HUMAN COLOSSUS
FOUNDATION

Overlays Capture Architecture (OCA)

Version 1.0
October 18th, 2022

Abstract

Overlays Capture Architecture (OCA) offers a data harmonisation solution between data models and data representation formats. Primarily devised for semantic object interoperability and privacy-compliant data sharing, OCA is a proposed global standard for data capture that promises to significantly enhance the ability to define, manage, and use data in terms of simplicity, accuracy, and allocation of resources.

As defined in its Articles of Association, the mission of the Human Colossus Foundation (HCF) is to develop open data management standards for societal benefit. This document represents the first version of this effort by combining a definition of the semantic domain and its first applications (Part A) with the OCA specification (Part B). HCF endorses this document as part of the overall Dynamic Data Economy (DDE) concept.

Document Management

Author:	Paul Knowles (Human Colossus Foundation)
Editors:	Ryan Barrett (Stratosoft LLC) Robert Mitwicki (Human Colossus Foundation) Michal Pietrus (ArgonAUTHs) Marcin Olichwiruk (ArgonAUTHs) Philippe Page (Human Colossus Foundation)
Copyrights:	Human Colossus Foundation, EUPL v1.2 licence.
Governance:	The Decentralised Semantics Working Group maintains the OCA specification at Human Colossus Foundation. Comments regarding this specification can be provided as issues on the official OCA Github. The definitive reference for OCA is the official OCA website.
Contact:	contact@humancolossus.org
Version mgt:	October 18th 2022, The Backgrounder + OCA Specification v1.0

Acknowledgements

OCA development follows a synergetic process that goes beyond technology and interoperability. The Decentralised Semantics Working Group, the official forum for OCA development at the Human Colossus Foundation, continues to reach out to other data management initiatives in academia, industry and legal affairs to develop the architecture further.

The evolution of OCA has been made possible through contributions from many participants in several open community forums, including Human Colossus Foundation (HCF), MyData, the European Union's New Generation Internet (NGI) DAPSI and eSSIF-Lab initiatives, and Linux Foundation's Trust over IP (ToIP) and Linux Foundation Public Health (LFPH) projects.

The document's author and editors wish to thank those contributors, with special mention to a few individuals who have contributed to reviewing, commenting and refining this document: Stephen Curran, Kevin Dean, Carly Huitema, Steven Milstein, Sankarshan Mukhopadhyay, Thomas Pichler, Petri Seljavaara, Neil Thomson, Meri Valtiala, John Walker, and Lucy Yang.

Table of Contents

PART A: The Backgrounder

1. Introduction

[Link to the 'Guide/Introduction' page on the official OCA website]

This section is non-normative.

1.1. Semantic domain

Semantic domain *[passive] / the meaning and use of what is put in, taken in, or operated on by any process or system.*

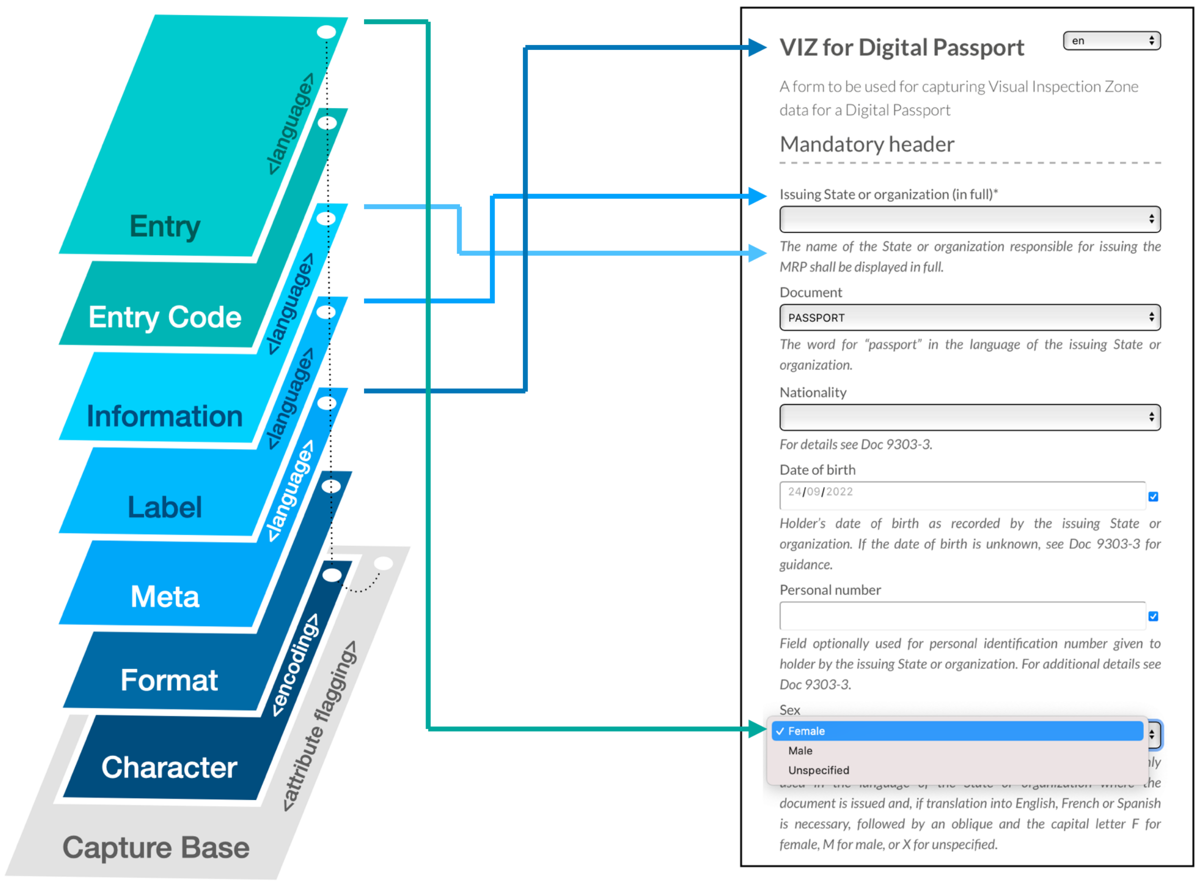
1.1.1. What is *Data semantics*?

Data semantics is the study of the meaning and use of specific pieces of data in computer programming and other areas that employ data. When studying a language, semantics refers to what individual words mean and what they mean when put together to form phrases or sentences. Data semantics focuses on how a data object represents a concept or object in the real world.

1.1.2. What is *Decentralised semantics*?

Decentralised semantics [KNO2022] is a term that describes the separation of semantic (*"definitional"*) and pragmatic (*"contextual"*) tasks into task-specific objects that, when combined, provide a digital representation of a complex object.

In data processing, metadata is *definitional data* that provides information about or documentation of other data managed within an application or environment. *Contextual data* is the background information that provides a broader understanding of an event, person, or item.



VIZ for Digital Passport

A form to be used for capturing Visual Inspection Zone data for a Digital Passport

Mandatory header

Issuing State or organization (in full)*

The name of the State or organization responsible for issuing the MRP shall be displayed in full.

Document

PASSPORT

The word for "passport" in the language of the issuing State or organization.

Nationality

For details see Doc 9303-3.

Date of birth

24/09/2022

Holder's date of birth as recorded by the issuing State or organization. If the date of birth is unknown, see Doc 9303-3 for guidance.

Personal number

Field optionally used for personal identification number given to holder by the issuing State or organization. For additional details see Doc 9303-3.

Sex

✓ Female

Male

Unspecified

...ly used in the language of the State or organization where the document is issued and, if translation into English, French or Spanish is necessary, followed by an oblique and the capital letter F for female, M for male, or X for unspecified.

Figure 1. Decentralised semantics. A complex digital object shown as an amalgamation of task-specific objects.

In the domain of decentralised semantics, task-specific objects are called *"Overlays"*. They provide layers of definitional or contextual information to a stable base object called a *"Capture Base"*.

Decentralised semantics requires the provision of deterministic object identifiers. An object is deemed deterministic if any operation's result and final state depend solely on the initial state and the operation's arguments. Object identifiers must be resolvable via the object's digest to be deemed deterministic.

1.2. Why decentralise semantics?

Working across governance boundaries is tricky because there is no common language of communication between digital ecosystems [BRU2019]. How can we hope to understand one another when we speak different languages? Add to that the complexities of language evolution and diverse governance frameworks; maintaining proper context is challenging in the current digital landscape.

The primary objective of decentralised semantics is *"data harmonisation"*, which refers to all efforts to combine data from different sources and provide users with an equivalent view of data from various studies. This objective involves the morphologic (*"structural"*), semantic (*"definitional"*), and pragmatic (*"contextual"*) harmonisation of digital objects for a common purpose. It also provides an opportunity to structure unstructured data while offering a long-term solution for data language unification within and across distributed data ecosystems.

Pending successful data harmonisation within a distributed data ecosystem, content provided by *"overlays"* underpins structured search queries for improved insights and analytics.

The key benefit of decentralised semantics is the *"distributed custodianship"* of task-specific objects (see section 2.4 for more information on distributed custodianship) without compromising the objectual integrity of the overall semantic structure. Furthermore, object interoperability is essential in an agile data economy where multiple actors from various institutions participate in complex use cases, supply chains, and data flows supported by multi-stakeholder data governance administrations and frameworks.

Decentralised semantics offers an evolutionary implementation for domain-driven design, an approach to software development that centres the development on programming a domain model with a rich understanding of the processes and rules of a domain.

1.3. What is Overlays Capture Architecture (OCA)?

Overlays Capture Architecture (OCA) is an explicit representation of task-specific objects that have deterministic relationships with other objects. These *"Overlays"* define individual semantic tasks, which, when combined, provide additional context to the object. An OCA bundle consists of a *"Capture Base"* and *"Overlays"*. The sum of its parts represents a contextually-rich schema.

The segregation of overlays by task enables interoperability in the construction process of any digital object without compromising the integrity of the semantic structure, modular components, or the relationship between those objects.

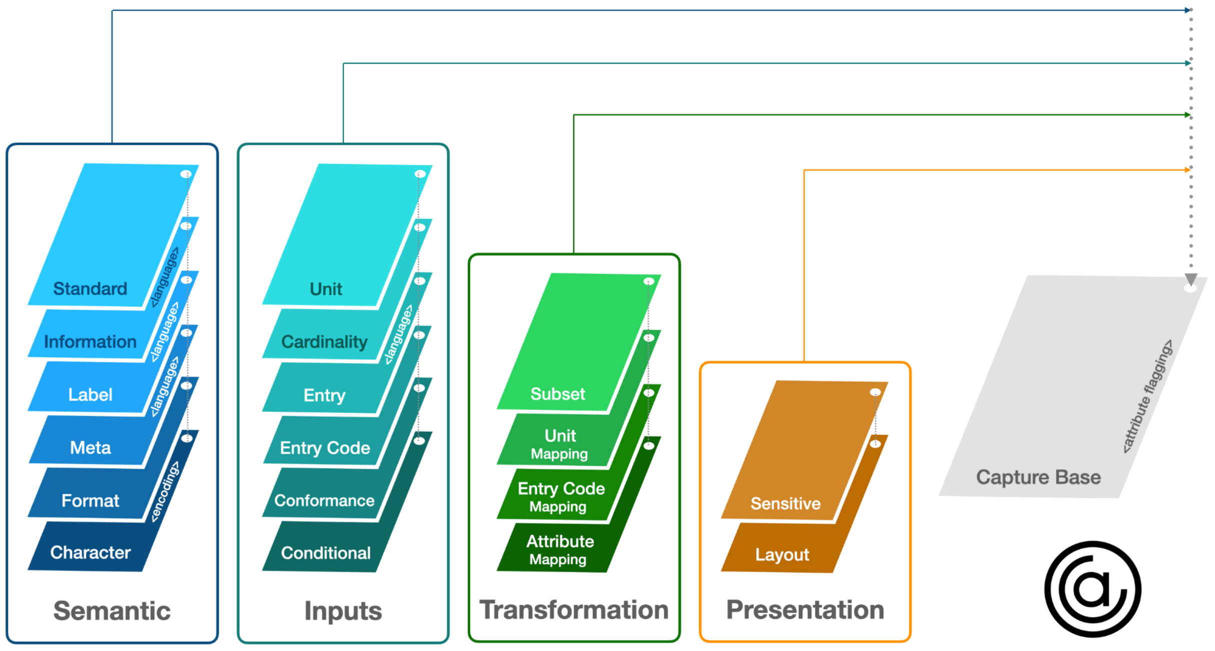


Figure 2. Semantic interoperability. Segregating task-specific objects (overlays) within a standard architecture enables different authorised controllers to update specific structural, definitional, or contextual components of the same semantic structure.

OCA is ontology-agnostic, offering a harmonisation solution between data models and data representation formats while providing a roadmap to resolve privacy-compliant data sharing between servers, networks, and across sectoral or jurisdictional boundaries.

The deterministic interplay between overlays combined with the unicity of the composite bundle is proving to be an exciting field of research.

2. Common applications

[Link to the 'Guide/Applications' page on the official OCA website]

Although *"data harmonisation"* is the core characteristic of the Semantic domain, OCA must also support the *"objectual integrity"* of any digital object and its relationships with other objects.

2.1. Application #1: Data transformation using overlays

Data transformation is a crucial data management requirement for integration, migration, data warehousing, and data preparation, involving converting data from one format to another (e.g., a database file, XML document or Excel spreadsheet). These modifications typically involve converting a raw data source into a cleansed, validated, and ready-to-use format.

OCA allows an issuer to transform data morphologically, operating on subsets of data while maintaining information over the data supply chain. Applied to machine learning (ML), OCA thus enables direct linkage of an ML-generated image with the training data used to produce the final result.

Separating overlays from the defined capture base offers a harmonisation solution between data models and data representation formats and from unstructured to structured data. Data harmonisation involves transforming datasets to fit together structurally while ensuring the definitional and contextual meaning of the source data is uniformly understood by all interacting actors, regardless of how it was collected initially.

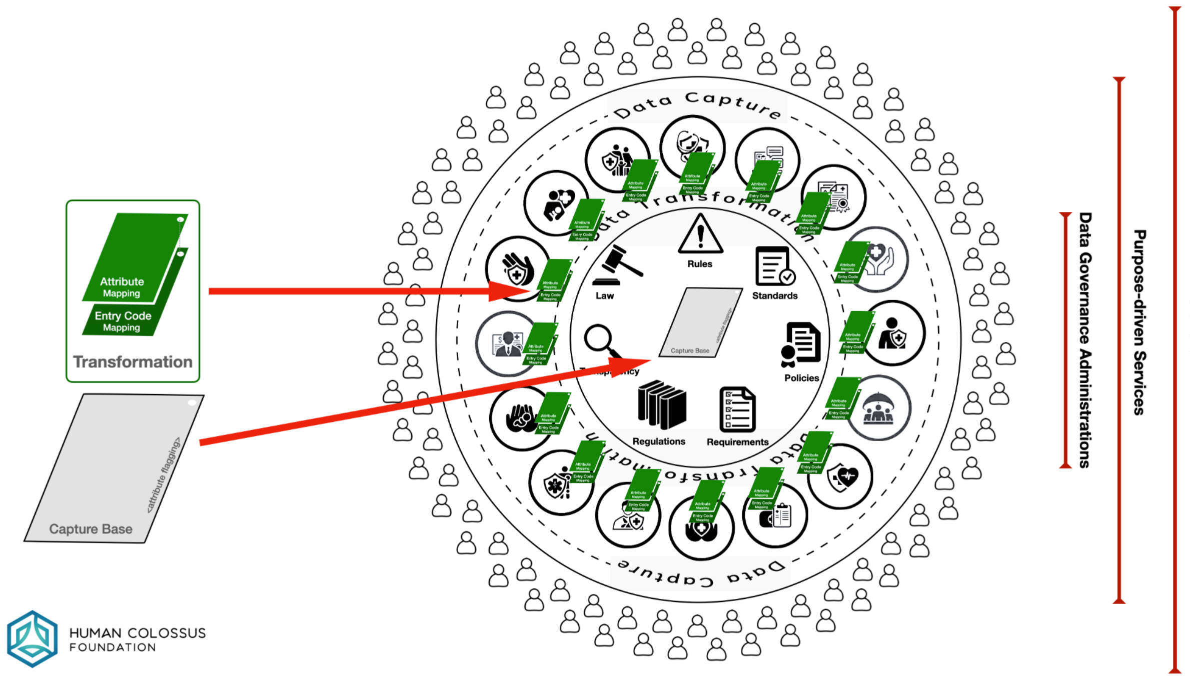


Figure 3. Data transformation. Decentralised semantics offer a harmonisation solution between data models and data representation formats, or from unstructured to structured data.

By issuing and controlling a set of proprietary transformation overlays, purpose-driven service providers can securely map source attribute names, entry codes, or unit conversions to a standard capture base defined by either a centralised organisation or a multistakeholder data governance administration. Capture bases provide a substrate for data harmonisation. Specifically, a cryptographic link is established from the transformation overlays to a consensually-defined capture base, ensuring the integrity of those objectual relationships and facilitating a secure means for data harmonisation.

Transformation overlays include:

- Attribute mapping
- Entry code mapping
- Subset
- Unit mapping

2.2. Application #2: Object presentation using overlays

In many instances of *object presentation*, the legal entity that issues the original data capture form may differ from the entity that issues the presentation objects required to produce an associated credential. For example, national passport issuance provides an opportunistic use case to demonstrate the advantages of this particular characteristic.

The International Civil Aviation Organization (ICAO) [ICAO], a specialised agency of the United Nations [UN], is tasked with planning and developing standards for safe international air transport. ICAO's primary role is to provide standards that will help regulate aviation worldwide. One of those standards is ICAO Document 9303 [ICAO9303] (endorsed by the International Organization for Standardization (ISO) [ISO] and the International Electrotechnical Commission (IEC) [IEC] as ISO/IEC 7501-1 [ISO7501]), a global standard for machine-readable travel documents (MRTD), including the data capture requirements of a machine-readable passport (MRP). As a result, ICAO is well-positioned to be the primary issuer of a standard capture base and the core overlays required for MRP form inputs, semantics, and presentation.

However, the issuance of any presentation objects needed to produce a national passport with branded design requirements would be under the remit of issuing governmental agencies, including cantonal passport offices in the country and at its embassies or consulates overseas. As an example for Switzerland, the Swiss Government is the authority to act as the primary issuer of presentation overlays to produce a branded Swiss passport, a credential identifying a traveller as a Swiss citizen or national with a right to protection while abroad, and a right to return to Switzerland.

The capture base and overlays are identifiable by *Self-Addressing IDentifiers* (SAID) [SAID], a particular type of content-addressable identifier based on a self-referential cryptographic digest. These identifiers are deterministic. In other words, there is no randomness in the identifier generation process, ensuring the objectual integrity of the digital objects and their relationships.

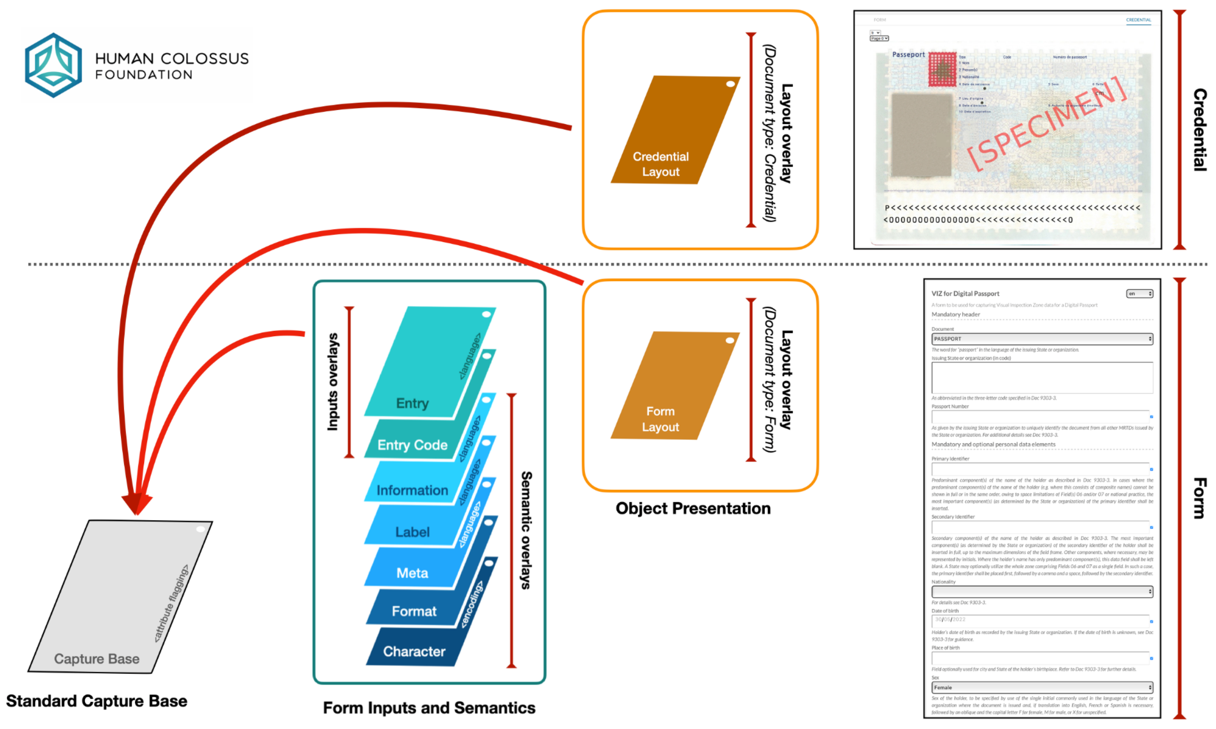


Figure 4. Dynamic presentation. The decentralised control of presentation overlays within a governed ecosystem enables the autonomous rendering of different transient objects cryptographically bound to the same capture base.

In this particular use case, authorised Swiss governmental agencies would inevitably store an instance of the ICAO-issued MRP objects in local repositories. However, the SAIDs of those digital objects would remain unchanged from the original identifiers held in an ICAO repository. As the object identifiers are deterministic, the dynamic presentation of national passports, in this case, can be established securely by maintaining a cryptographic thread from the presentation overlays to a standard capture base for global standardisation. Note that a national passport is an example of a credential presentation. However, for different use cases, the presentation of other transient object types, such as digital forms, contracts, and receipts, would also benefit from the dynamic issuance of presentation overlays.

Presentation overlays include:

- Layout
- Sensitive

2.3. Application #3: Internationalisation using language-specific overlays

Internationalisation involves designing and developing a product for target audiences that vary in culture, region, or language. The internationalisation of transient digital objects across ecosystems is essential for service providers to participate in a global market.

Let us take Switzerland as an example of a multilingual country. It is officially quadrilingual, with German, French, Italian, and Romansh as its national languages. However, many other minority languages, such as English, are becoming increasingly important. Since Switzerland is a federation, the sovereign cantons define their official language according to the primary language spoken by their inhabitants.

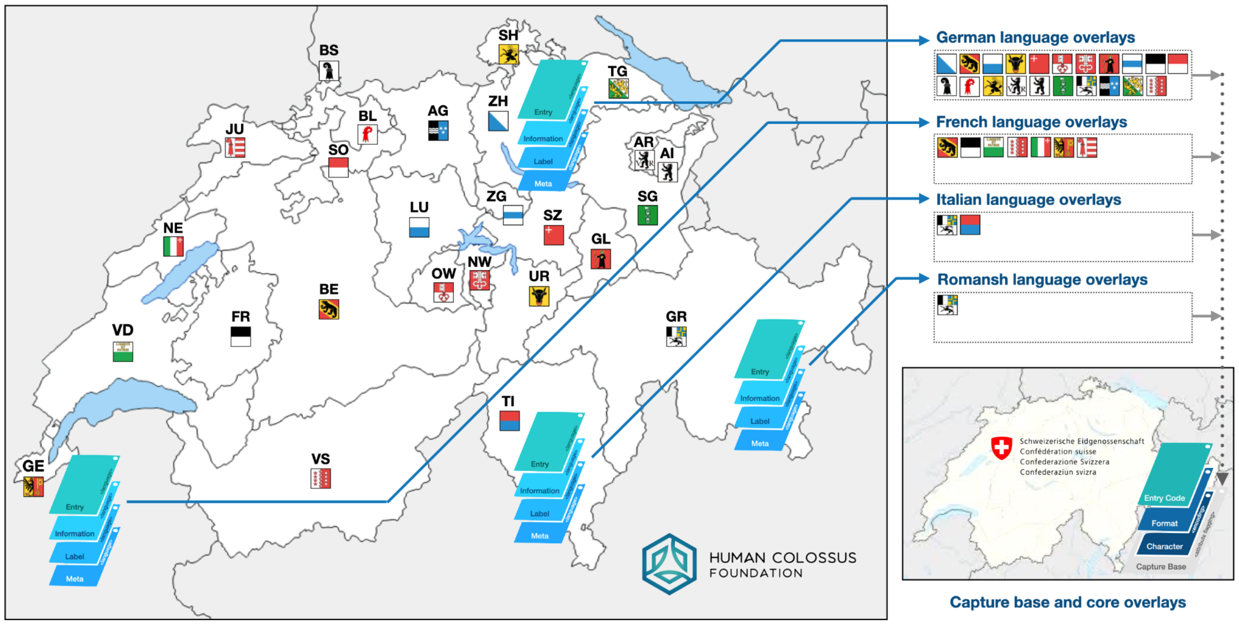


Figure 5. Internationalisation. Switzerland is a quadrilingual country. The decentralised control of language-specific overlays would enable cantonal authorities to translate official documents issued by the Swiss national federal government into their region's official language(s).

Presenting information for a purpose-driven activity in a language understandable to all recipients has commonly involved replicating digital forms, credentials, notices, and contracts into various languages based on user preferences. With federated or centralised governance authorities maintaining digital objects in multiple languages, internal data management inefficiencies are common to many organisations, institutions, and governments.

The FAIR (Findable, Accessible, Interoperable, and Reusable) data principles [FAIR2016] support the reusability of digital assets. Still, many legal entities have difficulty streamlining data management practices and processes to comply with these guiding principles.

OCA offers a solution for the internationalisation of digital objects within data ecosystems by enabling various authorised entities to control a different set of language-specific overlays for a particular transient object, such as a digital form, with a data governance administration defining and issuing a standard capture base and core language-agnostic or default language overlays.

With cantonal participation being an essential ingredient of Swiss-style federalism, separating language-specific overlays from any capture bases and core language-agnostic overlays issued by the federal government would enable a collaborative solution to internationalisation. In this scenario, decentralised semantics allow sets of language-specific overlays to be controlled and maintained by different cantons depending on their primary spoken language. In other words, distributed control of language-specific overlays would enable regional authorities to manage the official translation of any document issued by a national federal government into their region's official language(s).

The above example is globally scalable, with OCA enabling the translation of any digital object under established governance while preserving its objectual integrity. More importantly, it significantly impacts objectual inclusiveness within digital systems. Within an ecosystem, OCA allows for transient object design in a particular language, where additional interoperable language-specific overlays, including those for minority or indigenous languages, can be added dynamically.

Whether defining schemas within a centralised organisation or a multistakeholder data governance administration, OCA offers a network-agnostic solution for data harmonisation within any governance framework.

Language-specific overlays include:
- Entry
- Information
- Label
- Meta

2.4. Application #4: Distributed custodianship of task-specific objects

Distributed custodianship of capture bases, overlays, code tables, and other assets enables the responsibility of separate tasks to reside with different actors without compromising the objectual integrity of the overall semantic structure. As a result, multiple actors from various institutions can contribute to developing schemas for complex use cases, supply chains, and data flows supported by multistakeholder data governance administrations and frameworks.

2.5. Application #5: Internet of Things (IoT) applications using overlays

The Internet of Things (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. IoT data collection involves using sensors to track the performance of devices connected to the Internet of Things. In addition, the sensors track the status of the IoT network by collecting and transmitting real-time data that is stored and retrieved at any moment.

IoT applications make continuous, thorough measurements possible through low-power and wireless sensor nodes. Many existing IoT measurement mechanisms focus on obtaining real-time measurements, enabling improved insights on changes in the measurand. The closeness between the measurement's result and the measurand's true value indicates the measurement's accuracy. As such, a unit, a factor used to express the quantity of the measurand, often accompanies the measurement as a standardised quantity defined and adopted by convention or law.

Aligning measurement units is particularly important in areas requiring data sharing or conversion between units. For example, independent IoT sensors may use different units to represent their measurements, and a mapping is needed when consolidating their data.

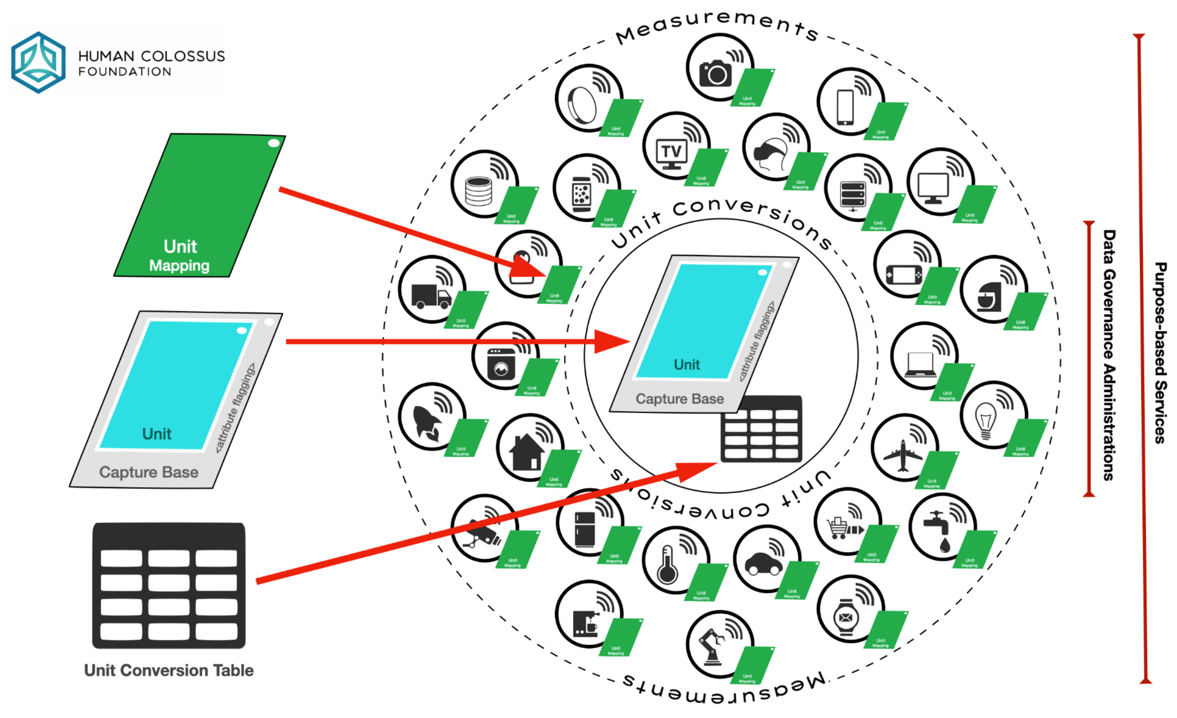


Figure 6. Unit conversions for measurements captured by IoT devices. Unit mapping overlays can be strongly bound to capture bases containing unit overlays with unit conversion tables providing recipes for seamless measurement conversions.

By issuing and controlling unit mapping overlays, purpose-driven service providers can provide source units for continuous measurements captured and transmitted by IoT devices. Capture bases and associated unit overlays issued and controlled by data governance administrations offer a target for unit harmonisation with unit conversion tables providing the recipe for seamless measurement conversions. Specifically, a cryptographic link is established from the unit mapping overlays to consensually-defined capture bases and unit overlays, ensuring the integrity of those objectual relationships and facilitating a secure means for unit harmonisation.

Overlays required for unit conversions include:

- Unit mapping *(source)*
- Unit *(target)*

PART B: Technical Specification

[Link to the 'Specification/v1.0.0' page on the official OCA website]

Disclaimer: *Strictly following DDE Principle 1.4 [HCF2022], OCA schema objects MUST be resolvable solely upon the encoded cryptographic digest of their content to ensure objectual integrity.*

All code snippets in this version of the document are in JavaScript Object Notation (JSON) [ISO21778] format. However, any serialisation format applies as long as all OCA objects have proper semantics.

1. Introduction

This section is non-normative.

1.1. How does OCA work?

OCA is based on the FAIR data principles [FAIR2016], a set of guiding principles to make data findable, accessible, interoperable, and reusable, to enable scientific data management and stewardship, and to be relevant to all digital economy stakeholders.

OCA represents transient objects (domain-agnostic) and persistent schemas (domain-specific) as multi-dimensional objects consisting of a stable *capture base* and interoperable *overlays*. By introducing *overlays* as linked task-specific objects within the schema stack, the architecture offers an optimal level of efficiency and interoperability in alignment with FAIR principles.

1.1.1. What is a *Capture Base*?

A *Capture Base* is the purest and simplest form of a schema, defining the structural characteristics of a dataset. It is the foundational layer upon which to bind task-specific objects to enhance the meaning of inputted data.

1.1.2. What are *Overlays*?

Overlays are task-specific objects that provide cryptographically-bound layers of definitional or contextual metadata to a *Capture Base*. Any actor interacting with a published *Capture Base* can use *Overlays* to transform how inputted data and metadata are displayed to a viewer or guide an agent in applying a custom process to captured data.

2. OCA object types

An *OCA object* is either a *Capture Base* or a task-specific *Overlay* with a deterministic relationship to a *Capture Base*. When amalgamated as a *Bundle*, OCA objects provide the necessary structural, definitional, and contextual information to determine the meaning of inputted data at the time of data capture.

2.1. Capture Base

A *Capture Base* is a stable base object that defines a single dataset in its purest form, providing a structural base to harmonise data. The object defines *attribute names* and *types*. It also contains a flagging block that allows schema issuers to mark potentially dangerous attributes that may capture identifying information about entities (i.e., *personally identifiable information* (PII) or *quasi-identifiable information* (QII)). Once flagged, all corresponding data can be treated as high-risk throughout the data lifecycle and encrypted or removed at any stage, reducing the risk of re-identification attacks against blinded datasets.

The *Capture Base* consists of the following attributes:

- Type
- Classification
- Attributes
- Flagged attributes

```
{
   "type":"spec/capture_base/1.0",
   "classification":"GICS:45102010",
   "attributes":{
      "dateOfBirth":"DateTime",
      "documentNumber":"Text",
      "documentType":"Array[Text]",
      "fullName":"Text",
      "height":"Numeric",
      "issuingState":"Text",
      "photoImage":"Binary",
      "sex":"Text"
   },
   "flagged_attributes":[
      "documentNumber",
      "fullName",
      "dateOfBirth",
      "photoImage"
   ]
}
```

Example 1. Code snippet for a Capture Base

2.1.1. Type

The *"type"* attribute identifies the schema object type.

Syntax: *type = "spec/capture_base/" sem_ver* *sem_ver = DIGIT "." DIGIT* Listing: ABNF core rules [RFC5234]

2.1.2. Classification

The *"classification"* attribute is for capturing a standardised classification scheme and taxonomy code to identify the primary sector, area, or topic of a published schema's intended use. Taxonomy codes provide a means for classifying schemas into groupings according to similar functions, markets, products, or services, ultimately leading to better search results for users interested in different categories.

Note: All classification schemes are supported. RECOMMENDED schemes include the *Global Industry Classification Standard* (GICS) [GICS] for industry sector classification and the

Sustainable Development Goals (SDGs) [UNSDG] for the categorisation of humanitarian advocacy and outreach activities.

```
Syntax:
classification = taxonomy ":" identifier
taxonomy = 1*( ALPHA / DIGIT / "-" )
identifier = 1*( ALPHA / DIGIT / "-" / "." )

Listing:
ABNF core rules
```

```
For the GICS classification example:
"classification": "GICS:10101010"

For the SDG classification example:
"classification": "SDG:1.1"
```

2.1.3. Attributes

The *"attributes"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the attribute type.

2.1.3.1. Attribute name

An *attribute name* is a string that uniquely identifies an attribute within an OCA layer and is used to reference that attribute by other layers throughout the OCA bundle.

2.1.3.2. Attribute type

An *attribute type* determines the attribute's syntax and how attributes of that type are compared and sorted. A *Capture Base* recognises seven core data types:

- **Text***:* a data type that defines a human-readable sequence of characters and the words they form, subsequently encoded into computer-readable formats such as ASCII [RFC0020].

- **Numeric***:* a data type that defines anything relating to or containing numbers. Examples of numeric data types include 8-byte integers, fixed precision and scale numeric data, floating precision number data, integer (whole number) data, and monetary data values.

- **Reference**: a data type that defines a *Self-Addressing IDentifier* (SAID) [SAID] that references a set of attributes through its associated parent. For example, the reference data type enables the development of nested data objects, where the organisation of information is in layers or where objects contain other similar objects. SAID is an identifier that is deterministically generated from and embedded in the content it identifies, making it and its data mutually tamper-evident.

- **Boolean**: a data type where the data only has two possible variables: *true* or *false*. In computer science, *Boolean* is an identification classifier for calculating logical truth values and algebraic variables.

- **Binary**: a data type that defines a binary code signal, a series of electrical pulses representing numbers, characters, and performed operations. Based on a binary number system, each digit position represents a power of two (e.g., *4*, *8*, *16*, etc.). A set of four binary digits or bits in binary code represents each decimal number (*0* to *9*). Each digit has two possible states: *off* and *on* (usually symbolised by *0* and *1*). Combining basic *Boolean* algebraic operations on binary numbers makes it possible to represent each of the four fundamental arithmetic operations of addition, subtraction, multiplication, and division.

- **DateTime***:* a data type that defines the number of seconds or clock ticks that have elapsed since the defined epoch for that computer or platform. Common formats include dates (e.g., *YYYY-MM-DD*), times (e.g., *hh:mm:ss*), dates and times concatenated (e.g., *YYYY-MM-DDThh:mm:ss.sss+zz:zz*), and durations (e.g., *PnYnMnD*).

  Note: The ISO 8601 [ISO8601] date and time format is the RECOMMENDED representation format for the dateTime data type, in which the Unix epoch is 1970-01-01T00:00:00Z**.**

For the Unix epoch example:
Data type: DateTime
Character encoding: UTF-8 *(default)*
Standard: ISO 8601
Format: YYYY-MM-DDThh:mm:ssZ

- **Array[***data type***]**: a data type that defines a structure that holds several data items or elements of the same data type. When you want to store many pieces of data that are related and have the same data type, it is often better to use an array instead of many separate variables (e.g., *Array[Text]*, *Array[Numeric]*, etc.).

2.1.4. Flagged attributes

Any *attributes* defined in a *Capture Base* that may contain identifying information about entities (i.e., *personally identifiable information* (PII) or *quasi-identifiable information* (QII)) can be flagged.

The *Blinding Identity Taxonomy* (BIT) [KAN2020] is a practical tool for any practitioner whose organisation has custody or control of a dataset containing identifiable information about entities, including a natural person, organisation, or device with signing capabilities that make that entity uniquely identifiable. For example, data protection officers and schema issuers can use the BIT to flag a list of elements which require cryptographic encoding to reduce the risk of identifying a data principal.

2.2. Overlays

Overlays are cryptographically-linked objects that provide layers of task-oriented definitional or contextual information to a *Capture Base*. Any actor interacting with a published *Capture Base* can use *Overlays* to add metadata to the underlying object, transform how information is displayed to a viewer, or guide an agent in applying a custom process to captured data.

Overlays consist of the following attributes:

- Capture base
- Type
- Overlay-specific attributes
    - See specific overlay types for more information.

2.2.1. Common attributes

2.2.1.1. Capture base

The *"capture_base"* attribute contains the SAID of the *Capture Base* to cryptographically anchor to that parent object.

2.2.1.2. Type

The *"type"* attribute identifies the schema object type.

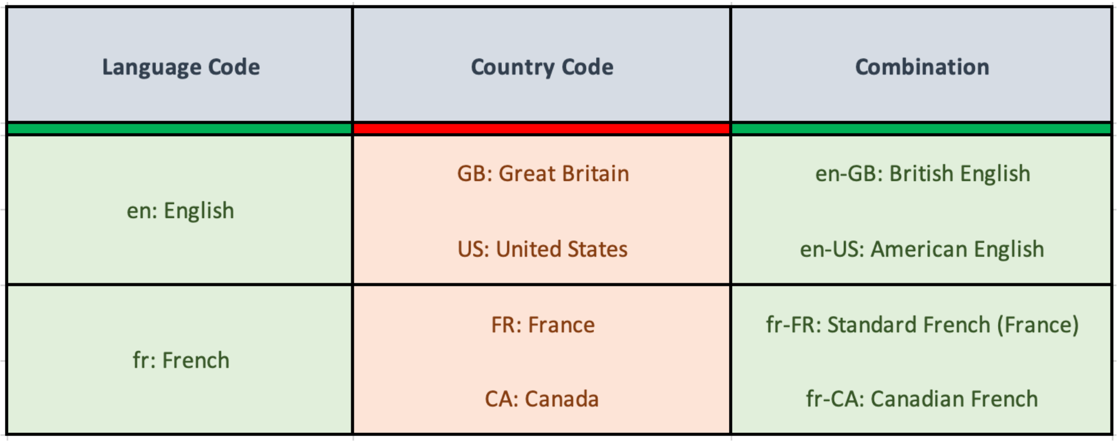
Syntax:
```
type = "spec/overlay/" overlay_name "/" sem_ver
overlay_name = ALPHA
sem_ver = DIGIT "." DIGIT
```

Listing:
ABNF core rules

2.2.1.3. Language

The International Organization for Standardization (ISO) [ISO] has standardised two lists of language-related codes: the language codes called ISO 639-1 alpha-2 [ISO639] codes (*"Codes for the representation of names of languages"*) and ISO 3166-1 alpha-2 [ISO3166] codes (*"Codes for the representation of names of countries"*). Both consist of two letters. The language code is written in lowercase while the country code is written in uppercase. However, both ISO classifications may be combined to differentiate regional languages.

The *"language"* attribute MUST contain either the two-letter language code *(lowercase)* for a national language or the combined language *(lowercase)*/country *(uppercase)* code for a regional language or locale.



Language Code	Country Code	Combination
en: English	GB: Great Britain	en-GB: British English
	US: United States	en-US: American English
fr: French	FR: France	fr-FR: Standard French (France)
	CA: Canada	fr-CA: Canadian French

Table 1. An example of ISO standard values for language and combined language/country codes.

2.2.2. Semantic Overlays

Semantic overlays provide contextual meaning to describe objects and their relationships.

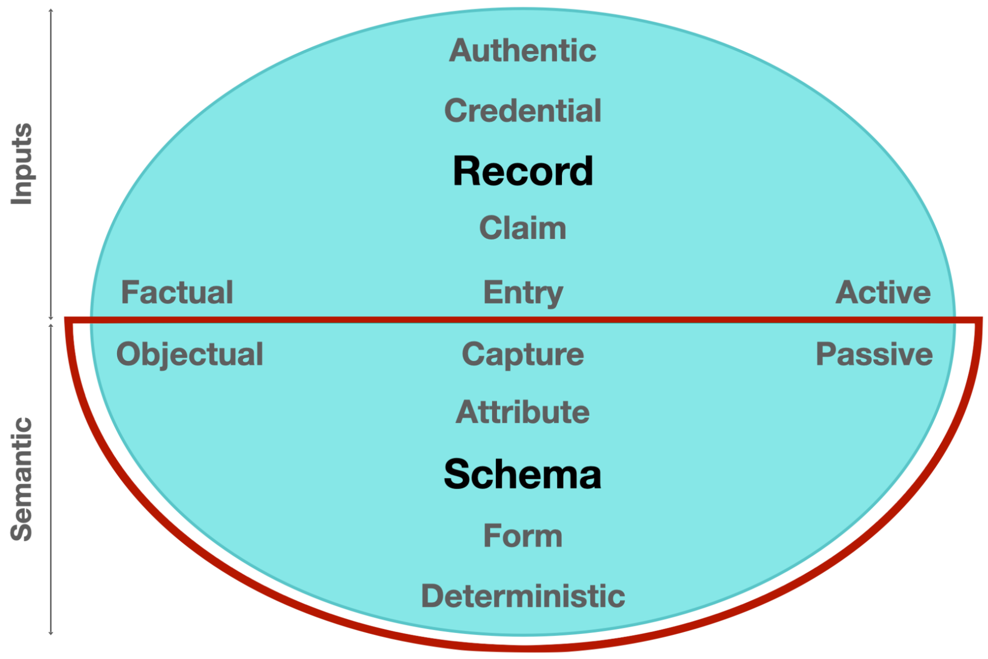


Figure 1. In a balanced network, semantic overlays determine the meaning and use of what is put in, taken in, or operated on by any process or system.

2.2.2.1. Character Encoding Overlay

A *Character Encoding Overlay* defines the process of assigning numbers to graphical characters, especially the written characters of human language, allowing them to be stored, transmitted, and transformed using digital computers. Character encoding using internationally accepted standards permits worldwide interchange of text in electronic form.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Character Encoding Overlay* MAY include the following attributes:

- *default_character_encoding*

  The *"default_character_encoding"* attribute specifies the default character encoding for the attributes contained in the parent *Capture Base*.

The most common default character set is UTF-8 [RFC3629], which accounts for 98% of all web pages in the World Wide Web and up to 100.0% for some languages, as of 2021.

- *attr_character_encoding*

  The *"attr_character_encoding"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the character encoding.

  Any attributes contained in the *"attr_character_encoding"* attribute override the behaviour of the *"default_character_encoding"* attribute.

There are many encoding standards including Base64 [RFC4648], UTF-8, and ASCII to name a few. Each standard has a purpose, and applications using those encoding standards expect to receive data compliant with that encoding standard. The most popular types of character encoding are ASCII and Unicode [UNICODE]. While ASCII is still supported by nearly all text editors, Unicode is more commonly used because it supports a larger character set. Unicode is often defined as UTF-8, UTF-16 [RFC2781], or UTF-32 [ISO10646], which refer to different Unicode standards.

An example of character encoding for a text format:
Data type: Text
Character encoding: UTF-8 *(default)*
Standard: ReGex
Format: [A-Z0-9]{9}

An example of binary-to-text encoding for an image format:
Data type: Binary
Character encoding: Base64
Standard: ISO/IEC 10918
Format: image/jpeg

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/character_encoding/1.0",
   "default_character_encoding":"utf-8",
   "attribute_character_encoding":{
      "photoImage":"base64"
   }
}
```
Example 2. Code snippet for a Character Encoding Overlay

2.2.2.2. Format Overlay

A *Format Overlay* defines an input and display format for data fields. The data format enables conversion of the input buffer to the program variable and displays program variable data to form fields.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Format Overlay* MUST include the following attribute:

- *attribute_formats*

  The *"attribute_formats"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the format.

The inputted format values are dependent on the following core data types as defined by the attribute types in the *Capture Base*:

- **Text**: The inputted format value for this core data type MAY be a *regular expression* [REGEX], a sequence of characters that specifies a search pattern in text.
- **Binary***:* The inputted format value for this core data type MAY be a MIME type registered with the Internet Assigned Numbers Authority (IANA) [IANA]
- **DateTime***:* The inputted format value for this core data type MAY be a date and time representation as defined by ISO 8601, a standard for the worldwide exchange and communication of date and time-related data.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/format/1.0",
   "attribute_formats":{
      "dateOfBirth":"YYYY-MM-DD",
      "documentNumber":"[A-Z0-9]{9}",
      "photoImage":"image/jpeg",
      "sex":"[A-Z]{1}"
   }
}
```
Example 3. Code snippet for a Format Overlay

2.2.2.3. Information Overlay

[language-specific object]

An *Information Overlay* defines attribute field descriptions and usage notes to assist the data entry process or to add context to presented data.

In addition to the *"capture_base"*, *"type"*, and *"language"* attributes (see section 2.2.1), the *Information Overlay* MUST include the following attribute:

- *attribute_information*

  The *"attribute_information"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the informational prose in a specific language.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/information/1.0",
   "language":"en",
   "attribute_information":{
      "dateOfBirth":"Holder’s date of birth as recorded by the issuing State or
organization.",
      "documentNumber":"Unique identification number of the document.",
      "documentType":"The word for “passport” in the language of the issuing State or
organization.",
      "fullName":"Full name of the passport holder.",
      "height":"Recorded height of the passport holder.",
      "issuingState":"Name of the State or organization responsible for issuing the
passport.",
      "photoImage":"Portrait image of the passport holder.",
      "sex":"Sex of the passport holder."
   }
}
```
Example 4. Code snippet for an Information Overlay (language: en)

2.2.2.4. Label Overlay

[language-specific object]

A *Label Overlay* defines attribute and category labels. For example, for an attribute named `dateOfBirth`, you may wish to display the label as `Date of birth`, which is more meaningful and user-friendly when displayed to an end user in places such as form inputs and error messages.

In addition to the *"capture_base"*, *"type"*, and *"language"* attributes (see section 2.2.1), the *Label Overlay* MUST include the following attribute:

- *attribute_labels*

  The *"attribute_labels"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is a human-meaningful attribute label in a specific language.

and MAY include the following attributes:

- *attribute_categories*

  The *"attribute_categories"* attribute contains category identifiers.

- *category_labels*

  The *"attribute_categories"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is a human-meaningful category label in a specific language.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/label/1.0",
   "language":"en",
   "attribute_labels":{
      "dateOfBirth":"Date of birth",
      "documentNumber":"Passport Number",
      "documentType":"Document",
      "fullName":"Name",
      "height":"Height",
      "issuingState":"Issuing State or organization (in full)",
      "photoImage":"Portrait image",
      "sex":"Sex"
   },
   "attribute_categories":[
      "_cat-1_",
      "_cat-2_",
      "_cat-3_",
      "_cat-4_"
   ],
   "category_labels":{
      "_cat-1_":"Mandatory header",
      "_cat-2_":"Mandatory personal data elements",
      "_cat-3_":"Mandatory identification feature",
      "_cat-4_":"Optional data elements"
   }
}
```
Example 5. Code snippet for a Label Overlay (language: en)

2.2.2.5. Meta Overlay

[language-specific object]

A *Meta Overlay* defines any language-specific information about a schema. It is used for discovery and identification and includes elements such as the schema name and description.

In addition to the *"capture_base"*, *"type"*, and *"language"* attributes (see section 2.2.1), the *Meta Overlay* SHOULD include the following attributes:

- *name*

  The *"name"* attribute contains the name of the schema in a specific language.

- *description*

  The *"description"* attribute contains a description of the schema in a specific language.

and MAY include other attributes at the discretion of the overlay producer, such as an *"affiliation"* attribute in the example below. How the overlay producer conveys the purpose of the additional attributes in the Meta Overlay is outside the scope of this specification.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/meta/1.0",
   "language":"en",
   "name":"Digital Passport",
   "description":"An example of a Digital Passport schema",
   "affiliation":"The Government of Antarctica"
}
```
Example 6. Code snippet for a Meta Overlay (language: en)

2.2.2.6. Standard Overlay

A *Standard Overlay* defines a documented agreement or technical specification published by a standards organisation used to represent, format, define, structure, tag, transmit, manipulate, use, and manage data.

In addition to the *"capture_base"* and *"type"* attributes (see [section 2.2.1](#)), the *Standard Overlay* MUST include the following attribute:

- *attr_standards*

  The *"attr_standards"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the standard.

There are many international standards organisations establishing tens of thousands of standards covering almost every conceivable topic. The three largest and most well-established standards organisations are the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC) [IEC], and the International Telecommunication Union (ITU) [ITU]. Standards tend to contain the acronym of the standards organisation followed by an internal document identifier.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/standard/1.0",
   "attr_standards":{
      "dateOfBirth":"ISO 8601"
   }
}
```
Example 7. Code snippet for a Standard Overlay

2.2.3. Inputs Overlays

Inputs overlays provide predefined inputs for data attestations.

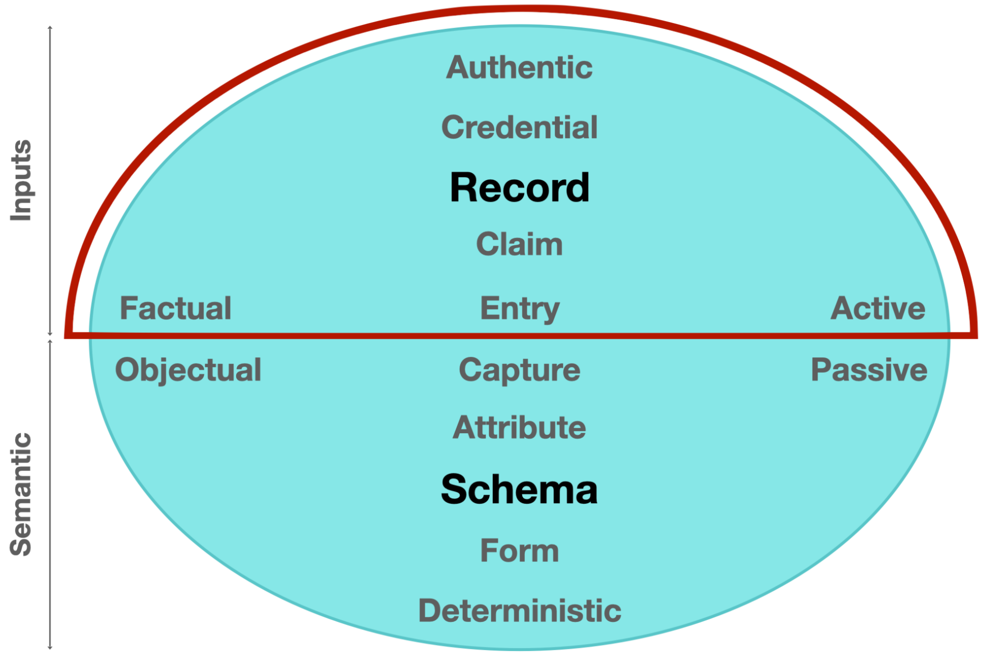


Figure 2. In a balanced network, inputs overlays determine what is put in, taken in, or operated on by any process or system.

2.2.3.1. Cardinality Overlay

A *Cardinality Overlay* defines the minimum and maximum number of values that an attribute can have. For a relationship, the cardinality interval specifies the minimum and maximum number of relationship targets.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Cardinality Overlay* MUST include the following attribute:

- *attr_cardinality*

  The *"attr_cardinality"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the cardinality interval.

The logic of cardinality intervals is as follows:

- *n* : The cardinality interval denotes exactly *'n'* entries;
- *n-* : The cardinality interval denotes a minimum of *'n'* entries;
- *n-m* : The cardinality interval denotes a minimum of *'n'* and maximum of *'m'* entries;
- *-m* : The cardinality interval denotes a maximum of *'m'* entries.

Note that *'n'* and *'m'* are positive integers.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/cardinality/1.0",
   "attr_cardinality":{
      "documentType":"1-2"
   }
}
```
Example 8. Code snippet for a Cardinality Overlay.

2.2.3.2. Conditional Overlay

A *Conditional Overlay* defines conditional expressions (or *rules*) that trigger specific computations or actions depending on whether, upon evaluation, programmer-defined Boolean expressions return *true* or *false* values. Met conditions can have a direct impact on data capture and data validation processes where, for example, expressions:

- MAY facilitate schema extensions;
- MAY enable schema abstractions;
- MAY activate validation processes.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Conditional Overlay* MUST include the following attributes:

- *attribute_conditions*

  The *"attribute_conditions"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the conditional expression.

  Expressions MAY contain placeholders to be substituted by values defined by the *"attribute_dependencies"* attribute.

Syntax:

```
conditional-statement = 1*conditional-expression

conditional-expression = equality-relational / equality-relational logical-operator

equality-relational = equality-expression / relational-expression

logical-operator = *SP ("&&" / "||") *SP

equality-expression = assignment eql-op assignment

eql-op = "=="/ "!="

relational-expression  = assignment relational-op assignment

relational-op = "<" / ">" / "<=" / ">="

assignment = *SP (ALPHA / DIGIT / "\${" DIGIT "}") *SP
```

Listing:
ABNF core rules

- *attribute_dependencies*

  The *"attribute_dependencies"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is an array value which triggers the evaluation process of the conditional expression defined by the *"attribute_conditions"* attribute.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/conditional/1.0",
   "attribute_conditions":{
      "height":"${0}=='PM'"
   },
   "attribute_dependencies":{
      "height":[
         "documentType"
      ]
   }
}
```

Example 9. Code snippet for a Conditional Overlay. "${0}" is an integer placeholder that refers to a replacement value during the substitution process. The "attribute_dependencies" attribute provides that replacement value through an array of attributes. Therefore, the placeholder's integer value refers to an array index that points to the value. In other words, "documentType" is bound by the integer placeholder, which triggers the evaluation process of the expression.

2.2.3.3. Conformance Overlay

A *Conformance Overlay* indicates whether data entry for each attribute is mandatory or optional.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Conformance Overlay* MAY include the following attributes:

- *attribute_conformance*

  The *"attribute_conformance"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the data entry conformance of the attribute, which is set to either M *(mandatory)* or O *(optional)*.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/conformance/1.0",
   "attribute_conformance":{
      "dateOfBirth":"M",
      "documentNumber":"M",
      "documentType":"M",
      "fullName":"M",
      "height":"O",
      "issuingState":"M",
      "photoImage":"M",
      "sex":"M"
   }
}
```
Example 10. Code snippet for a Conformance Overlay

2.2.3.4. Entry Code Overlay

An *Entry Code Overlay* defines the entry keys in a series of key-value pairs stored in a *code table* (also known as a “*lookup table”*) or dataset. The key is a unique identifier that points to its associated value.

Attribute Name	**Entry Code Overlay - Pre-defined Entry Keys**	**Entry Overlay - Pre-defined Entry Values**
sex	0: F	F: Female
	1: M	M: Male
	2: X	X: Unspecified

Table 2. An example of how the values in an array of key-value pairs provided by an *Entry Code Overlay* subsequently define a set of pre-defined entry keys in a nested series of key-value pairs. The specified values are often standardised categorisation codes, valuable data inputs for statistical analysis, machine learning (ML), and artificial intelligence (AI) algorithms.

In addition to the *“capture_base”* and *“type”* attributes (see section 2.2.1), the *Entry Code Overlay* MUST include the following attribute:

- *attribute_entry_codes*

  The *“attribute_entry_codes”* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is either:

    (i.) a set of pre-defined entry keys for a nested series of *key-value* pairs; or

    (ii.) a SAID that references a *code table* from an external data source to retrieve an array of pre-defined entry *keys* for a nested series of *key-value* pairs. See section 2.4 for more information on *code tables*.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/entry_code/1.0",
   "attribute_entry_codes":{
      "documentType":[
         "PE",
         "PM"
      ],
      "issuingState":"EGyWgdQR9dW_I5oH1HBMoO9AA_eMeb2p3XzcCRCBbKCM",
      "sex":[
         "F",
         "M",
         "X"
      ]
   }
}
```
Example 11. Code snippet for an Entry Code Overlay

2.2.3.5. Entry Overlay

[language-specific object]

An *Entry Overlay* defines the entry values in a series of key-value pairs stored in a *code table* (also known as a “*lookup table”*) or dataset. A value is either the identified data or a pointer to that data.

Attribute Name	Entry Code Overlay - Pre-defined Entry Keys	Entry Overlay - Pre-defined Entry Values
sex	0: F 1: M 2: X	F: Female M: Male X: Unspecified

Table 3. An example of how an *Entry Overlay* can leverage a set of pre-defined entry keys in a nested series of key-value pairs provided by an *Entry Code Overlay* to provide human-meaningful values in a specified national or regional language.

In addition to the *"capture_base"*, *"type"*, and *"language"* attributes (see section 2.2.1), the *Entry Overlay* MUST include the following attribute:

- *attribute_entries*

  The *"attribute_entries"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is either:

  (i.) a set of pre-defined values in a nested series of key-value pairs that are human-meaningful and language-dependent where the entry keys are taken from an associated *Entry Code Overlay*; or

  (ii.) a SAID that references a *code table* from an external data source to retrieve an array of pre-defined values from a nested series of key-value pairs that are human-meaningful and language-dependent where the entry keys are taken from an associated *Entry Code Overlay*. See section 2.4 for more information on *code tables*.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/entry/1.0",
   "language":"en",
   "attribute_entries":{
      "documentType":{
         "PE":"DIPLOMATIC PASSPORT",
         "PM":"PASSPORT"
      },
      "issuingState":"Els6NxGvFfyL5aiBWR3j7YiaS7F4j4O-F0EIlZu-dO0g",
      "sex":{
         "F":"Female",
         "M":"Male",
         "X":"Unspecified"
      }
   }
}
```
Example 12. Code snippet for an Entry Overlay (language: en)

2.2.3.6. Unit Overlay

A *Unit Overlay* defines the units of measurement adopted by convention or law, used as a standard for measuring the same kind of quantitative data. The RECOMMENDED system to use is the International System of Units (SI) [BIPM], French *Système International d'Unités*, an international decimal system of weights and measures derived from and extending the metric system of units.

Adopted by the 11th General Conference on Weights and Measures (CGPM) in 1960, it is abbreviated SI in all languages. To date, the SI comprises seven base units: the *meter* (m), the *kilogram* (kg), the *second* (s), the *ampere* (A), the *kelvin* (K), the *candela* (cd) and the *mole* (mol).

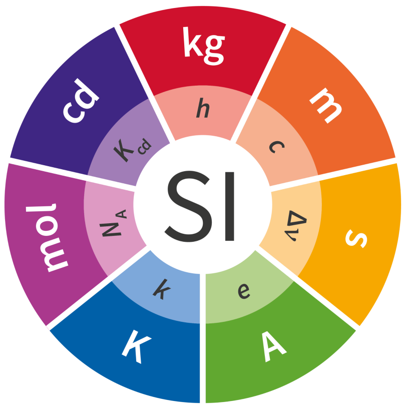


Figure 3. The seven defining constants of the SI

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Unit Overlay* SHOULD include the following attribute:

- *metric_system*

  The *"metric_system"* attribute contains the acronym of the chosen system of units (a coherent system of units of measurement) used for defining attribute units.

and MUST include the following attribute:

- *attribute_units*

  The *"attribute_units"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is a standard unit of measurement from a known metric system.

```
{
   "capture_base":"EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
   "type":"spec/overlays/unit/1.0",
   "metric_system":"SI",
   "attribute_units":{
      "height":"cm"
   }
}
```
Example 13. Code snippet for a Unit Overlay

2.2.4. Transformation Overlays

Transformation overlays provide information to convert data from one format or structure to another, such as raw data to processed, or unstructured to structured.

2.2.4.1. Attribute Mapping Overlay

An *Attribute Mapping Overlay* defines attribute mappings between two distinct data models. Data mapping provides a preliminary step for data integration tasks, including data transformation or data mediation between a data source and a destination or consolidation of multiple databases into a single database and identifying redundant columns of data for consolidation or elimination.

```
 {
    "capture_base":"Ev_RaB-gIOn8VAB3sg40mINxjiYRxdLVQrgce0aZbFcc",
    "type":"spec/overlays/mapping/1.0",
    "attribute_mapping":{
      "first_name":"firstName",
      "last_name":"surname"
    }
  }
```
Example 14. Code snippet for an Attribute Mapping Overlay

2.2.4.2. Entry Code Mapping Overlay

An *Entry Code Mapping Overlay* defines the entry key mappings between two distinct *code tables* or datasets.

```
{
   "capture_base":"Ev_RaB-gIOn8VAB3sg40mINxjiYRxdLVQrgce0aZbFcc",
   "type":"spec/overlays/entry_code_mapping/1.0",
   "attr_entry_codes_mapping":{
      "country_code":[
         "AFG:AF",
         "ALB:AL",
         "DZA:DZ",
         "ASM:AS",
         "AND:AD",
         "AGO:AO",
         "AIA:AI",
         "ATA:AQ",
         "ATG:AG",
         [
            "..."
         ]
      ]
   }
}
```

Example 15. Code snippet for an Entry Code Mapping Overlay

2.2.4.3. Subset Overlay

A *Subset Overlay* defines a customised version of a published schema containing a subset of source attributes, including their properties, types, codes, and relationship dependencies required for the information exchange.

```
{
  "capture_base": "EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
  "type": "spec/overlays/subset/1.0",
  "attributes": [
    "dateOfBirth",
    "documentNumber",
    "documentType"
  ]
}
```

Example 16. Code snippet for a Subset Overlay

2.2.4.4. Unit Mapping Overlay

A *Unit Mapping Overlay* defines target units for quantitative data when converting between different units of measurement. Conversion of units is the conversion between different units of measurement for the same quantity, typically through multiplicative *conversion factors* (see section 2.4.3 for more information on *conversion factors*) which change the measured quantity value without changing its effects. The process of conversion depends on the specific situation and the intended purpose. This may be governed by regulation, contract, technical specifications or other published standards.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Unit Mapping Overlay* MUST include the following attributes:

- *metric_system*

  The *"metric_system"* attribute contains the acronym of the chosen system of units (a coherent system of units of measurement) used for defining attribute units.

- *code_table*

  The *"code_table"* attribute contains a SAID that references an external *code table*. See section 2.4 for more information on *code tables*.

- *attr_units*

  The *"attr_units"* attribute maps *key-value* pairs where the *key* is the attribute name and the *value* is the desired unit of measurement.

```
{
   "capture_base":"Ev_RaB-gIOn8VAB3sg40mINxjiYRxdLVQrgce0aZbFcc",
   "type":"spec/overlays/unit/1.0",
   "metric_system":"SI",
   "code_table":"E3YDLacdI1GSGWhHywzrb5B0hOL/9TYWBsUkXC8fA4EY",
   "attr_units":{
     "blood_glucose":"mg/dL"
   }
 }
```
Example 17. Code snippet for a Unit Mapping Overlay

2.2.5. Presentation Overlays

Presentation overlays provide information to display digital documents at the application layer, including digital *forms* and *credentials*.

2.2.5.1. Layout Overlay

[Currently under review by Decentralised Semantics Working Group]

A *Layout Overlay* defines presentation information required by an application to display a digital document, including a digital *form* or *credential*.

Disclaimer: *Collaborators in several open communities have shown an interest in leveraging OCA's "task-specific" ethos to work on new overlay types to ensure that data presentation (see section 3.4) is both extensible and interoperable. Deprecation of the Layout Overlay in favour of a more granular approach is likely in the future.*

2.2.5.2. Sensitive Overlay

A *Sensitive Overlay* defines attributes not necessarily flagged in the *Capture Base* that need protecting against unwarranted disclosure. For example, data that requires protection for legal or ethical reasons, personal privacy, or proprietary considerations.

In addition to the *"capture_base"* and *"type"* attributes (see section 2.2.1), the *Sensitive Overlay* MUST include the following attribute:

- *attributes*

  The *"attributes"* attribute is an array of attributes considered sensitive.

```
{
  "capture_base": "EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis",
  "type": "spec/overlays/sensitive/1.0",
  "attributes": [
    "sex"
  ]
}
```
Example 19. Code snippet for a Sensitive Overlay

2.3. Bundle

An *OCA Bundle* contains a set of OCA objects consisting of a *Capture Base* and bound *Overlays.* An encoded cryptographic digest of the contained objects produces a deterministic identifier for the bundle.

The following object types are REQUIRED in any OCA bundle to preserve the minimum amount of structural, definitional, and contextual information to capture the meaning of inputted data.

- Capture base
- Character encoding overlay
- Format overlay

The cardinality of several overlay types, particularly the language-specific ones (Entry, Information, Label, and Meta), can be multiple depending on the number of defined supported languages.

```
 EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis.json
├── E3SAKe0z83pfBnhhcZl19PGGKBheb35WeCJ3V6RdqwY8.json
├── Ejx0o0yuwp99vi0V-ssP6URZIXRMGj1oNKIZ1BXi4sHU.json
├── EZv1B5nNl4Rty8CXFTALhr8T6qXeO0CcKliM03sdrkRA.json
├── Eri3NLi1fr4QrKoFfTlK31KvWpwrSgGaZ0LLuWYQaZfI.json
├── EY0UZ8aYAPusaWk_TON8c20gHth2tvZs4eWh7XAfXBcY.json
├── E1mqEb4f6eOMgu5zR857WWlMUwGYwPzZgiM6sWRZkQ0M.json
├── ESEMKWoKKIf5qvngKecV-ei8MwcQc_pPWCH1FrTWajAM.json
├── EyzKEWuMs8kspj4r70_Lc8sdppnDx-hb9QqUQywjmDRY.json
├── EIGknekgJFqjgQ8ah2NwL8zNWbFrllvXVLqezgB6U3Yg.json
├── EgBxL29VsxoZso7YFirlMP334ZuC1mkel-lO7TxPxEq8.json
├── ED9PH0ZBaOci-nbnYfPgYZWGQdkyWxA-nW3REmB3vhu0.json
├── ElJEQGfAvfJEuB7JeNIcvmAPO2DIOaKkpkZyvxO-gQoc.json
├── EpW9bQGs0Lk6k5cJikN0Ep-DN6z29fwZIsbVzMBgTlWY.json
├── EIGj0LQKT9-6gCLV2QZVgi4YQZhrUl0-GKbN7sFTCSAI.json
├── EHDwC_Ucuttrsxh2NVptgBnyG4EMbG5D8QsdbeF9G9-M.json
└── meta.json
```

Example 20. A representation of an OCA Bundle as a ZIP file containing a Capture Base (first row), multiple Overlays, and a metafile (meta.json) that provides key-value mappings between the file names and the names of the OCA object types. Apart from the metafile, each file name directly represents the encoded cryptographic digest of the file.

See Appendix A for more information on the content of a *metafile* (`meta.json` in the above example).

2.4. Code Tables

A *code table* is an external dataset structured as either:

(i.) an array of data; or
(ii.) a map of *key-value* pairs.

A code table MUST be identifiable, verifiable, and resolvable by a SAID.

2.4.1. Code Table for *Keys*

A *Code Table* for *Keys* provides an anchor to a reusable dataset for a common purpose, such as a list of country codes. Therefore, this object MAY be a reference target in an *Entry Code Overlay*. See section 2.2.3.4 for more information on the *Entry Code Overlay*.

A *Code Table* for *Keys* MUST include the following attribute:

- *keys*

  The *"keys"* attribute is an array of pre-defined entry *keys* for a nested series of *key-value* pairs. A *key* is a unique identifier that points to an associated value.

```
{
   "keys":[
      "AF",
      "AL",
      "DZ",
      "AS",
      "AD",
      "AO",
      "AI",
      "AQ",
      "AG",
      "..."
   ]
}
```

Example 21. Code snippet for a Code Table for Keys, providing an anchor for, in this case, two-character ISO country codes.

2.4.2. Code Table for *Key-Value pairs*

A *Code Table* for *Key-Value pairs* provides a mapping of input values to output values.

A *Code Table* for *Key-Value pairs* MUST include the following attribute:

- *entries*

  The *"entries"* attribute is a map of *key-value* pairs, where the *key* is the input value (the *source*) and the *value* is the output value (the *product*).

```
{
   "entries":{
      "AFG":"AF",
      "ALB":"AL",
      "DZA":"DZ",
      "ASM":"AS",
      "AND":"AD",
      "AGO":"AO",
      "AIA":"AI",
      "ATA":"AQ",
      "ATG":"AG",
      [
         "..."
      ]
   }
}
```

Example 22. Code snippet for a Code Table for Key-Value pairs, providing a mapping from, in this case, three-character to two-character ISO country codes.

2.4.3. Code Table for *Unit mappings*

A *Code Table* for *Unit mappings* provides a mapping of input units to output units for quantitative data.

The unit conversion process consists of the following steps:

1. Read *source* unit.
2. Read *target* unit.
3. Convert *source* unit to *target* unit.

Conversion between units is defined as follows:

*Target unit = source unit * conversion factor + offset*

Except when converting between temperature units, offset equals 0 in most cases.

An example of Celsius to Kelvin conversion:
1. Given 37 Celsius
2. Expect Kelvin
3. 37 * 1 + 273.15 = 310.15 K

An example of Celsius to Fahrenheit conversion:
1. Given 37 Celsius
2. Expect Fahrenheit
3. 37 * 1.8 + 32 = 98.6 F

Implementers MAY find E.J. Roschke's *"Units and Unit Conversions"* (2001) [ROS2001] a helpful resource for *conversion factors*.

A *Code Table for Unit mappings* MUST include the following attribute:

- *entries*

  The *"entries"* attribute is a map of *key-value* pairs, where the *key* denotes the conversion from source to target (e.g., *"m->mm"* or *"deg_c->deg_f"*) and the *value* contains the conversion factor and the offset.

  All units and unit prefixes follow the *"Data Protocols Lightweight Standards and Patterns for Data"* [BER2013] proposal for describing units associated with numeric quantities.

```
{
   "entries":{
      "m->mm":{
         "cf":1000
      },
      "m->yd":{
         "cf":1.0936133
      },
      "deg_c->deg_f":{
         "cf":1.8,
         "o":32
      }
   }
}
```
Example 23. Code snippet for a Code Table for Unit mappings

Code Table for Unit mappings is in denormalised form, meaning that the conversion between units and unit prefixes is pre-defined for all standard unit conversions for maximum efficiency.

3. Basic concepts

This section is non-normative.

Characters provide the essential elements required for written language in the physical world. In the digital world, stored sequences of bytes known as *"data"* represent these elements. However, without a system of interpretation, data has no inherent morphological, definitional, or contextual meaning. This interpretation is provided by *"metadata"*, sets of data that provide meaning to any stored sequence of bytes.

OCA is a core utility architecture for capturing the metadata necessary to interpret and preserve the meaning of inputted data. In addition, the architecture introduces a comprehensive solution to support data validation, transformation, and presentation requirements throughout a data lifecycle.

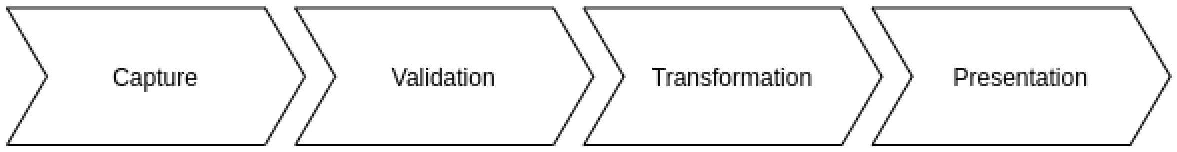


Figure 4. Universal OCA lifecycle.

If well-structured, the metadata in an OCA bundle can facilitate many ways for users to search for information, present results, and even manipulate and present information objects without compromising their integrity.

3.1. Capture

Data capture is the process of collecting structured and unstructured information electronically and converting it into data readable by a computer.

Data capture MAY involve Semantic, Inputs, and Presentation Overlays.

3.2. Validation

Data validation is the process of checking the integrity, accuracy and structure of data before it is used for a business operation.

Data validation MAY involve Semantic and Inputs Overlays.

3.3. Transformation

Data transformation is the process of converting data from one format to another, typically from the format of a source system into the required format of a destination system.

Data transformation MUST involve Transformation Overlays.

3.4. Presentation

Data presentation is the process of using various graphical formats to visually represent the relationship between two or more data sets so that, based on the results, the reader or verifier can make an informed decision.

Data presentation MAY involve Semantic, Inputs, and Presentation Overlays.

4. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted when, and only when, they appear in all capitals, as described in RFC 2119 [RFC2119].

5. References

5.1. Normative References

[ISO21778] ISO/IEC 21778:2017, *Information technology — The JSON data interchange syntax* (2017) https://www.iso.org/standard/71616.html

[SAID] Smith, S. *Self-Addressing IDentifier (SAID)* (2022) https://datatracker.ietf.org/doc/html/draft-ssmith-said

5.2. Informative References

[ISO21778] ISO/IEC 21778:2017, *Information technology — The JSON data interchange syntax* (2017) https://www.iso.org/standard/71616.html

[SAID] Smith, S. *Self-Addressing IDentifier (SAID)* (2022) https://datatracker.ietf.org/doc/html/draft-ssmith-said

[BER2013] Berkeley, A., Pollock, R., Smith, J. *Data Protocols Lightweight Standards and Patterns for Data*, Version 0.1 (2013) http://dataprotocols.org/units/

[BIPM] Bureau International des Poids et Mesures (BIPM). *The International System of Units (SI)* https://www.bipm.org/en/measurement-units

[BRU2019] Brush, K. *Digital ecosystem* (2019) https://www.techtarget.com/searchcio/definition/digital-ecosystem

[FAIR2016] Wilkinson, M. et al. *The FAIR Guiding Principles for scientific data management and stewardship* (2016) https://www.nature.com/articles/sdata201618

[GICS] MSCI. *The Global Industry Classification Standard (GICS®)* https://www.msci.com/our-solutions/indexes/gics

[HCF2022] Human Colossus Foundation. *Principles of a Dynamic Data Economy (DDE)*, Version 1.0 (2022) https://static1.squarespace.com/static/5ead4c8660689c348c80958e/t/62f288b25f9c364d7945e6eb/1660061875006/HCF+DDE+Principles+v1.0.0.pdf

[IANA] Internet Assigned Numbers Authority (IANA) https://www.iana.org/

[ICAO] International Civil Aviation Organization (ICAO) https://www.icao.int/Pages/default.aspx

[ICAO9303] Doc 9303, *Machine Readable Travel Documents*, Eighth Edition - *Part 3: Specifications Common to all MRTDs* (2021) https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf

[IEC] International Electrotechnical Commission (IEC) https://iec.ch/homepage

[ISO] International Organization for Standardization (ISO) https://www.iso.org/home.html

[ISO639] ISO 639-1:2002, *Codes for the representation of names of languages — Part 1: Alpha-2 code* (2019) https://www.iso.org/standard/22109.html

[ISO3166] ISO 3166-1:2020, *Codes for the representation of names of countries and their subdivisions — Part 1: Country code* (2020) https://www.iso.org/standard/72482.html

[ISO7501] ISO/IEC 7501-1:2008, *Identification cards — Machine readable travel documents — Part 1: Machine-readable passport* (2021) https://www.iso.org/standard/45562.html

[ISO8601] ISO 8601:2019, *Date and time format* (2019) https://www.iso.org/iso-8601-date-and-time-format.html

[ISO10646] ISO/IEC 10646:2020, *Information technology — Universal coded character set (UCS)* (2020) https://www.iso.org/standard/76835.html

[ITU] International Telecommunication Union (ITU) https://www.itu.int/en/Pages/default.aspx

[KAN2020] Knowles, P., Klingenstein, K., Wunderlich, J. *Blinding Identity Taxonomy (BIT)*, Version 1.0 (2020, Kantara Initiative) https://docs.kantarainitiative.org/Blinding-Identity-Taxonomy-Report-Version-1.0.pdf

[KNO2022] Knowles, P., Mitwicki, R., Page, P. *Decentralised semantics in distributed data ecosystems: Ensuring the structural, definitional, and contextual harmonisation and integrity of deterministic objects and objectual relationships* (2022)

[REGEX] DOCS.RS, *Crate RegEx* (Regular Expression), Version 1.6.0 https://docs.rs/regex/latest/regex/#syntax

[ROS2001] Roschke, E., *Units and Conversion Factors* (2001, Jet Propulsion Laboratory) https://www.its.caltech.edu/~culick/documents/Roschke.pdf

[RFC0020] Cerf, V. *ASCII format for network interchange*, STD 80, RFC 20, DOI 10.17487/RFC0020 (October 1969) https://www.rfc-editor.org/info/rfc2

[RFC2119] Bradner, S. *Key words for use in RFCs to Indicate Requirement Levels*, BCP 14, RFC 2119, DOI 10.17487/RFC2119 (March 1997) https://www.rfc-editor.org/rfc/rfc2119

[RFC2781] Hoffman, P., Yergeau, F. *UTF-16, an encoding of ISO 10646*, RFC 2781, DOI 10.17487/RFC2781 (February 2000) https://www.rfc-editor.org/info/rfc2781

[RFC3629] Yergeau, F. *UTF-8, a transformation format of ISO 10646*, STD 63, RFC 3629, DOI 10.17487/RFC3629 (November 2003) https://www.rfc-editor.org/rfc/rfc3629

[RFC4648] Josefsson, S. *The Base16, Base32, and Base64 Data Encodings*, RFC 4648, DOI 10.17487/RFC4648 (October 2006) https://www.rfc-editor.org/info/rfc4648

[RFC5234] Crocker, D., Ed., Overell, P. *Augmented BNF for Syntax Specifications: ABNF*, RFC 5234 (January 2008) https://datatracker.ietf.org/doc/html/rfc5234

[UN] United Nations https://www.un.org/en/

[UNICODE] Unicode https://home.unicode.org/

[UNSDG] United Nations. *Sustainable Development Goals (SDGs)* https://sdgs.un.org/goals

6. Appendices

Appendix A. An example of Metafile content

```
{
  "files": {
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] character_encoding":
"E3SAKe0z83pfBnhhcZl19PGGKBheb35WeCJ3V6RdqwY8",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] conditional":
"Ejx0o0yuwp99vi0V-ssP6URZIXRMGj1oNKIZ1BXi4sHU",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] conformance":
"EZv1B5nNl4Rty8CXFTALhr8T6qXeO0CcKliM03sdrkRA",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] entry (en)":
"Eri3NLi1fr4QrKoFfTlK31KvWpwrSgGaZ0LLuWYQaZfI",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] entry (fr)":
"EY0UZ8aYAPusaWk_TON8c20gHth2tvZs4eWh7XAfXBcY",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] entry_code":
"E1mqEb4f6eOMgu5zR857WWlMUwGYwPzZgiM6sWRZkQ0M",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] format":
"ESEMKWoKKIf5qvngKecV-ei8MwcQc_pPWCH1FrTWajAM",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] information (en)":
"EyzKEWuMs8kspj4r70_Lc8sdppnDx-hb9QqUQywjmDRY",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] information (fr)":
"EIGknekgJFqjgQ8ah2NwL8zNWbFrllvXVLqezgB6U3Yg",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] label (en)":
"EgBxL29VsxoZso7YFirlMP334ZuC1mkel-lO7TxPxEq8",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] label (fr)":
"ED9PH0ZBaOci-nbnYfPgYZWGQdkyWxA-nW3REmB3vhu0",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] layout":
"ElJEQGfAvfJEuB7JeNIcvmAPO2DIOaKkpkZyvxO-gQoc",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] meta (en)":
"EpW9bQGs0Lk6k5cJikN0Ep-DN6z29fwZIsbVzMBgTlWY",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] meta (fr)":
"EIGj0LQKT9-6gCLV2QZVgi4YQZhrUl0-GKbN7sFTCSAI",
    "[EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis] unit":
"EHDwC_Ucuttrsxh2NVptgBnyG4EMbG5D8QsdbeF9G9-M",
    "capture_base-0": "EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis"
  },
  "root": "EVyoqPYxoPiZOneM84MN-7D0oOR03vCr5gg1hf3pxnis"
}
```