APIs and NIEM

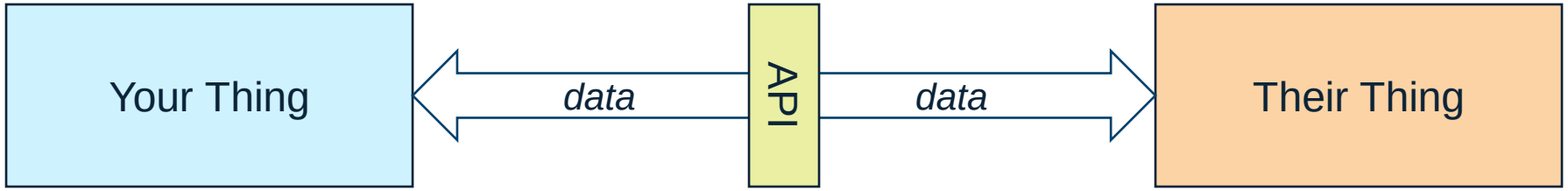
Dr. Scott Renner

1 February 2023

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

DRAFT

Application Programming Interface (API)



- **Interface**
  - Tells the people how to interact with the thing on the other side
- **Programming**
  - Those people are software developers
  - Tells developers what they must know to write code that interacts with the other thing
  - An API is a contract between developers
- **Application**
  - Once upon a time, only end-user applications had “APIs”, by definition
  - That distinction is no longer maintained

Kinds Of API

- **Library API**
  - Tells you how to use a software library (“their thing”)
  - A form of software reuse – now you don’t have to write the library functionality yourself
  - Library code becomes part of your application

Kinds Of API

- **Library API**
- **Computing Platform API**
  - Tells you how to write code interacting with the application server, operating system, etc.
  - Your program interacting with “their thing” on the same machine

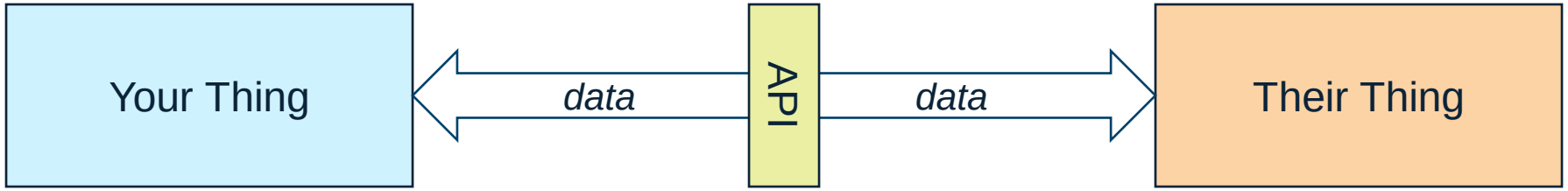
Kinds Of API

- **Library API**
- **Computing Platform API**
- **Remote API**
  - When “their thing” isn’t executing on the same machine
  - Tells you how to write code to interact with a remote resource over the network
  - Various forms of plumbing (CORBA, Java RMI, message passing, pub/sub, etc.)

Kinds Of API

- **Library API**
- **Computing Platform API**
- **Remote API**
- **Web API**
  - A kind of remote API
  - Plumbing uses HTTP / HTTPS
  - Content exchanged is usually in XML or JSON format
  - Examples: OData, OpenAPI, SOAP+WSDL

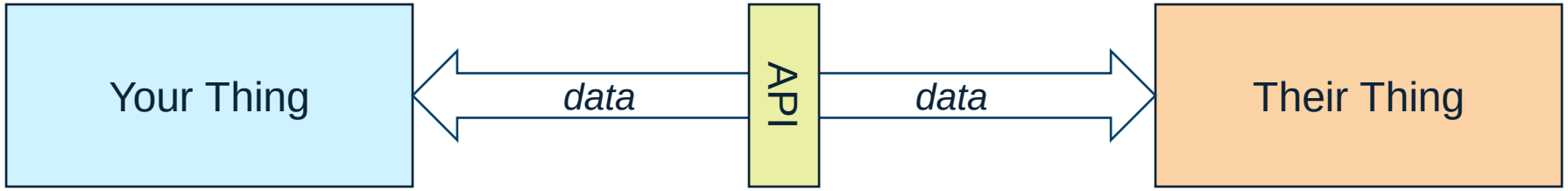
Abstraction and Information Hiding



- **The API presents a useful abstraction of "their thing" implementation**
  - For example: The OS and hardware present a "file system" abstraction
  - In reality, there are no "folders", just sectors on a spinning magnetic disk
  - But you do not have to know or care about any of that
- **In theory, you do not care about the actual implementation**
- **In practice, all abstractions leak**
  - Sometimes the implementation details do matter
  - Example: file shredding tools don't do what you want on a solid-state hard drive

"In theory, there's no difference between theory and practice. In practice, there is"

APIs, Data Specifications, and NIEM, Oh My!



- **Different developers are writing software for Your Thing and for Their Thing**
- **They must have a compatible understanding of the exchanged data**
- **Usually accomplished through an implementation-level data specification (the contract)**
  - What data must be passed
  - What data may be included
  - What that data means
- **NIEM is a framework for developing those implementation-level data specifications**

Plumbing, Content, and Process

- **Plumbing**
  - Connection between your programming language code and “their thing”
  - Parts may be fully specified, tool-generated, or written by hand
  - NIEM does not do plumbing; NIEM works with any plumbing
- **Content**
  - What data must be present, what data may be present, and what it all means
  - Documentation sufficient for the intended purpose
  - NIEM message specifications can define the content part of an API
    - Directly, for APIs that exchange XML or JSON documents
    - Indirectly (at this time), for APIs with an interface definition language (Google Protocol Buffers, etc.)
- **Process**
  - An API may specify a sequence of interactions
  - Shopping cart example: Authenticate, find item, add item to cart, checkout
  - At present, NIEM does not say how to specify an interaction sequence

NIEM and APIs

NIEM Model

reused in

Developers

understand

data specification

defines data content of

Your Thing

data

API

data

Their Thing

Users

NIEM Offers

- A standard format for data specifications, suitable for an API registry / repository
- Reuse of community-agreed data models, with local extensions
- A technology-neutral data modeling language compatible with ontology formalisms
- Knowledge graph representation of exchanged data
- Automated conversion between supported serializations (XML to JSON, etc.)
- Machine validation of message specifications for NIEM conformance
- Machine validation of exchanged data for message specification conformance

A NIEM Tradespace

- **Data exchange overlap**
  - High: Developers supporting multiple exchanges with overlapping subject-area content
  - Low: Developers supporting single exchange or exchanges with nothing in common
- **Developer cohesion**
  - High: Small team, single organization, short duration
  - Low: Many developers and organizations, long durations
- **NIEM is most valuable in an enterprise in which overlap is high and developer cohesion is low**
- **Most MITRE sponsors are firmly in the lower-right corner**