTP(S), REST, XML	High	High	Low	(mTLS)	Low level specification and requirements. Less flexible than OpenADR. Can be also implemented for DR.
Modbus	Legacy and industrial systems	Master-Slave	TCP/IP, Serial	Low	Moderate	Low	Basic	Simple and robust. Best for constrained or legacy systems.

1. Architecture Model

- **OCPP** Client-server models, suitable for centralized management of devices (EV chargers).
- **ISO 15118**: Peer-to-peer model, ideal for direct communication between vehicles and chargers.
- **OpenADR and IEEE 2030.5**: Hub-and-spoke architectures enable centralized data aggregation and control for distributed systems.

2. Communication Protocols

- **Modern Protocols**:
  - **OCPP, ISO 15118, OpenADR**: Use secure, internet-based communication protocols (HTTP, WebSocket, TLS).
- **Legacy Protocols**:
  - **Modbus**: Include specialized or legacy protocols like GOOSE, SMV, or serial communications

3. Complexity

- **High Complexity**:
  - **ISO 15118, IEEE 2030.5**: Involves intricate processes like Public Key Infrastructure (PKI) and certificate exchange.
- **Moderate Complexity**:
  - **OCPP, OpenADR**: Relatively simpler implementations but require understanding of API development and endpoint integration.

4. Scalability

- **High Scalability**:
  - **OCPP**: Supports large EV charging networks.
  - **OpenADR, IEEE 2030.5**: Scalable for managing DERs and demand response across multiple locations.
- **Moderate Scalability**:
  - **ISO 15118**: Limited scalability due to the peer-to-peer nature.
  - **Modbus**: Best suited for small-scale, legacy applications.

5. Integration Flexibility

- **High Flexibility**:
  - **OCPP, OpenADR, IEEE 2030.5**: Easily integrated with other systems using REST APIs or middleware.
- **Moderate Flexibility**:
  - **IEEE 2030.5**: WLow level specification and requirements. Less flexible than OpenADR. Can be also implemented for DR.

codibly.com

Layers

codibly | part of spyrosoft

openADR ALLIANCE

System Example: Multi-Standard EV Charging Ecosystem

- **Physical Layer**: **(Hardware Level)**

  Purpose: Interfaces hardware devices such as EV chargers, smart meters, DERs, or substations with communication systems.

- **Communication Layer: (Transport and Connectivity)**

  Purpose: Ensures reliable and secure data transfer between devices, systems, and cloud platforms.

- **Application Layer**: **(Data and Protocol Handling)**

  Purpose: Manages device-specific protocols, application-specific logic, and standards implementations.

- **Middleware Layer: (Abstraction and Interoperability)**

  Purpose: Decouples the hardware and software components to simplify integration of multiple standards.

- **Cloud/Control Layer**: **(Management and Orchestration)**

  Purpose: Ensures data integrity, user authentication, and secure communication.

- **Security Layer**
  - **Purpose**: Ensures user authentication, and secure communication.

codibly | part of spyrosoft

Approaches to Implementing Standards

codibly | part of spyrosoft

openADR ALLIANCE

In-house vs Outsourcing dilemma

codibly | part of spyrosoft

openADR ALLIANCE

Aspect	In-House Development	External Provider (Outsourced)
Cost	- High initial investment (salaries, tools, infrastructure)	- Predictable cost, pay-per-project or service-based pricing
Expertise	- Limited to in-house skills, may require training	- Access to specialized expertise and industry best practices
Speed of Delivery	- Potential delays due to competing internal priorities	- Faster execution due to focused project teams
Scalability	- Requires hiring and onboarding for additional capacity	- Easily scalable based on project needs
Control & Flexibility	- Full control over process and customization	- Limited control, dependent on contract terms
Security & Compliance	- Easier to ensure adherence to internal standards	- Must carefully vet provider for compliance and security
Maintenance	- Continuous responsibility on internal teams	- Provider may offer ongoing support as part of the contract
Innovation Potential	- May be constrained by existing knowledge	- Exposure to new technologies and solutions

In-house vs Outsourcing dilemma

codibly | part of spyrosoft

Implementation models

codibly.com

1. **Build from scratch**
2. **Open Source**
3. **Proxy**
4. **Accelerated**

Build from scratch

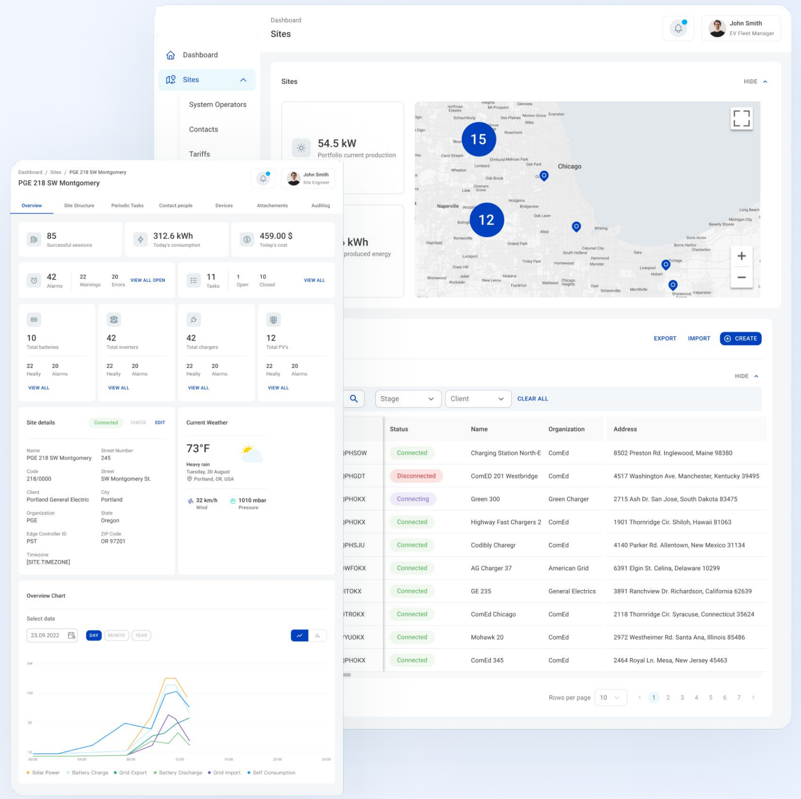
Develops a fully customized solution to implement the standard without relying on open source or ready to implement solutions. Often used by organizations with unique requirements or proprietary systems where solution can bring comptetitive advantage and largely base on company’s know-how.

PROS

- **Full Control**:
  - Tailor the implementation to specific business needs.
  - Enables optimization for performance and integration.
- **Flexibility**:
  - Can adapt the implementation to future requirements without dependencies.
- **Long-Term Viability**:
  - No reliance on third-party code, which may become obsolete.
- **IPR**
  - Full IP Rights for the code and solution

CONS

- **High Cost**:
  - Requires significant investment in resources, including skilled developers and testing.
- **Long Development Cycles**:
  - Building from scratch can delay time-to-market compared to other approaches.
- **Steep Learning Curve**:
  - Teams must gain expertise in the standard’s intricacies before implementation.
  - Requires in-depth knowledge of the standard and its specifications.
- **Maintenance Overhead**:
  - Responsibility for bug fixes, updates, and compliance remains entirely in-house.



codibly | part of spyrosoft

Leverage Open Source Solutions

Use open-source libraries, tools, or reference implementations to implement the standard.
Used by companies for the solutions that have strong community support and has access to knowledge.

codibly.com

PROS

- **Accelerated Development**:
  - Leverage pre-existing implementations to reduce coding and testing time.
- **Cost Efficiency**:
  - Typically free to use, reducing licensing costs.
- **Community Support**:
  - Access to a global community for troubleshooting and enhancements.
- **Focus on Customization**:
  - Developers can focus on customizing the implementation instead of building core functionality.

CONS

- **Limited Flexibility**:
  - Restricted by the architecture and features of the open-source solution.
- **Dependency Risk**:
  - Relying on community updates may lead to delays or project risks if the library is abandoned.
- **Performance Constraints**:
  - Open-source solutions may not be optimized for specific business needs, leading to inefficiencies.
- **Security and Compliance Risks**:
  - May require extensive audits to ensure the library adheres to security and regulatory standards.
- **Certification process**
  - Needs to be conducted by internal team with no support

codibly | part of spyrosoft

Using Standards as a Proxy

Utilize middleware or adapters provided by external suppliers. The proxy layer acts as an intermediary, reducing the need to implement the standard directly. Usually provided as SaaS.

PROS

- **Simplified Integration**:
  - Avoids the need for direct implementation by bridging existing systems with the standard.
- **Fast Deployment**:
  - Ideal for organizations that need quick compliance without full implementation.
- **Cost Efficiency short term**:
  - Reduces development effort by leveraging system as a service.

CONS

- **Performance Trade-Offs**:
  - Communication may introduce latency or reduce system performance.
- **Limited Control**:
  - Developers usually don't have flexibility to optimize the proxy solution.
  - No access to source code.
- **Dependency Risk**:
  - Reliance on third-party services introduces risks if the provider discontinues support.
- **Access to Data**
  - Service provider will have full access to data.
- **Cost long-term**
  - For larger implementations overal cost may be higher than other options.

Accelerated Development

Utilize commercial Accelerator that is pre-built for implementing standards. Accelerators typically offer configurable options, allowing for quicker implementation with some customization.

PROS

- **Short Time-to-Market**:
  - Faster than building from scratch and open source but still allows for customization.
- **High Reliability**:
  - Accelerators often come with tested and validated implementations.
- **Support Options**:
  - Implementation may include dedicated technical support.
  - No need for long learning curve.
  - Supplier usually provides verion updates with new functionalities or security fixes.
- **Reduced Risk**:
  - Fewer bugs and issues as the framework is built by experienced providers.
  - Already tested by other companies.
  - Often Suppliers allow for contacts with other clients.
  - Supplier takes responsibility for the implementation.
- **Certification**
  - Vendors can help with certification process that will speed up and simplify it.

CONS

- **May be costlier than open source**:
  - Licensing or subscription fees increase the overall implementation cost.
- **Predefined Stack**:
  - Technology stack may be different than stack that internal team is using.
- **Quality and Security**:
  - Implementing pre-built code may require audit to ensure it meets quality and security standards.
- **Integration Complexity**:
  - May require effort to align the accelerators with existing systems or above mentioned standards.

	With ACCELERATORS	CUSTOM BUILD
OCPP 1.6J + 2.0.1	4-8 weeks	4-12 months
OCPI	4-8 weeks	3-4 months
OpenADR	4-8 weeks	3-4 months

codibly | part of spyrosoft

Factors to Consider in decision making process

- **Project Timeline**:

  Tight deadlines favor open source, accelerators, or proxies

- **Budget**:

  Cost-sensitive projects may benefit from open-source or proxy solutions

- **Business Continuity:**

  Critical business solutions will favor solutions that are robust and Company can further support and develop the solution (no vendor-lock situation)

- **Scale and Future Growth**:

  Large-scale deployments may benefit from accelerators or custom build to ensure scalability

- **Complexity and Customization Needs**:

  Highly customized requirements may necessitate building from scratch or leveraging accelerators

- **Expertise and Resources**:

  Teams with limited expertise in the standard may prefer accelerator-based solutions

codibly | part of spyrosoft

openADR ALLIANCE

Models comparison

Approach	Time-to-Market	Cost	Business Continuity	Flexibility	Scalability	Complexity
Build from Scratch	Long	High	High	High	High	High
Open Source	Medium	Low-Medium	Medium	Medium	Medium	Medium
Accelerators	Short	Medium	High	Medium	High	Low
Proxy	Short	Low-Medium	Low	Low	High	Low

codibly.com

codibly | part of spyrosoft

CASE STUDY QUIZ

codibly | part of spyrosoft

CASE STUDY 1

Company 1 - EVSE startup

Recently established with revolutionary idea for EV market. The team consist of hardware and embedded engineers with limited knowledge of cloud software development and standards. They need to implement OCPP and OpenADR to participate in NEVI (National Electric Vehicle Infrastructure) program in Alaska state - US.

1. Limited knowledge and resources
2. Team focused on internal know-how that can assure competitiveness on the market
3. Important time-to-market (very often connected with investors and their expectations)
4. Needs to fulfill compliance with standards but this is not business critical

Solution ?

codibly | part of spyrosoft

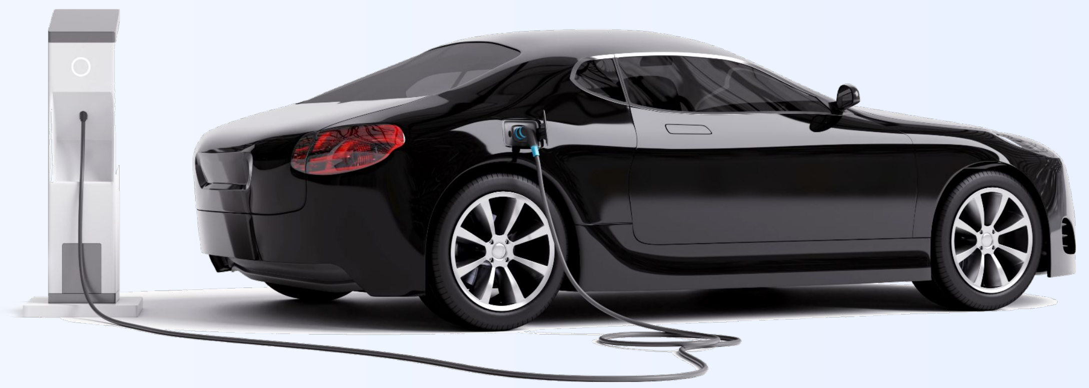
CASE STUDY 2

Company 2 - Large Enterprise

Car OEM that provides wallboxes to their clients. Want to provide Demand Response programs for their clients.
The department has KPI to implement OpenADR standard within 6 months.

1. Large development teams
2. Complex architecture
3. Security and quality is critical
4. Compliance with the standards is important for revenue
5. Time-to-market (short to mid term) / depending on particular KPIs

Solution ?



codibly.com

CASE STUDY 3

Company 3 - Scaleup

Charge Point Management System provider needs to implement OCPP server 2.0.1. to secure contract with large client. They already have Their implementation of OCPP 1.6. Due to fast growth of the company and large competition on the market the Roadmap is fully packed with additional features that can help growing the system.

1. Medium size development team
2. Team usually focused on uniqe functionalities
3. Extensive roadmap where priorities are must due to limited resources
4. Compliance with the standards is important for securing contracts with clients
5. Time-to-market (short to mid term)

Solution ?

Which stack should I use?

Agnostic approach

codibly | part of spyrosoft

Why Standard Integration Doesn't Have to Use the Same Technology Stack as other systems

A. Standards Are Protocol-Based, Not Stack-Dependent

- Standards like OpenADR, OCPP, and OCPI define communication protocols (e.g., REST, JSON, WebSocket), not implementation technologies.
- These protocols are interoperable across diverse technology stacks, making direct stack alignment unnecessary.

B. Enables Use of the Best Tools for the Job

- Different standards may have libraries, SDKs, or tools optimized for specific stacks.

D. Facilitates Microservices and Decoupled Architectures

- Standards can be implemented as microservices, each in its preferred stack, enabling independent scaling and updates.
- **Example**: A Node.js-based OCPI microservice can integrate seamlessly with a Java-based billing system.

E. Encourages Innovation and Flexibility

- Teams can experiment with cutting-edge technologies or frameworks for standard implementation without affecting the core stack.
- **Example**: Implementing ISO 15118 using Go for high performance while retaining the legacy system in .NET.

F. Supports Heterogeneous Environments

- Companies often have diverse systems due to acquisitions, partnerships, or legacy infrastructure.
- **Example**: An energy company may use .NET for its CRM but JAVA for OpenADR-based DER integration due to pre-existing expertise or tools.

G. Simplifies Outsourcing

- Outsourced development of a standard can occur in a stack optimized for that standard, independent of the internal systems.
- **Example**: A third-party vendor may implement OCPP in Python while the company maintains its core applications in C#.

Ensuring a Stack-Agnostic Approach for Implementing Standards

A stack-agnostic approach enables companies to integrate standards like OpenADR, OCPP, or OCPI without being tied to the same technology stack as their existing solutions. This approach promotes flexibility, scalability, and long-term adaptability. Here's how to achieve it and reasons why standards don't need to be implemented in the same stack as other company solutions.

A. Adopt Modular Architecture

- Separate the implementation of the standard from other system components (e.g., user management, billing, analytics).
- **Example**: Deploy OpenADR as a standalone microservice that interacts with the rest of the system through REST APIs or event-driven architecture.

B. Utilize Containerization

- Implement the standard in isolated containers (e.g., Docker) that can run independently of the underlying stack.
- **Example**: A containerized OCPP service can communicate with a .NET backend-CPMS.

C. Leverage Standard Communication Protocols

- Standards often rely on universal communication protocols like HTTP, REST, JSON, or WebSocket, enabling seamless interaction between heterogeneous stacks.
- **Example**: OpenADR uses XML and JSON payloads that can be processed by any system, regardless of the technology stack.

D. Adopt Cloud-Native Approaches

- Deploy the standard implementation on cloud platforms using serverless or platform-agnostic services (e.g., AWS Lambda, Azure Functions).
- **Example**: Use OpenADR in a cloud-based VTN while integrating it with an on-premises DER management system.

E. Focus on Abstraction Layers

- Create abstraction layers that encapsulate standard-specific logic, ensuring that the core business logic is unaffected by the choice of implementation stack.
- **Example**: A Java-based abstraction layer for ISO 15118 can interact with Python-based EV charging systems.

Most common Questions from our Clients

codibly | part of spyrosoft

openADR ALLIANCE

We need to own IPR.

codibly.com

codibly | part of spyrosoft

I have .NET as main technology. How can I implement your solution that is written in JAVA?

How to assure independency from the service provider.

codibly | part of spyrosoft

Who will maintain the solution after deployment?

codibly.com

CASE STUDY EVSE x DEMAND RESPONSE

codibly | part of spyrosoft

Case Study from 2024

Implementation of the OpenADR standard and integration with one of the aggregators.

EVSE NORTH AMERICA

Leading EVSE from North America that needs to achieve two goals that will help business growth.

Goals

1. Compliance with OpenADR and certification to meet NEVI program requirements
2. Integration with Aggregator to participate in DR programs for additional revenue.

SOLUTION

EVSE NORTH AMERICA

Chose the OpenADR accelerator ensuring that:

1. there is full knowledge transfer to the internal dev team
2. can be independent from vendor provider
3. can be OpenADR certified within max 2 months.
4. will participate in DR Program starting from new year (participation is based on quarterly intervals) (3 months from project start)
5. the solution can be integrated with this current tech stacks (PHP, .NET)

EVSE EUROPE

Leading EVSE from EU. Want to participate in Demand Response Programs. They calculated that on yearly basis it can bring them around X Mio EUR of additional revenue.

Goals

1. Integration with Aggregator to participate in DR programs for additional revenue.

SOLUTION

EVSE EUROPE

Chose the custom developed solution from scratch based on fixed price model ensuring that:

1. solution will be delivered on time within exact timeframe
2. will participate in DR Program starting from new year (participation is based on quarterly intervals)
3. can be independent from vendor provider and their team can take over the solution at any time needed
4. SLA responsibility will be transferred to external vendor
5. the solution will be developed in particular tech stack (in this example node.js)

codibly.com

codibly | part of spyrosoft

Case Study from 2024

Comparison of the results

	Time-to-Market	Cost	Scope	Knowledge transfer	Scalability	Testing	IPR
EVSE NA	Short	Medium	OpenADR cert + Integration	Full	High	Short	Vendor / Client
EVSE EU	Medium	Medium-High	Integration	Full	High	Medium-Long	Client

codibly.com

Questions?

Łukasz Kulczyński
Executive Vice President | Head of eMobility

+48 538 050 158
lukasz.kulczynski@codibly.com
Linkedin