Beneficial Ownership Data Standard (alpha) Documentation

Release 0.1

OpenOwnership

Mar 13, 2020

Contents

> **Warning:** This is an old version of the data standard. See latest version.

This is a stub documentation site for the alpha version of the Beneficial Ownership Data Standard.

CHAPTER 1

About

This work is taking place under the auspices of the Open Ownership project. More details on the project are avaiable at http://www.openownership.org

The work is guided by the Data Standard Working Group, and the initial phase of development is taking place between December 2016 and March 2017.

Attention: This documentation site is a work in progress.

A draft schema is ready for review on the schema page.

1.1 Contents

1.1.1 Credits (stub)

Warning: This is an old version of the data standard. See latest version.

Credits for the project will be included here.

1.1.2 Examples

Warning: This is an old version of the data standard. See latest version.

Examples will be given here shortly.

1.1.3 Governance (stub)

> **Warning:** This is an old version of the data standard. See latest version.

The governance arrangements for the standard will be documented here.

1.1.4 Identifiers

> **Warning:** This is an old version of the data standard. See latest version.

Statement ids

Each statement must have a unique id. This id must be globally unique: such that no two statements from the same organisation, or from different organisations, could ever have the same identifier.

Once published, statements must be immutable. This means any time the underlying record changes, a new statement id should be generated.

Suggested strategies for assigning ids to statements include:

- Generating a UUID for each statement, storing this in internal systems, and updating it whenever the relevant record(s) that make up a statement are updated;
- Generating a UUID as a prefix, and appending a local record identifier, and version identifier to it;
- Assigning a URI in a domain controlled by the publisher to each statement.

Whilst the schema is agnostic as to the exact strategy that data publishers use to generate statement ids, it enforces a minimum length of 32 characters (the length of a hexidecimal UUID) in order to avoid use of ids that are likely to fail a uniqueness test.

Identifying people, companies and other entities

To create a link between statements, and the real-world organisations and people they relate to, statements may include a range of identifying information. We use a common identifier object, with two required properties, and one optional property.

- **scheme** must be a value from a codelist of known identifier sources. Separate codelists exist for entities and persons.
- **id** must be the value assigned to the relevant entity or person in that scheme; ** **uri** may be used to provide a canonical URI from this scheme.

For example, if a source system holds:

- A registered company number; and
- A VAT number;

for a company, two entries could be created in the `entities.identifiers` array, as in the example below:

```
[
    {
        "scheme":"GB-COH",
        "id":"012345678"
    },
    {
        "scheme":"GB-VAT",
        "id":"65251235"
    }
]
```

Entity Identifiers

The values for scheme within an entity statement identifier should be drawn from the http://org-id.guide codelist.

Where the publisher is providing an internal identifier, the publisher should either:

- Publish their full list of internal identifiers, and register this list with the http://org-id.guide codelist; or
- Use MISC-{Publisher_Name} as the scheme

Person Identifiers

The values for scheme within a person statement should be based on the following pattern:

{JURISDICTION}-{TYPE}

Where jurisdiction is expressed using thee extended ISO 3-digit country codes list proposed by in ICAO Document 9303 §5 (pages 22-29).

For example, a passport number from Afghanistan would have the scheme:

AFG-PASSPORT-{NUMBER}

Where the publisher is providing an internal identifier, these should use 'MISC-{Publisher_Name}' as the scheme.

> **Warning:** When using BODS to provide open data, it is important to ensure any person identifiers are suitable for publication under national laws and data protection frameworks.
>
> Most of the identifier types listed below **are not** suitable for publication as part of an open dataset.

The following identification types are currently documented. Suggestions for new types should be made through the issue tracker.

PASSPORT

Passport numbers should follow the format of the identifier (second) line in a machine-readable passport (see Appendix B to Part 4 of ICAO Doc 9303) including at least the document number.

Parsers should be able to extract the document number from the first 9 characters, and to access any subsequent information supplied according to the ICAO format.

IDCARD

Country ID card systems vary. Where specific guidance on including numbers from a particular jurisdiction is required, this may be included here.

1.1.5 Overview (stub)

> **Warning:** This is an old version of the data standard. See latest version.

The standard can be used to guide:

- Data collection
- Data publication

This documentation will need to provide relevant pointers for different user groups.

1.1.6 Provenance Information

> **Warning:** This is an old version of the data standard. See latest version.

Note: This page provide work-in-progress background documentation on the provenance approach taken in the standard.

Design considerations

It is important to have access to provenance information about each of the statements made as part of a beneficial ownership disclosure.

Provenance information may be used in a number of ways, including:

- Identifying the source of information, and how it can be corrected;
- Deciding whether or not to trust a particular source of information;
- Signposting the documentary evidence on which data is based;
- Describing the ways in which data has been modified by source systems;

Any particular statement of provenance may have a range of scopes, including:

- **All the statements in a particular file**. For example, to describe that the statements were downloaded from the OpenOwnership.org database;
- **A group of statements**. For example, to describe the individual responsible for submitting information about a particular set of statements describing ownership of a single firm.
- **A single beneficial ownership statement** - made up of entity, person and qualification statements. For example, to describe the point at which disclosure was made, and the steps taken to verify the information.
- **A single person statement**. For example, to describe how the information was obtained, and to link to any supporting documentation or verification of identification.

These scopes are nested. For example, a person statement might be referenced within a beneficial ownership statement, within a group of statements, and within a particular file - and the provenance information from each of these scopes should be taken to apply to that person statement.

Modelling

Following the PROV-DM Provenance Data Model we model provenance in terms of Activities, Entities and Agents.

A collection of statements, a beneficial ownership statement, and the individual statements that make this up are all considered to be **entities**.

Each entity was derived from some **source** (also, in PROV-DM terms, an entity).

This source will have been generated by some **activity**, such as:

- A self-declaration by an individual agent;
- Extraction of information from an existing register;
- Primary research using public documents or news sources;
- Verification of identity using official documentation;
- and so on.

For each source there will be at least one associated **agent** who was involved, such as:

- The person filling in the form;
- The researcher compiling documentation; or
- The organisation responsible for validating documents.

An source may, itself, be derived from some other source as it's input. For example, when a validation process draws upon documents orginally submitted by an individual.

Provenance block

The provenance building block of the schema can be attached at the `statementGroup`, `beneficialOwnershipStatement` or individual entity, person and qualification statement levels.

Provenance statements can also be chained together using the `derivedFrom` property.

In PROV-DM terms, all the properties within a provenance block attach to the statement they are nested within (i.e. asserting that this statement `wasAtributedTo` or `wasGeneratedBy`).

Field Name	Description	Format
id	See *ID*	Object
type:	Source type: What kind of source is this? [ToDo: Identify an appropriate codelist for this field] List options: ['official-records', 'unverified-submission', 'verified-submission']	string
attributedTo	Attributed to: Which agent (individual, organisation or software process) was responsible for directly contributing this source. See *attributedTo*	Object
generatedBy	Generated by: Which activity led to the creation of this source? See *generatedBy*	Object
primarySource	Primary source: A link to a primary source. This may be a resolvable URI, or some other identifier for the source.	uri string
derivedFrom	Derived from: If this source was derived from a prior source either provide the identifier of a provenanceStatement about that prior source, or nest the provenace statement here. One of *ProvenanceStatement* or *StatementReference*	
replacesStatement	See *ReplacesStatement*	Object

Note: How should applications interpret the nesting of provenance information?

For example: does a provenanceStatement attached to a statementGroup apply to all the statements within that group?

.wy-table-responsive { margin-bottom: 24px; max-width: 100%; overflow: visible !important; }

.wy-table-responsive th:nth-of-type(1) { width:10%; }

.wy-table-responsive th:nth-of-type(2) { width:10%; }

.wy-table-responsive th:nth-of-type(3) { width:60%; }

.wy-table-responsive th:nth-of-type(4) { width:10%; }

.wy-table-responsive th:nth-of-type(5) { width:10%; }–>

1.1.7 Schema

Warning: This is an old version of the data standard. See latest version.

The beneficial ownership standard is made up of two parts:

- A data schema that sets out how beneficial ownership data MUST or SHOULD be formatted for interoperability, and that describes the fields of data that systems MUST or SHOULD provide.
- A set of implementation recommendations that describe the way in which beneficial ownership data SHOULD be collected and published.

Attention: This is the first **rough draft** of the schema. It is a living document, and undergoing constant updates.

It currently contains a draft **structure** and **fields** but does not yet specify any constraints or explicit required fields.

Comments are inviting using hypothes.is annotations (see sidebar on right-hand side), or GitHub issues (https://github.com/openownership/data-standard/issues/) before 20th March.

Conceptual model

The conceptual model for the standard was developed in late 2016/early 2017 and is documented here.

We model information on beneficial ownership in terms of a collection of statements. Each statement represents the assertions made by a particular agent at a particular point in time.

It is up to data consumers to decide which statements to trust, and to reconcile the identity of the entities and persons described in those statements based on the identifying information contained within each statement.

This abstraction is important to represent the reality of how data is provided, to support integration of data from different systems and bi-temporal modelling, and to recognise that any dataset may contain overlapping or conflicting claims about ownership and control that need to be resolved in application specific ways.

Schema browser

The draft Beneficial Ownership Data Standard is defined using JSON Schema 0.4. The structured schema can be accessed on GitHub or explored using the viewer below.

Serializations

We have currently modelled the schema with the option for:

- (1) Entity, person, qualification and provenance statements to be nested inside a beneficial ownership statement;
- (2) Each kind of statement to be provided at the same level of heiarchy, with a cross-reference between them;

This second option is sketched out with a view of serialisations that may make use of the JSON Lines format for sharing or streaming large quantities of statements, rather than enclosing all statements ot be exchanged in a single object.

Sections

The following tables are generated from the schema, and outline the different components of the data model.

Statement Groups

At the top level of any structured file is always an array of `statementGroups`.

Field Name	Description	Format
statementGroups	Statement group: A statement group is used to collect together statements relating to a particular disclosure, company or individual. Statement groups are a logical grouping designed to limit duplication of provenance information, and bring together statements that contain cross references. Where statements in a statementGroup cross-references to other statements, those statements MUST also be contained within the group. See *statementGroups*	Object

Each statementGroup MUST include an array of one or more `beneficialOwnershipStatements` and, where a cross-reference publication pattern is followed, may include arrays of other statements.

Field Name	Description	Format
id	Statement group identifier: An optional globally unique and persistent identifier for this statement group.	string
beneficialOwnershipStatements	Beneficial ownership statements: A collection of statements that describe the relationship between legal entities or between legal entities and natural persons, or that explain the non-availability of this information. Entity, person, qualification and provenance statements may be embeded within these statements, or provided in the neighbouring arrays and cross-referenced. See *BeneficialOwnershipStatement* section for further details.	Object Array
entityStatements	Entity statements: A collection of statements that describe legal persons or arrangements. Cross-referenced within beneficial ownership statements. See *EntityStatement* section for further details.	Object Array
personStatements	Person statements: A collection of statements that describe natural persons. Cross-referenced within beneficial ownership statements. See *EntityStatement* section for further details.	Object Array
qualificationStatements	Qualification statements: A collection of statements that qualify a beneficial ownership statement. Cross-referenced within beneficial ownership statements. See *QualificationStatement* section for further details.	Object Array
provenanceStatements	Provenance statements: A collection of provenance statements. See *ProvenanceStatement* section for further details.	Object Array
provenance	One of *ProvenanceStatement* or *StatementReference*	
replacesStatementGroup	Replaces statement group: If this statement group replaces all the statements from a previously published group, provide the globally unique identifier for the previous group here. Consuming applications are advised to mark all statements from the identified group as no longer active.	string

BeneficialOwnershipStatement

A beneficial ownership statement is made up of statements about an entity, an interestedParty (either an entity or a person), and detailes of the interest. Additionally, qualifications on this, provenance and versioning information can be provided.

Field Name	Description	Format
id	See *ID*	Object
date	See *StatementDate*	Object
entity	One of *StatementReference* or *EntityStatement*	
interestedParty	One of *EntityStatement* or *PersonStatement* or *StatementReference*	
interests	Interests: A description of the interests held by the interestedParty covered by this statement in the entity covered by this statement. See *Interest* section for further details.	Object Array
qualifications	Qualifications: A qualification statement can be used to record any additional information about this Beneficial Ownership Statement, including information about any reasons non-disclosure of information.	Array
provenance	One of *ProvenanceStatement* or *StatementReference*	
replacesStatement	See *ReplacesStatement*	Object

Interest

Field Name	Description	Format
type	Type of interest: A codelist value indicating the nature of the interest. List options: ['shareholding', 'voting-rights', 'appointment-of-board', 'influence-or-control']	string
interestLevel	Interest level: Is this interest held directly or indirectly? List options: ['direct', 'indirect', 'unknown']	string
details	Details: This field may be used to provide the local name given to this kind of interest, or any further semi-structured or unstructured information to clarify the nature of the interest held.	string
share	Percentage share: Where an exact percentage is available, this should be given, and maximum and minimum values set to the same as the exact percentage. Otherwise, maximum and minimum can be used to record the range into which the share of this kind of interest falls. See *share*	Object
startDate	State date: When did this interest first occur. Please provide as precise a date as possible in ISO 8601 format. When only the year or year and month is known, these can be given as YYYY or YYYY-MM.	string
endDate	End date: When did this interest cease. Please provide as precise a date as possible in ISO 8601 format. When only the year or year and month is known, these can be given as YYYY or YYYY-MM.	string

Share

Field Name	Description	Format
exact	Exact share: The exact share of this interest held (where available).	number
maximum	Maximum share: The upper bound of the share of this interest known to be held.	number
minimum	Minimum share: The lower bound of the share of this interest known to be held.	number
exclusiveMinimum	Exclusive minimum: If exclusiveMinimum is true, then the share is at least greater than the minimum value given. E.g. if minimum is 25, the share is at least 25.1, and not simply 25.	boolean
exclusiveMaximum	Exclusive maximum: If exclusiveMaximum is true, then the share is at least less than the maximum value given. E.g. if maximum is 50, the share is less than 49.999, and not simply 50.	boolean

EntityStatement

Field Name	Description	Format
id	See *ID*	Object
type	Type: What kind of entity is this? The 'registeredEntity' code covers any legal entity created through an act of official registration, usually resulting in an identifier being assigned to the entity. The 'arrangement' code covers artificial entities, described in the data model for the purpose of associating one or more natural or legal persons together in an ownership or control relationship. List options: ['registeredEntity', 'arrangement']	string
date	See *StatementDate*	Object
name	Name: The declared name of this entity.	string
jurisdiction	Jurisdiction: The jurisdiction in which this entity is registered, expressed using an ISO ISO_3166-2 2-Digit country code, or ISO_3166-2 sub-division code, where the sub-division in question (e.g. a sub-national state or region) has relevant jurisdiction over the registration of operation of this entity.	string
identifiers	Identifiers: One or more official identifiers for this entity. Where available, official registration numbers should be provided. See *Identifier* section for further details.	Object Array
createdDate	Created date: When was this entity first created or registered. Please provide as precise a date as possible in ISO 8601 format. When only the year or year and month is known, these can be given as YYYY or YYYY-MM.	string
endDate	End date: If this entity is no longer active, provide the date on which it ceased. Please provide as precise a date as possible in ISO 8601 format. When only the year or year and month is known, these can be given as YYYY or YYYY-MM.	string
addresses	Addresses: One or more addresses for this entity. See *Address* section for further details.	Object Array
uri	URI: Where a persisten URI is available for this entity this should be included.	uri string
provenance	One of *ProvenanceStatement* or *StatementReference*	
replacesStatement	See *ReplacesStatement*	Object

PersonStatement

Field Name	Description	Format
id	See *ID*	Object
type	Type: The ultimate beneficial owner of a legal entity is always a natural person. List options: ['naturalPerson']	string
date	See *StatementDate*	Object
name	Name: The full name of this person.	string
alternateNames	Alternate names: Other known names for this individual. See *AlternateName* section for further details.	Object Array
identifiers	Identifiers: One or more official identifiers for this perrson. Where available, official registration numbers should be provided. See *Identifier* section for further details.	Object Array
nationalities	Nationality: An array of ISO 2-Digit country codes representing nationalities held by this individual.	Array
placeOfResidence	See *Address*	Object
placeOfBirth	See *Address*	Object
birthDate	Created date: The date of birth for this individual. Please provide as precise a date as possible in ISO 8601 format. When only the year or year and month is known, these can be given as YYYY or YYYY-MM.	string
deathDate	End date: If this individual is no longer alive, provide their date of death. Please provide as precise a date as possible in ISO 8601 format. When only the year or year and month is known, these can be given as YYYY or YYYY-MM.	string
addresses	Addresses: One or more addresses for this entity. See *Address* section for further details.	Object Array
provenance	One of *ProvenanceStatement* or *StatementReference*	
replacesStatement	See *ReplacesStatement*	Object

AlternateName

Field Name	Description	Format
type	Type: What kind of alternative name is this? Select from 'translation', 'formerName','birth', and 'alias'. List options: ['translation', 'former', 'alias', 'birth']	string
fullName	Full name: The full name contains the complete name of a person as one string.	string
familyName	Family name: A family name is usually shared by members of a family. This attribute also carries prefixes or suffixes which are part of the Family Name, e.g. 'de Boer', 'van de Putte', 'von und zu Orlow'. Multiple family names, such as are commonly found in Hispanic countries, are recorded in the single Family Name field so that, for example, Miguel de Cervantes Saavedra's Family Name would be recorded as 'Cervantes Saavedra.'	string
givenName	Given names: A given name, or multiple given names, are the denominator(s) that identify an individual within a family. These are given to a person by his or her parents at birth or may be legally recognised as 'given names' through a formal process. All given names are ordered in one field so that, for example, the given name for Johan Sebastian Bach is 'Johan Sebastian.'	string
patronymicName	Patronymic Name: Patronymic names are important in some countries. Iceland does not have a concept of family name in the way that many other European countries do, for example. In Bulgaria and Russia, patronymic names are in every day usage, for example, the 'Sergeyevich' in 'Mikhail Sergeyevich Gorbachev'	string

QualificationStatement

Field Name	Description	Format
id	See *ID*	Object
date	See *StatementDate*	Object
type	Type of qualification statement: List options: ['non-disclosure', 'redaction', 'restrictions-on-control']	string
description	Description: A description of this qualification	string
provenance	One of *ProvenanceStatement* or *StatementReference*	
replacesStatement	See *ReplacesStatement*	Object

ProvenanceStatement

See *the provenance pages* for a discussion of provenance modelling.

Field Name	Description	Format
id	See *ID*	Object
type:	Source type: What kind of source is this? [ToDo: Identify an appropriate codelist for this field] List options: ['official-records', 'unverified-submission', 'verified-submission']	string
attributedTo	Attributed to: Which agent (individual, organisation or software process) was responsible for directly contributing this source. See *attributedTo*	Object
generatedBy	Generated by: Which activity led to the creation of this source? See *generatedBy*	Object
primarySource	Primary source: A link to a primary source. This may be a resolvable URI, or some other identifier for the source.	uri string
derivedFrom	Derived from: If this source was derived from a prior source either provide the identifier of a provenanceStatement about that prior source, or nest the provenace statement here. One of *ProvenanceStatement* or *StatementReference*	
replacesStatement	See *ReplacesStatement*	Object

StatementReference

Field Name	Description	Format
type	Type: What type of statement is being referred to? List options: ['entityStatement', 'personStatement', 'provenanceStatement', 'qualificationStatement']	string
id	ID: The identifier of the statement being referenced.	string
uri	URI: A persistent URI for the statement being referenced.	uri string

Common components

The following components are used at a number of points in the schema

Address

Field Name	Description	Format
type	Type: What type of address is this? List options: ['placeOfBirth', 'home', 'residence', 'registered', 'service', 'alternative']	string
address	Address: The address, with each line or component of the address separated by a line-break or comma. This field may also include the postal code.	string
postCode	Postcode: The postal code for this address.	string
country	Country: The ISO 2-Digit county code for this address.	string

Identifier

The identifier component is used to connect a statement to the person or entity that it refers to, using one or more existing known identifiers.

Field Name	Description	Format
id	ID: The identifier for this entity as provided in the declared scheme.	string
scheme	Scheme: For entity statements, the scheme should be a entry from the org-id.guide codelist. For person statements, recognised values include 'passport', 'internal' and 'id-card'.	string
uri	URI: Where this identifier has a canonical URI this may be included	uri string

Date

See https://github.com/openownership/data-standard/issues/12 for a discussion of handling fuzzy dates.

Our current schema uses a regular expression to allow YYYY, YYYY-MM, YYYY-MM-DD or full datetimes.

ID

Publishers MUST generate globally unique and persisent identifiers for each statement.

These SHOULD start with a uuid to avoid any clash between identifiers from different publishers, and MAY be suffixed with additional characters to distinguish versions of a statement as required by local implementations.

In many implementation scenarios, it will be appropriate to simply generate a distinct uuid for each statement.

Publication and use considerations

This section outlines considerations for publishers and consumers of the data

Immutability of statements

Statements are considered immutable. If a field is updated, this should be considered to create a new statement, with a new identifier.

Updating statements

Where a `statementGroup` or `statement` replaces a previous statement this should be explicitly declared using a `replacesStatementGroup` or `replacesStatement` property.

1.1.8 Serialization (stub)

> **Warning:** This is an old version of the data standard. See latest version.

Information on serialization approaches for the project will be included here.

1.2 Partners and funders

The initial development of the Beneficial Ownership Data Standard is funded through support for the Open Ownership project from the UK Department for International Development. OpenOwnership is a project of Transparency International, OpenCorporates, One, the Open Contracting Partnership, the World Wide Web Foundation, Global Witness and The B Team

This draft has been developed by Open Data Services Co-operative and OpenCorporates

1.3 Contact

For more details about the OpenOwnership project, please contact the project coordinator, Zosia Sztykowski