Designing an API for the Physical World

October 5th, 2023

Sy Bohy, CEO at Seam

Presenter

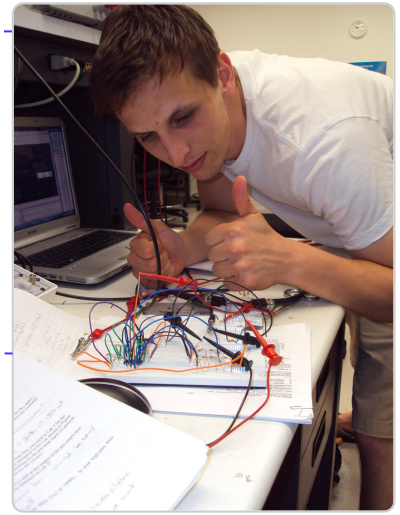
About Sy

Bio

Grew up in the French Alps. Studied Materials Science, Electrical Engineering, and Computer Science at Stanford University. Early Nest engineer. Now based in San Francisco, CA, and CEO at Seam.

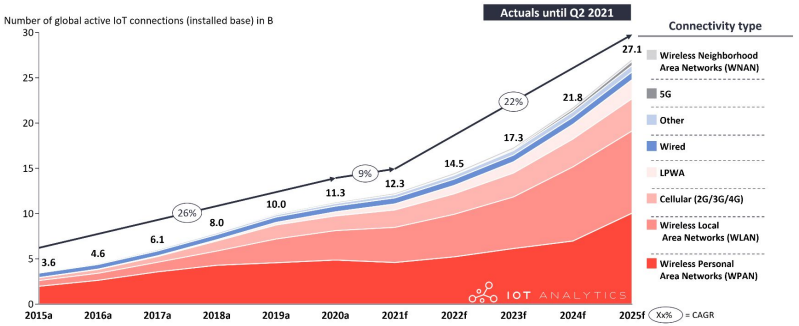
Interests

The future can be fantastic; but it has to be invented first.



Context

The world currently deploys 2B+ net new IoT devices/year



Global growth rate of connected IoT devices through 2025

Opportunity

Every time an important resource was difficult to tap into, an API company arose

Payments → stripe

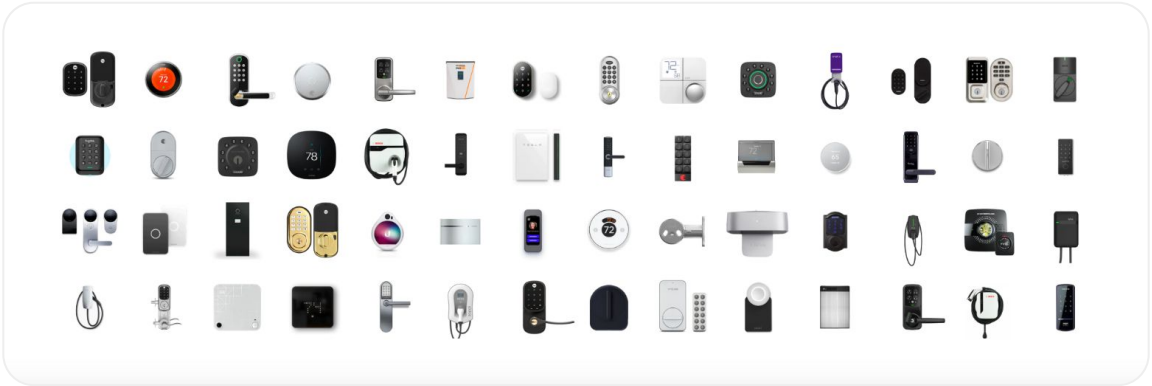
Banking → PLAID

Telco → twilio

Mission

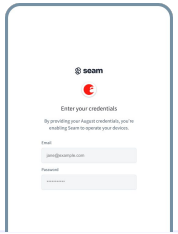
Build the API to the Physical World

Seam builds digital and physical infrastructure to enable softwares developers and businesses to connect their applications with the physical devices around us.



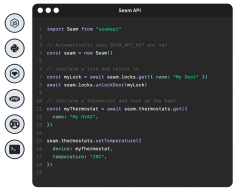
Product

Seam provides everything needed to make device integrations quick & easy



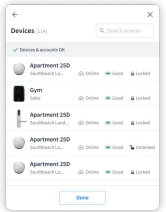
Authorization Flows

Seam provides pre-built authorization flows that handle brand selection, account login, 2FA, and device retrieval.



Standardized APIs

Seam's API and SDKs standardize device functions across brands to simplify workflow integration.



UI Components

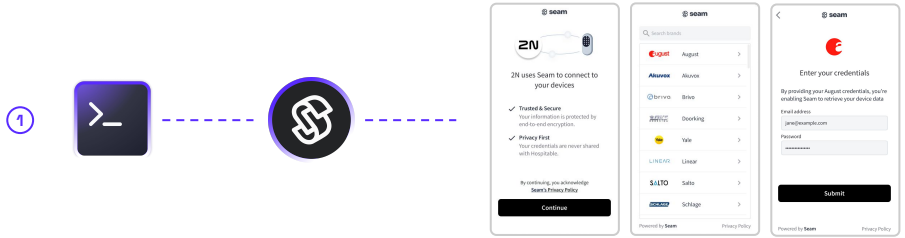
Developers can use Seam's library of pre-built UI components to offer advanced device management functionality.

Product

How it Works

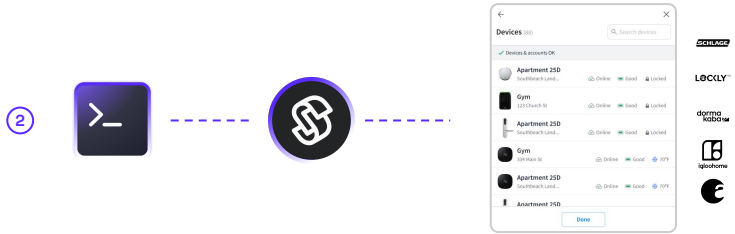
1 – Link Device Accounts

Present a Seam auth flow to your app users for them to link their device accounts with your application.



2 – Retrieve Devices

Once accounts are connected, use the Seam API to retrieve devices across brands. Use Seam Components to offer users advanced device management UI or build your own using the API endpoints provided by Seam.



3 – Control Devices

Seam standardizes all device functions behind a single API. This lets you unlock doors, check on battery levels, create unique credentials, and more, all without having to worry about the underlying implementations.

Shared Vision

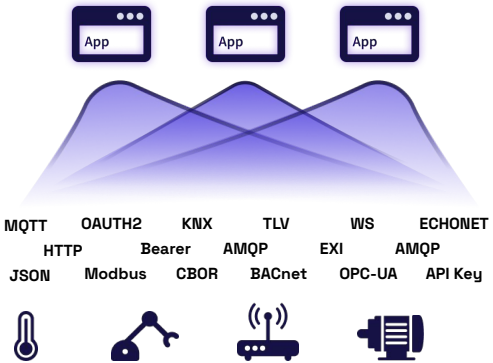
The W3C WoT and Seam have the same mission—just stated differently

WoT Mission

"The WoT provides a set of standardized technology building blocks that help to simplify IoT application development […] WoT **unlocks commercial potential being held back by IoT fragmentation**"

Seam Mission

"Seam builds digital and physical infrastructure to enable software developers and businesses **to connect their applications with the physical devices around us**."

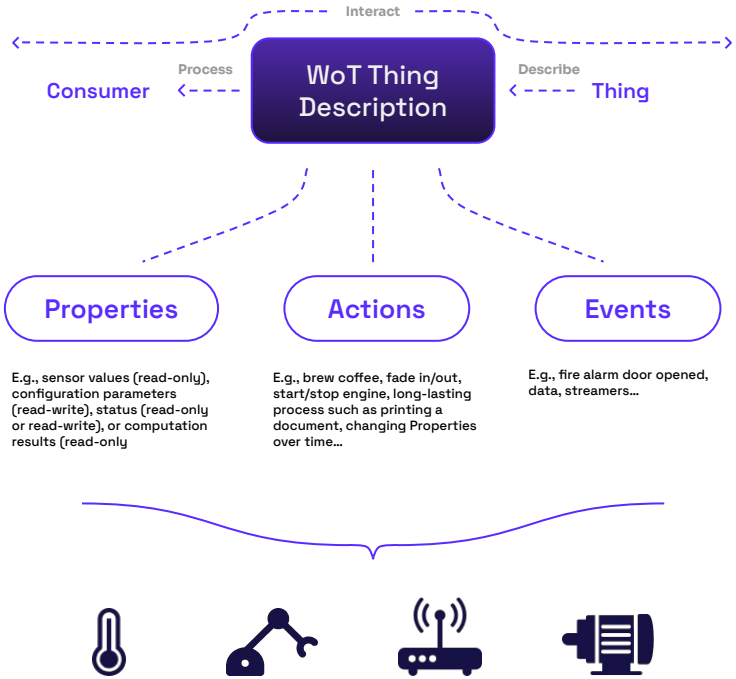


Background

Thing Description (“TD”) is a powerful concept

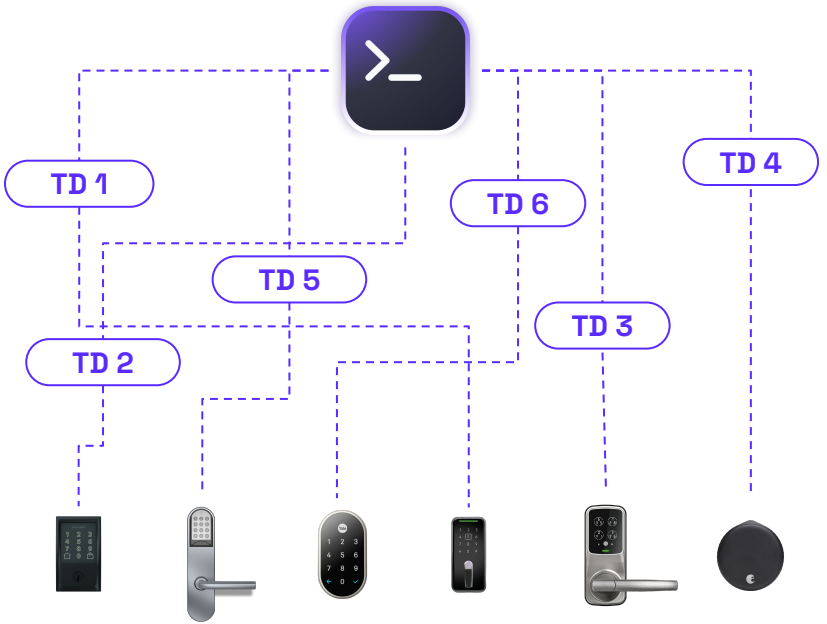
Things Description

“The WoT Thing Description (TD) is a standardized, machine-readable metadata representation format that **allows Consumers to discover and interpret the capabilities of a Thing** (through semantic annotations) and to adapt to different implementations (e.g., different protocols or data structures) when interacting with a Thing, thereby enabling interoperability across different IoT platforms, i.e., different ecosystems and standards.”



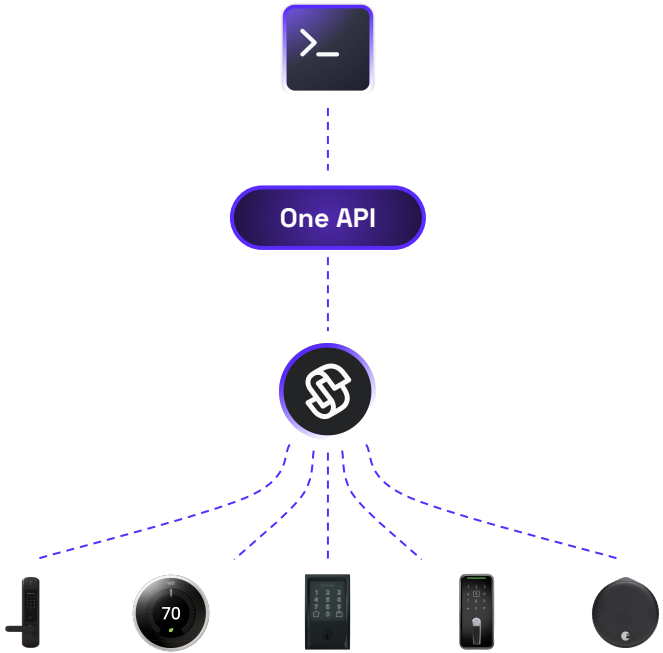
Limitation

TD does not solve the fragmentation problem app developers still face



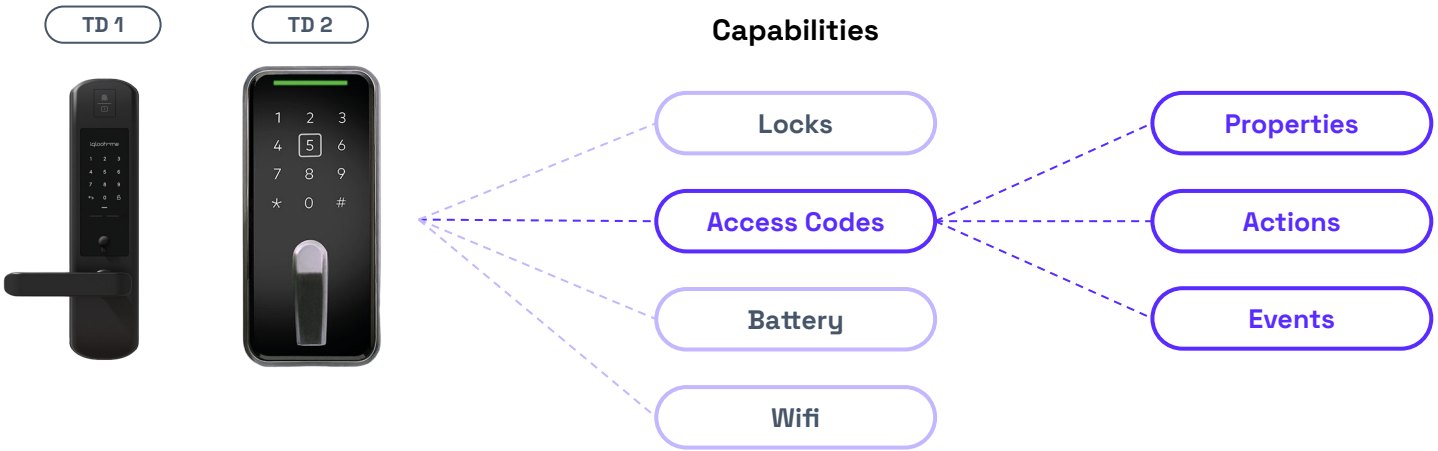
Solution

What app developers want is a standardization layer to simplify integration



Capabilities

Standardizing across brands requires capability interfaces to encapsulate common device features



Capability:

1. Ben Francis, Mozilla WebThings
2. Mozilla WebThings, Capability Schema
3. SmartThings, Developer Docs
4. Z-Wave Alliance, Command Classes

Outcome

Standard capabilities lets you write code that works across brands

```
# Unlock door on Salto KS
seam.locks.unlockDoor(deviceId)

# create code on Yale lock
seam.accessCodes.create({
  device_id: deviceId,
  name: "my ongoing code"
})

```

Limitations

Of course, some device brands still have hard limitations

```
# Try to set a code with a leading 0
seam.accessCodes.create({
  device_id: deviceId,
  name: "my ongoing code"
  code: "0248"
})

=> ERROR - code can't start with "0"

```

Document

Documenting those rules can help guide implementation logic

```
{
   "device":{
      "device_id":"525547360b9b",
      "capabilities_supported":[ "access_code", "lock" ],
      "properties":{
         "supported_code_lengths":[4, 5, 6, 7, 8],
         "max_active_codes_supported":250,
         "code_constraints":[
            {
               "constraint_type":"no_zeros"
            },
            {
               "constraint_type":"name_length",
               "min_length":1,
               "max_length":12
            }
         ],
…
}
```

DevUX Focus

And you can hide those brand limitations with various abstractions

```
# Create a code that preferably matches
# last 4 digits of user phone number
seam.accessCodes.create({
  device_id: deviceId,
  name: "Airbnb res1234",
  derive_available_code_from: {
    tel:"+1415867 0550" # E.164 format
  }
})

=> code set to 4158 as “0550” isn’t valid
```

Roadmap

Our goal is to take a user-story-driven approach across all important device categories

2022-23

Smart Locks & Access Systems

2023-24

Thermostats, Submeters, EV Chargers, Lights

2024-25

Sensors & Camera

We Are Hiring!

Looking for sharp minds with kind souls

Company Description

Seam is a Series-A company backed by Y-Combinator and other investors. Our team is based in San Francisco, and throughout the United States, Europe, and Asia. All roles receive salary compensation, equity, and education stipend.

Roles

- **Full-stack Engineers** (Typescript/Node)
- **Technical Sales**
- **Product Manager**

All jobs at seam.co/jobs