Open Supporter Data Interface Charter

1 Overview

The Open Supporter Data Interface (OSDI) is an effort to reduce customer costs related to moving data between different systems by defining a common API for products in the non-profit and campaign space. We are a team of vendors and customers who are prioritizing customer cost reduction and interoperability.

2 Problem Space

Today, customers often seek to use a variety of digital tools from different vendors to build their optimal solution. Systems such as CRMs, email blasters, donation management systems, social media tools, voter engagement tools, and volunteer management tools may come from different vendors. However, in order to keep the data consistent, customers often need to do frequent manual imports and exports of data via mechanisms such as CSV files. Sometimes options are unavailable or are so complex that the systems remain inconsistent and valuable data are lost.

Systems typically contain a common set of resources, including, but not limited to people (supporters), addresses, donations, events, or social actions. For example, each product typically represents a person differently. How addresses are handled varies from system to system and in some cases, even the field names are different.

There is no competitive advantage for vendors to model a person differently. The difference merely serves as a cost to customers in the form of added complexity, data loss during transfer, and extra staff & volunteer time.

3 Goal

3.1 Summary

OSDI aims to define a common commodity API interface for these resources. The API will cover the most common customer use cases.

The API will define interfaces including by not limited to resources representing people, donations, questions, tags, and events.

The group will determine the order in which to define resource models and which version of the API to include them in.

3.2 Maintain Customer Focus

The OSDI team will solicit feedback from customers to review use cases and technical designs to make sure that we are building something that is cost-effective to implement and solves the problems customers care about.

3.3 Allow Vendor Differentiation

The core API will also allow for proprietary extensions to be built on top of it. These extensions may represent vendor or party specific features, innovations, or differentiations. They may also be special purpose features that are not relevant to the wider market. They may be ideas where it is too early for industry-wide consensus to form. Over time, these may be integrated into the common API.

By using such a layered model, customer integration costs are reduced by ensuring that as much common code may be reused as possible. For resources defined by the core there should be very little differential integration code necessary to work with products from different vendors. Even when using a vendor or party specific feature extension, the core elements and concepts can be reused, leaving only the extension as conditional code.

3.4 Accelerate Innovation

The common API can also accelerate innovation in multiple ways, including:

1. Reducing duplicate work by each vendor. Vendors can focus on new customer features rather than solving the same problems over and over again.
2. Providing a common application platform that allows entrepreneurs, startups, and consultancies to build applications that can run across all vendor platforms that support OSDI rather than having to spend development resources writing individual connectors for each vendor platform.

4 Current Deliverables

4.1 Requirements

Requirements document which outlines the use cases (i.e., user stories) and resource data models. Use cases or user stories are examples of tasks customers need to do. Example: Query new supporter signups from today that wish to volunteer.

4.2 V1 API Specification

Resource data models defining the fields and operations of a given object. Example: A Person resource has fields like first_name, last_name, ZIP, etc.

API specification which defines the common operations and resources to allow reading, updating and querying.

4.3 Prototype

A prototype implementation useful for customers to experiment with, exercise scenarios, and prototype client implementations.