## § 2.4 Person Schema

There are many reasons to track, and disambiguate, natural persons in the world of compliance as Code. Natural persons (a human being as distinguished from a person as a corporation created by operation of law), heretofore referred to simply as *“persons”*, fall into two categories: contributors and targets of compliance.

### Contributors

Every organization that has a documented schema in the realm of **Compliance as Code** (listed [HERE](https://docs.grcschema.org/schema-standard/1.-general-introduction-to-the-schema-and-federated-mapping-standard/common-uses-federated-mapping)) identifies various forms of contributors to their content. These contributors are listed as anything from primary authors, through editors, through “commentators” of contributed content to their various repositories of information. We will this collective group **contributors**.

In addition, many frameworks, such as HITRUST, SCF, and others, *don’t* present *who* contributed to the content. As of this writing there is no possibility for adjudicating the veracity of *how* the mapping was done, the quality of the mapping, or even the research behind the mapping. This renders any mapping without the documented contributors (and the facts behind the mapping) non-satis probandi (no evidence).

Contributors to these systems must be tracked; for accountability of content, for accolades as contributors, for any number or reasons.

### Targets of Compliance

People aren’t just contributors. We are also the *target* of compliance. People who

* are assigned to follow compliance documents;
* are given permission to use systems;
* are given access to physical locations and assets;
* have taken actions that must be logged;
* etc.

Thus, “Ed” or “Eunice” in real-life must be directly linked to “Ed” or “Eunice” in a system’s permission structure, or who signed the HR policy, or who walked through the front door’s sentry system.

Before we discuss the schema, we must first discuss disambiguating a person from an agent, or another person.

### Digital Disambiguation of persons

Now that we know real people can be both contributors to the compliance effort *and* targets of the compliance effort, we need to understand a few things about *disambiguating* people through their digital identities[[1]](#endnote-1). Let’s look at a few use cases from our company, Univest.VIP located in Canoga Falls.

#### The problem of the Smiths

Within Univest.VIP, even in the small town of Canoga Falls, they have two people named Joseph Smith – cousins. One goes by “Joe” and the other is more formal, only answering to “Joseph”. When *Joe* joined the company, Lily, the HR director, assigned him the email address [jsmith@univest.vip](mailto:jsmith@univest.vip). However, that caused a digital identity problem for Lily six months later when his cousin *Joseph* joined. He couldn’t *also* be [jsmith@univest.vip](mailto:jsmith@univest.vip). So they changed their email naming policy (causing much consternation with IT) and now there is [joesmith@univest.vip](mailto:joesmith@univest.vip) and [josephsmith@univest.vip](mailto:josephsmith@univest.vip).

**Organizational email**, because each address has to be unique, is *one* way to disambiguate the digital representation of people in an organization. While simple and elegant for the time that *Joe* is **actively** using [joesmith@univest.vip](mailto:joesmith@univest.vip), there’s nothing to stop the organization from assigning that address to *some other* Joe Smith after *our* Joe Smith has left the organization. Unless, of course, the naming convention was like those of email service providers wherein an address is *never used twice*.

#### Ed’s problem

Then there’s Ed Higgins. Ed is a prolific contributor to many compliance websites. Ed is also a frequent job changer. Before his email address was [edhiggins@univest.vip](mailto:edhiggins@univest.vip), it was edward@conogabank.com, and [edh@conagalib.gov](mailto:edh@conagalib.gov) when we worked at the library. We’ll call these “Univest Ed”, “Bank Ed”, and “Librarian Ed”. *Librarian* Ed is the author of 15 research documents. *Bank* Ed is the author of 10 banking best practice guides. And *Univest* Ed is the author of almost a dozen investment best practice guides.

Without a way to *trace* Ed’s past emails, and link them to his digital persona, he wouldn’t get credit for two thirds of his published contributions.

#### Then there’s Eunice

Before Eunice met and married Ed, when she first began working at Univest, she was Eunice Harper. For the first two years after marriage, she was Eunice Harper Higgins. Then in their third year of marriage, she changed it again, this time to Eunice Higgins. So what’s the problem?

She signed the employee handbook as Eunice Harper. The investment applications that Univest released the year she got married have her digital persona as Eunice Harper Higgins, as do the investment logs it creates. Her new computer (and new email address) have her digital persona as Eunice Higgins. There are various policies and standards she’s signed off on with all three of her names.

#### Persons versus agents

Then there’s the problem of disambiguating a person from an *agent*. As Dan Brickley and Libbey Miller, co-authors of the FOAF (Friend of a Friend) Project[[2]](#endnote-2) wrote, people *as well as* bots can have e-mail addresses, chat addresses, etc. Distinguishing a person from a bot can’t be done by just checking an email address. Therefore, personal identity disambiguation must begin with disambiguating *people* from *things*. Until one passes the Turing test, a computer, which manages bots and various software agents, will remain an antonym of person[[3]](#endnote-3). Therefore, throughout the rest of this discussion, we will use the following terms and definitions (the term is a linked item to the [ComplianceDictionary.com](https://compliancedictionary.com/)’s entry for the term):

|  |  |
| --- | --- |
| **Term** | **Definition** |
| [agent](https://compliancedictionary.com/term/3457) | A program acting on behalf of a person or organization. |
| [bot](https://compliancedictionary.com/term/11315) | A robot; A piece of software designed to complete a minor but repetitive task automatically and on command. |
| [natural person](https://compliancedictionary.com/term/257044) | A living human being. Legal systems can attach rights and duties to natural persons without their express consent. |

Because a bot is a type of agent, we will refer to all computerized processes that communicate, as **agents**.

##### Disambiguating a person from an agent

The world is inundated with fake accounts on social media platforms run by bots of various types and for various reasons. Merely having an e-mail address or social media address isn’t enough to disambiguate a person from an agent. A shibboleth of sorts is necessary to distinguish the two[[4]](#endnote-4).

In 2015 DARPA initiated a Twitter Bot challenge to determine person or agent[[5]](#endnote-5). At that time, they came up with five criteria (user profile, syntax, semantics, behavior, network features) to disambiguate between person and agent. Since that time, Google’s Duplex[[6]](#endnote-6) and IBM’s Watson and Project Debator[[7]](#endnote-7) have made huge leaps in their capabilities to use syntax, semantics, and even linguistic behaviors. This leaves two characteristics still in play as a potential shibboleth:

1. **User Profile** – links to other accounts, biography, etc.

2. **Network features** – how distinct the person’s “network” is, i.e., where they work and others they connect to.

We will save the discussion of the various objects that should be tracked in a person’s profile for later, for now, we’ll just call this collection of objects a person’s **information**.

Since the DARPA challenge, a great deal of research and practical application has been put into place in order to disambiguate natural persons from agents as well as personal name disambiguation (once you know Joe Smith isn’t an agent, knowing Joe Smith is *that* Joe Smith versus another one)[[8]](#endnote-8). So much so, that there are several methodologies for extracting information about both people and organizations from given information such as URLs and email addresses[[9]](#endnote-9). We aren’t going to cover any of these for now – they will sit “on the back burner” as they say. For now, the easiest way to provide disambiguating information is to build out a personal dossier, so to speak, starting with that person’s current email address and what can be found via their public profiles.

#### The problem, therefore

**…**is how to

A. disambiguate a user so that we know Joe Smith is *that* Joe Smith and not some *other* Joe Smith; and

B. allow those disambiguating characteristics to be persisted between systems and digital identities so that as factoids change (name, address, email, etc.) the disambiguated person remains *that* person instead of some *other* person; and

C. ensure that Joe Smith isn’t an automated agent.

### Extracting personal information from disambiguating APIs

Various organizations have developed full-blown APIs for providing in-depth information about people, usually for marketing purposes, using their email addresses and domain names.

First and foremost, personal email addresses, such as those from Hotmail, Apple, Google, Yahoo, etc. cannot be used for extracting personal information. They just can’t. Any personal information extracted must be extracted from organizational email because the applications, such as ClearBit, FullContact, BigPicture, Powrbot, Crunchbase, ID.me, etc. all focus on business-to-business marketing, and as such, organizational e-mails. We will call these **disambiguating APIs**.

It is our postulation that there is enough information found within amalgamating information from a couple of the disambiguating APIs to put together a personal profile that is both disambiguating *and* persistent in nature *without* the need for a personal UUID for coherency.

A natural person can be disambiguated with persisting information when described in JSON-LD format using an amalgam of information found using disambiguating APIs.

### Aligning the structure of Person across the Compliance as Code spectrum

There are over 20 organizations that have contributed schemas to the realm of Compliance as Code[[10]](#endnote-10). Of those, four of them specifically call out natural persons as either a contributor to compliance, or as a target of compliance. Those are StratML, OSCAL, STIX, Cybox, and ISO 19770-3. Below we explain the differences and relationships between those schemas and that of GRCSchema, which we present in the next section.

#### StratML

StratML calls a natural person a **submitter**. Their data structure for submitter is relegated to a simple table for name, phone number, and email address, as shown below.

Diagram

Description automatically generated

StratML

StratML has both a first name (GivenName) and last name (Surname), so this is easy to concatenate into a full name (fullname) in GRCSchema.

#### OSCAL

OSCAL refers to a natural person as a **party**. In addition to name, phone, and email, they also include address, as shown below.

Diagram

Description automatically generated

OSCAL

The problem with OSCAL, as you’ll see with some of the others, is that they only have a fully concatenated name. There doesn’t seem to be a way to discern first name and last name, let alone a person’s prefix, middle name, and suffix.

#### STIX

STIX’s schema for a **threat actor** is very minimal but includes one major distinction – *type*. STIX’s threat actor can be a natural person, agent, group, organization, or even nature. Hence *type* is used to disambiguate between the various forms the threat actor can take.

Diagram

Description automatically generated

STIX

STIX suffers the same single-field naming problem.

#### Cybox

Cybox, incorporated into a great many of OASIS’ schemas, adds the addition of *role* (which we will cover in its own JSON Thing, later) to its **contributor**.

Diagram

Description automatically generated

Cybox

Cybox suffers the same single-field naming problem.

#### ISO 19770-3

While ISO 19770-3 doesn’t come out and have a full-fledged person identity, they do deal with **trustdefinedbyname**.

Diagram

Description automatically generated

ISO 19770-3

ISO 19770-3 suffers the same single-field naming problem.

#### Zotero, RDA, FRBR

Zotero, RDA, and FRBR, as well as any of the other bibliography and library catalog schemas, have a full-blown person schema that matches the Person schema found in GRCSchema, so we won’t show their *very* detailed schema next to the GRCSchema, as the two are 99% identical.

### Solving the disambiguation problem through “passport control”

If the goal is Person disambiguation, the best way to achieve it is mimic what every country is already doing when allowing people to enter its borders – utilize some form of passport control.

The schema we present below is the *storage* of a Person’s information in a way that records their various names, postal addresses, email addresses, and social media addresses. It tracks their assignment to teams, groups, and organizations. And presents a unique and persistent ID *within the federated mapping* structure.

The disambiguation comes via an authentication strategy in front of the storage... and only disambiguating through a trusted authority. The trusted authority chosen by GRCSchema is auth0.

The main thing that needs to happen to solve identity is every application in the federated mapping space needs to agree to anchor our boats to a common identity provider that proves who people are.  We must drink from the same pool consistently.  Even with a schema perfectly defined, if there are 15 practical instances in the wild, we end up with a potential for one Person to be represented 15 times (once in each practical instance).

### § 2.4 Person Schema

This describes a natural person. The beginning schema for describing a disambiguated person will be presented in JSON-LD format herein, and will be maintained online at <http://grcschema.org/Person>. Core personal information is broken down into several values, listed below, with the common values being covered previously, and can be found [HERE](https://short.grcschema.org/shared).

| **Property** | **Expected Type** | **Description** |
| --- | --- | --- |
| CoreMetaData | Thing | Metadata documenting the ID and core information about a JSON Thing. |
| email | Email or Null | The person’s primary electronic mail address. |
| Emails | Thing or Null | Additional email addresses. |
| fullname | String | A concatenation of prefix, first\_name, middle\_initial, last\_name and suffix. |
| id | Integer | A unique and persistent identifier for the record. |
| local\_reference\_id | String or Null | This optional property is ephemeral and is only displayed in the object return during a POST or a PATCH operation for core objects (primarily those found in an @set array). This value will be returned in the response body for the object instance allowing you to identify the record using your key. A locally-unique, 16-byte string is recommended. The data in this property should be considered transient. |
| Names | Thing or Null | A collection of Name that are used by a Person. |
| PersonMemberships | Thing | A Person's membership. |
| PersonRoles | Thing | A collection of PersonRole. |
| PhoneNumbers | Thing | A collection of PhoneNumber. |
| PostalAddresses | Thing or Null | A collection of PostalAddress. |
| SocialAddresses | Thing or Null | The various Internet locations that help disambiguate a person, such as their FaceBook, LinkedIn, YouTube and Twitter Address. |

Many of the elements within Person are *Things* unto themselves and are described below.

#### § 2.4.1 CoreMetaData

CoreMetaData is fully described in Common Schema Elements, § 2.1.1.

#### § 2.4.1 email and Emails

Email and Emails is fully described in Common Schema Elements, § 2.1.8.

#### ‌§ 2.4.2 Names

Names is an array of the Name object. It is used for names of natural persons, as opposed to a name of an agent, group, organization, etc. There are two values that are foreign keys “xxx\_fk” from other tables: prefix and suffix. The International Federation of Library Associations, as well as a few publishers’ associations, have taken the stance that both name prefixes and suffixes should be standardized. As such, we’ve included that methodology into the Person schema. These two items have their own schemas, which we’ll cover below.

|  |  |  |
| --- | --- | --- |
| **Property** | **Expected Type** | **Description** |
| first\_name | String | The person's first name. |
| id | Integer | A unique and persistent identifier for the record. |
| last\_name | String | The person's last name. |
| local\_reference\_id | String or Null | This optional property is ephemeral and is only displayed in the object return during a POST or a PATCH operation for core objects (primarily those found in an @set array). This value will be returned in the response body for the object instance allowing you to identify the record using your key. A locally-unique, 16-byte string is recommended. The data in this property should be considered transient. |
| middle\_initial | Char | A person's middle initial. |
| name\_prefix\_fk | Integer or Null | The prefix before a person's name, such as Dr., Mr., Mrs. |
| name\_suffix\_fk | Integer | The suffix after a person's name, such as Jr., III, etc. |
| person\_fk | Integer | Person foreign key. |
| primary | Boolean or Null | Denotes whether an object is the primary record or not. |

##### § 2.4.2.1 NamePrefix

Name prefix is the object that defines the abbreviated (Dr.) version of a person’s name as opposed to the long version (Doctor).

| **Property** | **Expected Type** | **Description** |
| --- | --- | --- |
| abbreviation | String or Null | Used for PersonSuffix and PersonPrefix as the abbreviation for either. |
| CoreMetaData | Person | Metadata documenting the ID and core information about a JSON Thing. |
| id | Integer | A unique and persistent identifier for the record. |
| local\_reference\_id | String or Null | This optional property is ephemeral and is only displayed in the object return during a POST or a PATCH operation for core objects (primarily those found in an @set array). This value will be returned in the response body for the object instance allowing you to identify the record using your key. A locally-unique, 16-byte string is recommended. The data in this property should be considered transient. |
| prefix | String or Null | The prefix to a Person's name. |

##### § 2.4.2.2 NameSuffix

Name prefix is the object that defines the abbreviated (Jr.) version of a person’s name as opposed to the long version (Junior).

|  |  |  |
| --- | --- | --- |
| **Property** | **Expected Type** | **Description** |
| abbreviation | String or Null | Used for PersonSuffix and PersonPrefix as the abbreviation for either. |
| CoreMetaData | Person | Metadata documenting the ID and core information about a JSON Thing. |
| id | Integer | A unique and persistent identifier for the record. |
| local\_reference\_id | String or Null | This optional property is ephemeral and is only displayed in the object return during a POST or a PATCH operation for core objects (primarily those found in an @set array). This value will be returned in the response body for the object instance allowing you to identify the record using your key. A locally-unique, 16-byte string is recommended. The data in this property should be considered transient. |
| suffix | String or Null | The suffix to a Person's name. |

#### § 2.4.3 PersonMemberships

This is the array of the things (as of now) that a person can belong to within the schema.

|  |  |  |
| --- | --- | --- |
| **Property** | **Expected Type** | **Description** |
| PersonGroups | Thing | A Person's Groups. |
| PersonInitiatives | Thing | A Person's Initiatives. |
| PersonOrganizations | Thing | A Person's Organizations. |
| PersonTeams | Thing | A Person's Teams. |

#### § 2.4.4 PersonRoles

This is an array of Roles a person can be assigned to.

#### § 2.4.5 PhoneNumbers

PhoneNumbers is fully described in Common Schema Elements, § 2.1.4.

#### § 2.4.6 PostalAddresses

PostalAddresses is fully described in Common Schema Elements, § 2.1.2.

#### § 2.4.7 SocialAddresses

SocialAddresses is fully described in Common Schema Elements, § 2.1.3.

1. You’d think there was a great deal of research, or even “google-able” information on how to do this. Nope. As of this writing, Google has 88,000+ articles on digital disambiguation, but most of them are about how Netflix and others are trying to do it through data profiling. We have a few good research documents you can find in our online research library [HERE](https://www.zotero.org/groups/2477535/complianceframeworks/collections/ZPTI5TAS). [↑](#endnote-ref-1)
2. (“FOAF Vocabulary Specification” n.d.) [↑](#endnote-ref-2)
3. https://compliancedictionary.com/term/8166 [↑](#endnote-ref-3)
4. A shibboleth is a way the old Isrealite army used to detect imposters from their own by manner of dialect. The term was featured in a great episode of the television show West Wing wherein the President had to determine if a person was trying to lie about something important. You can see that part of the episode [HERE](https://www.youtube.com/watch?v=fqkaBEWPH18). [↑](#endnote-ref-4)
5. (Subrahmanian et al. 2016) [↑](#endnote-ref-5)
6. (“Google Demos Duplex, Its AI That Sounds Exactly like a Very Weird, Nice Human” n.d.) [↑](#endnote-ref-6)
7. (Fan 2019) [↑](#endnote-ref-7)
8. (Vu et al. 2007) [↑](#endnote-ref-8)
9. (Delgado et al. 2018) [↑](#endnote-ref-9)
10. <https://short.grcschema.org/CAC001> [↑](#endnote-ref-10)