Considering extensions to enumeration tables using Schema files

Written by: David Herron - Evoke Systems - david@davidherron.com or david.herron@evokesystems.com

Date: February 1, 2025

Introduction

An important hallmark of the OpenADR 3.x protocol is flexibility. Rather than a rigid attitude of "THERE IS ONLY ONE TRUE WAY TO USE OPENADR", the realization is that many times demand/response programs are customized.

In Definition.md, the principle is described this way:

> **Extensibility**: The OpenADR 3.0 protocol allows servers and clients to interoperate without custom integration. It is intended to provide a functional footprint that is sufficient to accommodate all common demand response use cases. However, some demand response program developers may find it useful to use content that cannot be expressed using the constructs of the specification, or could be expressed in a better form with an extension.There are two extension mechanisms offered by OpenADR 3.0: model extensions, and private strings.

Expressing the Enumeration tables in JSON Schema format allows the OpenADR ecosystem an additional avenue of extension.

Namely, an OpenADR program administrator may want to add new endpoints, add new fields to schema objects, or add new values to an enumeration table. The result is a program that has extended OpenADR in some way, while retaining the essential nature of OpenADR.

These types of extensions require modifications to

1. The OpenADR 3.x YAML specification.

2. The corresponding enumeration schema file, or creating a new enumeration schema file.

What would constitute an OpenADR extension?

Since the OpenADR protocol doesn't cover everything, we expect there's a need for guidance on how to proceed with a private extension.

It seems the most likely route is to extend the enumerations tables. For example, to support additional report data, or support additional VEN or Resource attributes.

To get a sense of this look at the EV Charging Use Data Specification. It's a set of JSON Schema files corresponding to a specification created by a group of electric utility, EV charging, and government stakeholders that tries to be a commonly accepted standard for reporting EV charging data. It covers EV charging sessions, outage periods, the hardware at EV charging locations, and even the funding sources. Most of the tables they describe can be communicated by a VEN to a VTN using VEN/Resource attributes and report payloads.

No additional API endpoints are required, nor are any additional schema objects. The EVCUD specification is solely about data reporting.

How should a group of collaborators proceed in extending OpenADR?

First, there should be a group of stakeholders. The EVCUD project involved a dozen organizations. Ideally they would reach out to the OpenADR community for participation. For the result to be an extension means that their needs are unique enough the broader OpenADR community doesn't want to incorporate the extension.

The stakeholders should produce

- *Specifications* - For the stakeholders to be on the same page, they must, depending on their needs, modify the OpenADR YAML specification, create additional OpenAPI specifications, add new enumeration schema entries, or create new enumeration schema files.
- *Documentation* - Just as the Definitions and User Guide assist with explaining OpenADR, the stakeholders must explain their extensions to OpenADR.
- *Testing* - Enforcing compliance with the extended OpenADR should require a testing program. The extent of these tests is to be determined by the stakeholders.
- *Technical committee* - This is a team of people who will discuss and document the extension, producing the above assets.

These extensions must be shared between the stakeholders. All must implement VENs and VTNs based on the extensions.

Extensions described as modifications to the concrete specifications make it easier for stakeholders to be on the same page.

Once an OpenADR extension has proved its worth, the stakeholders might submit it to the OpenADR team for inclusion.