Data on the Web Best Practices

W3C Recommendation 31 January 2017

This version:

https://www.w3.org/TR/2017/REC-dwbp-20170131/

Latest published version:

https://www.w3.org/TR/dwbp/

Latest editor's draft:

http://w3c.github.io/dwbp/bp.html

Implementation report:

http://w3c.github.io/dwbp/dwbp-implementation-report.html

Previous version:

https://www.w3.org/TR/2016/PR-dwbp-20161215/

Editors:

Bernadette Farias Lóscio, CIn - UFPE, Brazil

Caroline Burle, NIC.br, Brazil

Newton Calegari, NIC.br, Brazil

Contributors:

Annette Greiner

Antoine Isaac

Carlos Iglesias

Carlos Laufer

Christophe Guéret

Deirdre Lee

Doug Schepers

Eric G. Stephan

Eric Kauz

Ghislain A. Atemezing

Hadley Beeman

Ig Ibert Bittencourt

João Paulo Almeida

Makx Dekkers

Peter Winstanley

Phil Archer

Riccardo Albertoni

Sumit Purohit

Yasodara Córdova

Changes:

Change History

Diff to previous version

Please check the **errata** for any errors or issues reported since publication.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

This document provides Best Practices related to the publication and usage of data on the Web designed to help support a self-sustaining ecosystem. Data should be discoverable and understandable by humans and machines. Where data is used in some way, whether by the originator of the data or by an external party, such usage should also be discoverable and the efforts of the data publisher recognized. In short, following these Best Practices will facilitate interaction between publishers and consumers.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

The Data on the Web Best Practices Working Group was chartered to develop the open data ecosystem, facilitating better communication between developers and publishers; to provide guidance to publishers that will improve consistency in the way data is managed, thus promoting the reuse of data; to foster trust in the data among developers, whatever technology they choose to use, increasing the potential for genuine innovation. This best practice document is complemented by the Data Quality and Dataset Usage vocabularies.

This document was published by the Data on the Web Best Practices Working Group as a Recommendation. If you wish to make comments regarding this document, please send them to public-dwbp-comments@w3.org (subscribe, archives). All comments are welcome.

Please see the Working Group's implementation report.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 September 2015 W3C Process Document.

Table of Contents

1. Introduction

This section is non-normative.

The Best Practices described below have been developed to encourage and enable the continued expansion of the Web as a medium for the exchange of data. The growth in online sharing of open data by governments across the world [OKFN-INDEX] [ODB], the increasing online publication of research data encouraged by organizations like the Research Data Alliance [RDA], the harvesting, analysis and online publishing of social media data, crowd-sourcing of information, the increasing presence on the Web of important cultural heritage collections such as at the Bibliothèque nationale de France [BNF] and the sustained growth in the Linked Open Data Cloud [LODC], provide some examples of this growth in the use of Web for publishing data.

However, this growth is not consistent in style and in many cases does not make use of the full potential of the Open Web Platform's ability to link one fact to another, to discover related resources and to create interactive visualizations.

In broad terms, data publishers aim to share data either openly or with controlled access. Data consumers (who may also be producers themselves) want to be able to find, use and link to the data, especially if it is accurate, regularly updated and guaranteed to be available at all times. This creates a fundamental need for a common understanding between data publishers and data consumers. Without this agreement, data publishers' efforts may be incompatible with data consumers' desires.

The openness and flexibility of the Web create new challenges for data publishers and data consumers, such as how to represent, describe and make data available in a way that it will be easy to find and to understand. In contrast to conventional databases, for example, where there is a single data model to represent the data and a database management system (DBMS) to control data access, data on the Web allows for the existence of multiple ways to represent and to access data. For more details about the challenges see the section Data on the Web Challenges.

In this context, it becomes crucial to provide guidance to publishers that will improve consistency in the way data is managed. Such guidance will promote the reuse of data and foster trust in the data among developers, whatever technology they choose to use, increasing the potential for genuine innovation.

Not all data and metadata should be shared openly, however. Security, commercial sensitivity and, above all, individuals' privacy need to be taken into account. It is for data publishers to determine policy on which data should be shared and under what circumstances. Data sharing policies are likely to assess the exposure risk and determine the appropriate security measures to be taken to protect sensitive data, such as secure authentication and authorization.

Depending on circumstances, sensitive information about individuals might include full name, home address, email address, national identification number, IP address, vehicle registration plate number, driver's license number, face, fingerprints, or handwriting, credit card numbers, digital identity, date of birth, birthplace, genetic information, telephone number, login name, screen name, nickname, health records etc. Although it is likely to be safe to share some of that information openly, and even more within a controlled environment, publishers should bear in mind that combining data from multiple sources may allow inadvertent identification of individuals.

A general Best Practice for publishing Data on the Web is to use standards. Different types of organizations specify standards that are specific to the publishing of datasets related to particular domains & applications, involving communities of users interested in that data. These standards define a common way of communicating information among the users of these communities. For example, there are two standards that can be used to publish transport timetables: the General Transit Feed Specification [GTFS] and the Service Interface for Real Time Information [SIRI]. These specify, in a mixed way, standardized terms, standardized data formats and standardized data access. Another general Best Practice is to use Unicode for handling character and string data. Unicode improves multilingual text processing and makes software localization easier.

Best Practices cover different aspects related to data publishing and consumption, like data formats, data access, data identifiers and metadata. In order to delimit the scope and elicit the required features for Data on the Web Best Practices, the DWBP working group compiled a set of use cases [DWBP-UCR] that represent scenarios of how data is commonly published on the Web and how it is used. The set of requirements derived from these use cases were used to guide the development of the Best Practices, which are domain & application independent. However, they can be extended or complemented by other Best Practices documents or standards that cover more specialized contexts. Considering vocabularies, for example, the W3C Best Practices for Publishing Linked Data [LD-BP] is a useful reference. There are specific recommendations for expressing licenses and other permissions and obligations statements in ODRL [ODRL-model], a suite of standards related to provenance [PROV-Overview], and the best practices presented here have been extended to cover more specific advice around the discoverability, accessibility and interoperability of spatial and temporal data by the Spatial Data on the Web Best Practices [SDW-BP].

Whilst DWBP recommends the use of Linked Data, it also promotes best practices for data on the Web in other open formats such as CSV. Methods for sharing tabular data, including CSV files, in a way that maximizes the potential of the Web to make links between data points, are described in the Tabular Data Primer [Tabular-Data-Primer].

In order to encourage data publishers to adopt the DWBP, a number of distinct benefits were identified: comprehension; processability; discoverability; reuse; trust; linkability; access; and interoperability. They are described and related to the Best Practices in the section Best Practices Benefits.

2. Audience

This section is non-normative.

This document sets out Best Practices tailored primarily for those who publish data on the Web. The Best Practices are designed to meet the needs of information management staff, developers, and wider groups such as scientists interested in sharing and reusing research data on the Web. While data publishers are our primary audience, we encourage all those engaged in related activities to become familiar with it. Every attempt has been made to make the document as readable and usable as possible while still retaining the accuracy and clarity needed in a technical specification.

Readers of this document are expected to be familiar with some fundamental concepts of the architecture of the Web [WEBARCH], such as resources and URIs, as well as a number of data formats. The normative element of each Best Practice is the *intended outcome*. Possible implementations are suggested and, where appropriate, these recommend the use of a particular technology. A basic knowledge of vocabularies and data models would be helpful to better understand some aspects of this document.

3. Scope

This section is non-normative.

This document is concerned solely with Best Practices that:

- are specifically relevant to data published on the Web;
- encourage publication or reuse of data on the Web;
- can be tested by machines, humans or a combination of the two.

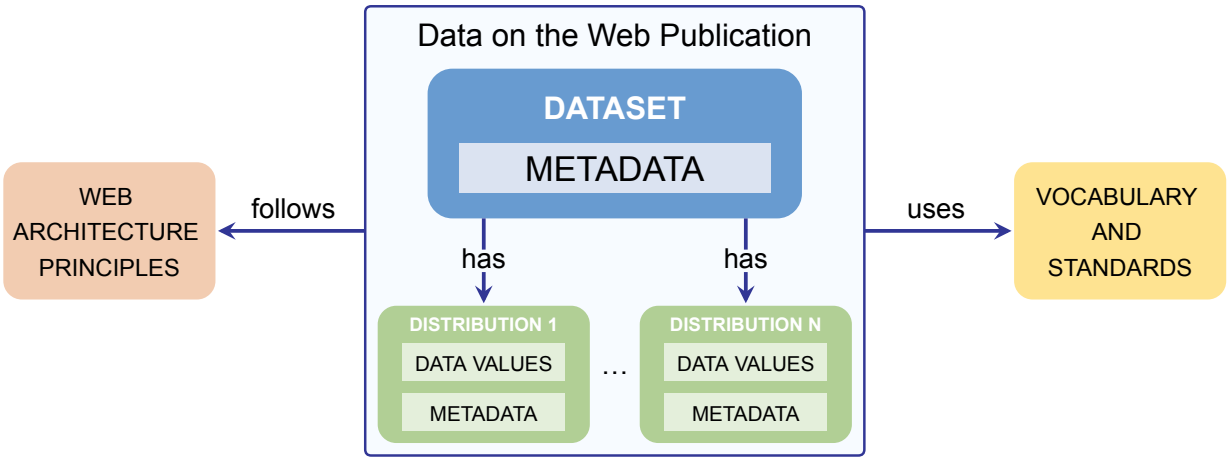
As noted above, whether a Best Practice has or has not been followed should be judged against the *intended outcome*, not the *possible approach to implementation* which is offered as guidance. A best practice is always subject to improvement as we learn and evolve the Web together.

4. Context

This section is non-normative.

The following diagram illustrates the context considered in this document. In general, the Best Practices proposed for publication and usage of Data on the Web refer to datasets and distributions. Data is published in different distributions, which are specific physical form of a dataset. By data, "we mean known facts that can be recorded and that have implicit meaning" [Navathe]. These distributions facilitate the sharing of data on a large scale, which allows datasets to be used for several groups of data consumers , without regard to purpose, audience, interest, or license. Given this heterogeneity and the fact that data publishers and data consumers may be unknown to each other, it is necessary to provide some information about the datasets and distributions that may also contribute to trustworthiness and reuse, such as: structural metadata, descriptive metadata, access information, data quality information, provenance information, license information and usage information.

An important aspect of publishing and sharing data on the Web concerns the architectural basis of the Web [WEBARCH]. A relevant aspect of this is the identification principle that says that URIs should be used to identify resources. In our context, a resource may be a whole dataset or a specific item of given dataset. All resources should be published with stable URIs, so that they can be referenced and make links, via URIs, between two or more resources. Finally, to promote the interoperability among datasets it is important to adopt data vocabularies and standards.



5. Namespaces

This section is non-normative.

The following namespace prefixes are used throughout this document.

Prefix	Namespace IRI	Description
dcat	http://www.w3.org/ns/dcat#	Data Catalog Vocabulary (DCAT)

dct	http://purl.org/dc/terms/	Dublin Core Metadata Initiative (DCMI) Metadata Terms
dqv	http://www.w3.org/ns/dqv#	DWBP Data Quality Vocabulary (DQV)
duv	http://www.w3.org/ns/duv#	DWBP Dataset Usage Vocabulary (DUV)
foaf	http://xmlns.com/foaf/0.1/	Friend of a Friend (FOAF) Vocabulary
oa	http://www.w3.org/ns/oa#	Web Annotation Ontology
owl	http://www.w3.org/2002/07/owl#	Web Ontology Language (OWL)
pav	http://pav-ontology.github.io/pav/	Provenance, Authoring and Versioning (PAV)
prov	http://www.w3.org/ns/prov#	Provenance Ontology (PROV)
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework (RDF)
rdfs	http://www.w3.org/2000/01/rdf-schema#	RDF Schema vocabulary (RDFS)
skos	http://www.w3.org/2004/02/skos/core#	Simple Knowledge Organization System (SKOS)

Namespaces used in the document

6. Best Practices Template

This section presents the template used to describe Data on the Web Best Practices.

Best Practice Template

Short description of the BP

Why

This section answers two crucial questions:

- Why this is specifically relevant to publishing or reusing data on the Web?
- How does this encourage publication or reuse of data on the Web?

A full text description of the problem addressed by the Best Practice may also be provided. It can be any length but is likely to be no more than a few sentences.

Intended Outcome

What it should be possible to do when a data publisher follows the Best Practice.

Possible Approach to Implementation

A description of a possible implementation strategy is provided. This represents the best advice available at the time of writing but specific circumstances and future developments may mean that alternative implementation methods are more appropriate to achieve the intended outcome.

How to Test

Information on how to test the BP has been met. This might or might not be machine testable.

Evidence

Information about the relevance of the BP. It is described by one or more relevant requirements as documented in the Data on the Web Best Practices Use Cases & Requirements document [DWBP-UCR]

Benefits

A benefit represents an improvement in the way how datasets are available on the Web. A Best Practice can have one or more benefits.

7. Best Practices Summary

Best Practice 1: Provide metadata

Best Practice 2: Provide descriptive metadata

Best Practice 3: Provide structural metadata

Best Practice 4: Provide data license information

Best Practice 5: Provide data provenance information

Best Practice 6: Provide data quality information

Best Practice 7: Provide a version indicator

Best Practice 8: Provide version history

Best Practice 9: Use persistent URIs as identifiers of datasets

Best Practice 10: Use persistent URIs as identifiers within datasets

Best Practice 11: Assign URIs to dataset versions and series

Best Practice 12: Use machine-readable standardized data formats

Best Practice 13: Use locale-neutral data representations

Best Practice 14: Provide data in multiple formats

Best Practice 15: Reuse vocabularies, preferably standardized ones

Best Practice 16: Choose the right formalization level

Best Practice 17: Provide bulk download

Best Practice 18: Provide Subsets for Large Datasets

Best Practice 19: Use content negotiation for serving data available in multiple formats

Best Practice 20: Provide real-time access

Best Practice 21: Provide data up to date

Best Practice 22: Provide an explanation for data that is not available

Best Practice 23: Make data available through an API

Best Practice 24: Use Web Standards as the foundation of APIs

Best Practice 25: Provide complete documentation for your API

Best Practice 26: Avoid Breaking Changes to Your API

Best Practice 27: Preserve identifiers

Best Practice 28: Assess dataset coverage

Best Practice 29: Gather feedback from data consumers

Best Practice 30: Make feedback available

Best Practice 31: Enrich data by generating new data

Best Practice 32: Provide Complementary Presentations

Best Practice 33: Provide Feedback to the Original Publisher

Best Practice 34: Follow Licensing Terms

Best Practice 35: Cite the Original Publication

8. The Best Practices

This section contains the Best Practices to be used by data publishers in order to help them and data consumers to overcome the different challenges faced when publishing and consuming data on the Web. One or more Best Practices were proposed for each one of the challenges, which are described in the section Data on the Web Challenges.

Each BP is related to one or more requirements from the Data on the Web Best Practices Use Cases & Requirements document [DWBP-UCR] which guided their development. Each Best Practice has at least one of these requirements as evidence of its relevance.

8.1 Running Example

> Adrian works for the Transport Agency of MyCity and is in charge of publishing data about public transport. Adrian wants to publish this data for different types of data consumers such as developers interested on creating applications and also for software agents. It is important that both humans and software agents can easily understand and process the data which should be kept up to date and be easily discoverable on the Web.

RDF examples of the application of some Best Practices are shown using Turtle [Turtle] or JSON-LD [JSON-LD].

8.2 Metadata

The Web is an open information space, where the absence of a specific context, such a company's internal information system, means that the provision of metadata is a fundamental requirement. Data will not be discoverable or reusable by anyone other than the publisher if insufficient metadata is provided. Metadata provides additional information that helps data consumers better understand the meaning of data, its structure, and to clarify other issues, such as rights and license terms, the organization that generated the data, data quality, data access methods and the update schedule of datasets. Publishers are encouraged to provide human-readable information in multiple languages, and, as much as possible, provide the information in the language(s) that the intended users will understand.

Metadata can be used to help tasks such as dataset discovery and reuse, and can be assigned considering different levels of granularity from a single property of a resource to a whole dataset, or all datasets from a specific organization. Metadata can also be of different types. These types can be classified in different taxonomies, with different grouping criteria. For example, a specific taxonomy could define three metadata types according to descriptive, structural and administrative features. A different taxonomy could define metadata types with a scheme according to tasks where metadata are used, for example, discovery and reuse.

Best Practice 1: Provide metadata

Provide metadata for both human users and computer applications.

Why

Providing metadata is a fundamental requirement when publishing data on the Web because data publishers and data consumers may be unknown to each other. Then, it is essential to

provide information that helps human users and computer applications to understand the data as well as other important aspects that describes a dataset or a distribution.

Intended Outcome

Humans will be able to understand the metadata and computer applications, notably user agents, will be able to process it.

Possible Approach to Implementation

Possible approaches to provide *human-readable metadata:*

- to provide metadata as part of an HTML Web page
- to provide metadata as a separate text file

Possible approaches to provide *machine-readable metadata:*

- machine-readable metadata may be provided in a serialization format such as Turtle and JSON, or it can be embedded in the HTML page using [HTML-RDFA] or [JSON-LD]. If multiple formats are published separately, they should be served from the same URL using content negotiation and made available under separate URIs, distinguished by filename extension. Maintenance of multiple formats is best achieved by generating each available format on the fly based on a single source of the metadata.
- when defining machine-readable metadata, reusing existing standard terms and popular vocabularies are strongly recommended. For example, Dublin Core Metadata (DCMI) terms [DCTERMS] and Data Catalog Vocabulary [VOCAB-DCAT] can be used to provide descriptive metadata. Such vocabularies are designed to be very flexible so it is often helpful to use a specific *profile* of a vocabulary such as the European Commission's DCAT-AP

> EXAMPLE 1
>
> **Human-readable**
>
> Example page with a human-readable description of an available dataset.
>
> **Machine-readable**
>
> Example file with a machine-readable description of an available dataset.

How to Test

Check if human-readable metadata is available.

Check if the metadata is available in a valid machine-readable format and without syntax error.

Evidence

Relevant requirements: R-MetadataAvailable, R-MetadataDocum, R-MetadataMachineRead

Benefits

Best Practice 2: Provide descriptive metadata

Provide metadata that describes the overall features of datasets and distributions.

Why

Explicitly providing dataset descriptive information allows user agents to automatically discover datasets available on the Web and it allows humans to understand the nature of the dataset and its distributions.

Intended Outcome

Humans will be able to interpret the nature of the dataset and its distributions, and software agents will be able to automatically discover datasets and distributions.

Possible Approach to Implementation

Descriptive metadata can include the following overall features of a dataset:

- The **title** and a **description** of the dataset.
- The **keywords** describing the dataset.
- The **date of publication** of the dataset.
- The **entity responsible (publisher)** for making the dataset available.
- The **contact point** for the dataset.
- The **spatial coverage** of the dataset.
- The **temporal period** that the dataset covers.
- The **date of last modification** of the dataset.
- The **themes/categories** covered by a dataset.

Descriptive metadata can include the following overall features of a distribution:

- The **title** and a **description** of the distribution.

- The **date of publication** of the distribution.
- The **media type** of the distribution.

The machine-readable version of the descriptive metadata can be provided using the vocabulary recommended by W3C to describe datasets, i.e. the Data Catalog Vocabulary [VOCAB-DCAT]. This provides a framework in which datasets can be described as abstract entities.

EXAMPLE 2

Machine-readable

The example below shows how to use [VOCAB-DCAT] to provide the machine-readable **discovery** metadata for the bus stops dataset (`stops-2015-05-05`). The dataset has one CSV distribution (`stops-2015-05-05.csv`) that is also described using the [VOCAB-DCAT]. The dataset is classified under the domain represented by the relative URI `mobility`. This domain may be defined as part of a set of domains identified by the URI `themes`. To describe both concepts and schema concepts, Adrian used SKOS . To express frequency of update an instance from the Content-Oriented Guidelines developed as part of the W3C Data Cube Vocabulary efforts was used. Adrian chose to describe the spatial and temporal coverage of the example dataset using URIs from Geonames and the Time Intervals dataset from data.gov.uk, respectively.

```
:stops-2015-05-05
    a dcat:Dataset ;
    dct:title "Bus stops of MyCity" ;
    dcat:keyword "transport","mobility","bus" ;
    dct:issued "2015-05-05"^^xsd:date ;
    dcat:contactPoint <http://data.mycity.example.com/transport
    dct:temporal <http://reference.data.gov.uk/id/year/2015> ;
    dct:spatial <http://sws.geonames.org/3399415> ;
    dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
    dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/20
    dcat:theme :mobility ;
    dcat:distribution :stops-2015-05-05.csv
    .

:mobility
    a skos:Concept ;
    skos:inScheme :themes ;
    skos:prefLabel "Mobility"@en ;
    skos:prefLabel "Mobilidade"@pt
    .

:themes
    a skos:ConceptScheme ;
    skos:prefLabel "A set of domains to classify documents"
    .

:stops-2015-05-05.csv
    a dcat:Distribution ;
    dct:title "CSV distribution of stops-2015-05-05 dataset" ;
    dct:description "CSV distribution of the bus stops dataset
    dcat:mediaType "text/csv;charset=UTF-8"
    .
```

Human-readable

Example page with human-readable description of dataset is available.

How to Test

Check if the metadata for the dataset itself includes the overall features of the dataset in a human-readable format.

Check if the descriptive metadata is available in a valid machine-readable format.

Evidence

Relevant requirements: R-MetadataAvailable, R-MetadataMachineRead, R-MetadataStandardized

Benefits

Best Practice 3: Provide structural metadata

Provide metadata that describes the schema and internal structure of a distribution.

Why

Providing information about the internal structure of a distribution is essential for others wishing to explore or query the dataset. It also helps people to understand the meaning of the data.

Intended Outcome

Humans will be able to interpret the schema of a dataset and software agents will be able to automatically process distributions.

Possible Approach to Implementation

Human-readable structural metadata usually provides the properties or columns of the dataset schema.

Machine-readable structural metadata is available according to the format of a specific distribution and it may be provided within separate documents or embedded into the document. For more details, see the links below.

- Tabular data: see Model for Tabular Data and Metadata on the Web
- JSON-LD: see JSON-LD 1.0
- XML: see XML Schema
- Multi-dimensional data: see Data Cube

EXAMPLE 3

Machine-readable

Adrian used the Model for Tabular Data and Metadata on the Web for publishing the CSV distribution of the bus stops dataset (`stops-2015-05-05.csv`). The example below presents the structural metadata for this distribution.

```
{
  "@context": ["https://www.w3.org/ns/csvw", {
    "@language": "en"
  }],
  "url": "http://data.mycity.example.com/transport/dataset/bus/st
  "dct:title": "CSV distribution of stops-2015-05-05 dataset",
  "dcat:keyword": ["bus", "stop", "mobility"],
  "dct:publisher": {
    "schema:name": "Transport Agency of MyCity",
    "schema:url": {
      "@id": "http://example.org"
    }
  },
  "dct:license": {
    "@id": "http://opendefinition.org/licenses/cc-by/"
  },
  "dct:issued": {
    "@value": "2015-05-05",
    "@type": "xsd:date"
  },
  "tableSchema": {
    "columns": [{
      "name": "stop_id",
      "titles": "Identifier",
      "dct:description": "An identifier for the bus stop.",
      "datatype": "string",
      "required": true
    }, {
      "name": "stop_name",
      "titles": "Name",
      "dct:description": "The name of the bus stop.",
      "datatype": "string"
    }, {
      "name": "stop_desc",
      "titles": "Description",
      "dct:description": "A description for the bus stop.",
      "datatype": "string"
    }, {
      "name": "stop_lat",
      "titles": "Latitude",
      "dct:description": "The latitude of the bus stop.",
      "datatype": "number"
    }, {
      "name": "stop_long",
      "titles": "Longitude",
      "dct:description": "The longitude of the bus stop.",
      "datatype": "number"
    }, {
```

```
      "name": "zone_id",
      "titles": "ZONE",
      "dct:description": "An identifier for the zone where the bu
      "datatype": "string"
    }, {
      "name": "stop_url",
      "titles": "URL",
      "dct:description": "URL that identifies the bus stop.",
      "datatype": "anyURI"
    }],
    "primaryKey": "stop_id"
  }
}
```


Human-readable

Example page with human-readable structural metadata is available.

How to Test

Check if the structural metadata is provided in a human-readable format.

Check if the metadata of the distribution includes structural information about the dataset in a machine-readable format and without syntax errors.

Evidence

Relevant requirements: R-MetadataAvailable

Benefits

8.3 Data Licenses

A license is a very useful piece of information to be attached to data on the Web. According to the type of license adopted by the publisher, there might be more or fewer restrictions on sharing and reusing data. In the context of data on the Web, the license of a dataset can be specified within the metadata, or outside of it, in a separate document to which it is linked.

Best Practice 4: Provide data license information

Provide a link to or copy of the license agreement that controls use of the data.

Why

The presence of license information is essential for data consumers to assess the usability of data. User agents may use the presence/absence of license information as a trigger for inclusion or exclusion of data presented to a potential consumer.

Intended Outcome

Humans will be able to understand data license information describing possible restrictions placed on the use of a given distribution, and software agents will be able to automatically detect the data license of a distribution.

Possible Approach to Implementation

Data license information can be available via a link to, or embedded copy of, a human-readable license agreement. It can also be made available for processing via a link to, or embedded copy of, a machine-readable license agreement.

One of the following vocabularies that include properties for linking to a license can be used:

- Dublin Core [DCTERMS] (`dct:license`)
- Creative Commons [CCREL] (`cc:license`)
- schema.org [SCHEMA-ORG] (`schema:license`)
- XHTML [XHTML-VOCAB] (`xhtml:license`)

There are also a number of machine-readable rights languages, including:

- The Creative Commons Rights Expression Language [CCREL]
- The Open Digital Rights Language [ODRL-model]
- The Open Data Rights Statement Vocabulary [ODRS]

EXAMPLE 4

Machine-readable

The CSV distribution of the bus stops dataset (`stops-2015-05-05.csv`) will be published under the Creative Commons Attribution-ShareAlike 3.0 Unported license. The property `dct:license` is used to include this information as part of the distribution metadata. In this case, the license is not written in a machine-readable language, but the property `dct:license` allows a software agent to automatically detect the data license of the distribution.

```
:stops-2015-05-05.csv
    a dcat:Distribution ;
    dct:title "CSV distribution of stops-2015-05-05 dataset" ;
    dct:description "CSV distribution of the bus stops dataset
    dcat:mediaType "text/csv;charset=UTF-8" ;
    dct:license <http://creativecommons.org/licenses/by-sa/3.0/
    .
```

Human-readable

Example page with human-readable data license information of the distribution.

How to Test

Check if the metadata for the dataset itself includes the data license information in a human-readable format.

Check if a user agent can automatically detect or discover the data license of the dataset.

Evidence

Relevant use cases: R-LicenseAvailable, R-MetadataMachineRead, R-LicenseLiability

Benefits

8.4 Data Provenance

The Web brings together business, engineering, and scientific communities creating collaborative opportunities that were previously unimaginable. The challenge in publishing data on the Web is provid-

ing an appropriate level of detail about its origin. The data producer may not necessarily be the data publisher and so collecting and conveying this corresponding metadata is particularly important. Without provenance, consumers have no inherent way to trust the integrity and credibility of the data being shared. Data publishers in turn need to be aware of the needs of prospective consumer communities to know how much provenance detail is appropriate.

Best Practice 5: Provide data provenance information

Provide complete information about the origins of the data and any changes you have made.

Why

Provenance is one means by which consumers of a dataset judge its quality. Understanding its origin and history helps one determine whether to trust the data and provides important interpretive context.

Intended Outcome

Humans will know the origin and history of the dataset and software agents will be able to automatically process provenance information.

Possible Approach to Implementation

The machine-readable version of the data provenance can be provided using an ontology recommended to describe provenance information, such as W3C's Provenance Ontology [PROV-O].

EXAMPLE 5

Machine-readable

The example below shows the machine-readable metadata for the bus stops dataset with the inclusion of the **provenance** metadata. The properties `dct:creator`, `dct:publisher` and `dct:issued` are used to give information about the origin of the dataset. The property `prov:actedOnBehalfOf` is used to designate that Adrian acted on behalf of the Transport Agency of MyCity.

```
:stops-2015-05-05
    a dcat:Dataset, prov:Entity ;
    dct:title "Bus stops of MyCity" ;
    dcat:keyword "transport", "mobility", "bus" ;
    dct:issued "2015-05-05"^^xsd:date ;
    dcat:contactPoint <http://data.mycity.example.com/transport
    dct:temporal <http://reference.data.gov.uk/id/year/2015> ;
    dct:spatial <http://sws.geonames.org/3399415> ;
    dct:publisher :transport-agency-mycity ;
    dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/20
    dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
    dct:creator :adrian
    .

:adrian
    a foaf:Person, prov:Agent ;
    foaf:givenName "Adrian" ;
    foaf:mbox <mailto:adrian@mycitytransport.org> ;
    prov:actedOnBehalfOf :transport-agency-mycity
    .
:transport-agency-mycity
    a foaf:Organization, prov:Agent ;
    foaf:name "Transport Agency of Mycity"
    .
```

Human-readable

Example page with human-readable provenance information about the bus stops dataset is available.

How to Test

Check that the metadata for the dataset itself includes the provenance information about the dataset in a human-readable format.

Check if a computer application can automatically process the provenance information about the dataset.

Evidence

Relevant requirements: R-ProvAvailable, R-MetadataAvailable

Benefits

8.5 Data Quality

The quality of a dataset can have a big impact on the quality of applications that use it. As a consequence, the inclusion of data quality information in data publishing and consumption pipelines is of primary importance. Usually, the assessment of quality involves different kinds of quality dimensions, each representing groups of characteristics that are relevant to publishers and consumers. The Data Quality Vocabulary defines concepts such as measures and metrics to assess the quality for each quality dimension [VOCAB-DQV]. There are heuristics designed to fit specific assessment situations that rely on quality indicators, namely, pieces of data content, pieces of data meta-information, and human ratings that give indications about the suitability of data for some intended use.

Best Practice 6: Provide data quality information

Provide information about data quality and fitness for particular purposes.

Why

Data quality might seriously affect the suitability of data for specific applications, including applications very different from the purpose for which it was originally generated. Documenting data quality significantly eases the process of dataset selection, increasing the chances of reuse. Independently from domain-specific peculiarities, the quality of data should be documented and known quality issues should be explicitly stated in metadata.

Intended Outcome

Humans and software agents will be able to assess the quality and therefore suitability of a dataset for their application.

Possible Approach to Implementation

The machine-readable version of the dataset quality metadata may be provided using the Data Quality Vocabulary developed by the DWBP working group [VOCAB-DQV].

EXAMPLE 6

Machine-readable

The example below shows the metadata for the CSV distribution of the bus stops dataset with the inclusion of the data quality metadata. The metadata was defined according to the Data Quality Vocabulary. Further examples can be found in the Data Quality Vocabulary document [VOCAB-DQV].

```
  :stops-2015-05-05.csv
      a dcat:Distribution ;
      dcat:downloadURL <http://data.mycity.example.com/transport/
      dct:title "CSV distribution of stops-2015-05-05 dataset" ;
      dct:description "CSV distribution of the bus stops dataset
      dcat:mediaType "text/csv;charset=UTF-8" ;
      dct:license <http://creativecommons.org/licenses/by-sa/3.0/
      dqv:hasQualityMeasurement :measure1, :measure2
      .
  :measure1
      a dqv:QualityMeasurement ;
      dqv:computedOn :stops-2015-05-05.csv ;
      dqv:isMeasurementOf :downloadURLAvailabilityMetric ;
      dqv:value "true"^^xsd:boolean
      .
  :measure2
      a dqv:QualityMeasurement ;
      dqv:computedOn :stops-2015-05-05.csv ;
      dqv:isMeasurementOf :csvCompletenessMetric ;
      dqv:value "0.5"^^xsd:double
      .

#definition of dimensions and metrics
  :availability
      a dqv:Dimension ;
      skos:prefLabel "Availability"@en ;
      skos:definition "Availability of a dataset is the extent to
      dqv:inCategory :accessibility
      .
  :completeness
      a dqv:Dimension ;
      skos:prefLabel "Completeness"@en ;
      skos:definition "Completeness refers to the degree to which
      dqv:inCategory :intrinsicDimensions
      .
  :downloadURLAvailabilityMetric
      a dqv:Metric ;
      skos:definition "It checks if dcat:downloadURL is available
      dqv:inDimension :availability
      .
  :csvCompletenessMetric
      a dqv:Metric ;
      skos:definition "Ratio between the number of objects repres
      dqv:inDimension :completeness
      .
```

Human-readable

> Example page with human-readable data quality information.

How to Test

Check that the metadata for the dataset itself includes quality information about the dataset.

Check if a computer application can automatically process the quality information about the dataset.

Evidence

Relevant Requirements: R-QualityMetrics, R-DataMissingIncomplete, R-QualityOpinions

Benefits

8.6 Data Versioning

Datasets published on the Web may change over time. Some datasets are updated on a scheduled basis, and other datasets are changed as improvements in collecting the data make updates worthwhile. In order to deal with these changes, new versions of a dataset may be created. Unfortunately, there is no consensus about when changes to a dataset should cause it to be considered a different dataset altogether rather than a new version. In the following, we present some scenarios where most publishers would agree that the revision should be considered a new version of the existing dataset.

- Scenario 1: a new bus stop is created and it should be added to the dataset;
- Scenario 2: an existing bus stop is removed and it should be deleted from the dataset;
- Scenario 3: an error was identified in one of the existing bus stops stored in the dataset and this error must be corrected.

In general, multiple datasets that represent time series or spatial series, e.g. the same kind of data for different regions or for different years, are not considered multiple versions of the same dataset. In this case, each dataset covers a different set of observations about the world and should be treated as a new dataset. This is also the case with a dataset that collects data about weekly weather forecasts for a given city, where every week a new dataset is created to store data about that specific week.

Scenarios 1 and 2 might trigger a major version, whereas Scenario 3 would likely trigger only a minor version. But how you decide whether versions are minor or major is less important than that you avoid making changes without incrementing the version indicator. Even for small changes, it is important to keep track of the different dataset versions to make the dataset trustworthy. Publishers should

remember that a given dataset may be in use by one or more data consumers, and they should take reasonable steps to inform those consumers when a new version is released. For real-time data, an automated timestamp can serve as a version identifier. For each dataset, the publisher should take a consistent, informative approach to versioning, so data consumers can understand and work with the changing data.

Best Practice 7: Provide a version indicator

Assign and indicate a version number or date for each dataset.

Why

Version information makes a revision of a dataset uniquely identifiable. Uniqueness can be used by data consumers to determine whether and how data has changed over time and to determine specifically which version of a dataset they are working with. Good data versioning enables consumers to understand if a newer version of a dataset is available. Explicit versioning allows for repeatability in research, enables comparisons, and prevents confusion. Using unique version numbers that follow a standardized approach can also set consumer expectations about how the versions differ.

Intended Outcome

Humans and software agents will easily be able to determine which version of a dataset they are working with.

Possible Approach to Implementation

The best method for providing versioning information will vary according to the context; however, there are some basic guidelines that can be followed, for example:

- Include a unique version number or date as part of the metadata for the dataset.
- Use a consistent numbering scheme with a meaningful approach to incrementing digits, such as [SchemaVer].
- If the data is made available through an API, the URI used to request the latest version of the data should not change as the versions change, but it should be possible to request a specific version through the API.
- Use Memento [RFC7089], or components thereof, to express temporal versioning of a dataset and to access the version that was operational at a given datetime. The Memento protocol aligns closely with the approach for assigning URIs to versions that is used for W3C specifications, described below.

The Web Ontology Language [OWL2-QUICK-REFERENCE] and the Provenance, Authoring and versioning Ontology [PAV] provide a number of annotation properties for version information.

EXAMPLE 7

Machine-readable

The example below shows the metadata for bus stops with the inclusion of the versioning metadata. The properties `owl:versionInfo` and `pav:version` are used to denote the version of the dataset.

```
:stops-2015-05-05
    a dcat:Dataset ;
    dct:title "Bus stops of MyCity" ;
    dcat:keyword "transport","mobility","bus" ;
    dct:issued "2015-05-05"^^xsd:date ;
    dcat:contactPoint <http://data.mycity.example.com/transport
    dct:temporal <http://reference.data.gov.uk/id/year/2015> ;
    dct:spatial <http://sws.geonames.org/3399415> ;
    dct:publisher :transport-agency-mycity ;
    dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/20
    dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
    dct:creator :adrian ;
    owl:versionInfo "1.0" ;
    pav:version "1.0"
    .
```

Using Memento

Assume:

- `http://data.mycity.example.com/transport/dataset/bus/stops` is the “generic URI” at which the current version of a dataset is always available
- `http://data.mycity.example.com/transport/dataset/bus/stops-2015-12-17` is the versioned URI for the current dataset
- `http://data.mycity.example.com/transport/dataset/bus/stops-2015-05-05` is the versioned URI of the prior version of the dataset

In the Memento protocol, the versioned URIs provide HTTP response header information to express their version datetime and their relation to the generic URI:

```
curl -I http://data.mycity.example.com/transport/dataset/bus/stop

HTTP/1.1 200 OK
Memento-Datetime: Thu, 17 Dec 2015 00:00:00 GMT
Link:<http://data.mycity.example.com/transport/dataset/bus/stops>
rel=“original”
```

The generic URI provides a link to a TimeGate, which supports datetime negotiation as a means to determine which version of a dataset was operational at a given datetime. Since the generic URI is not versioned, no version datetime is provided in the headers.

```
curl -i -H http://data.mycity.example.com/transport/dataset/bus/s

HTTP/1.1 200 OK
Link: <http://data.mycity.example.com/transport/dataset/bus/timeg
rel="timegate"
```

The versioned URIs can also provide a link to a TimeGate:

```
curl -I http://data.mycity.example.com/transport/dataset/bus/stop

HTTP/1.1 200 OK
Memento-Datetime: Tue, 05 May 2015 00:00:00 GMT
Link: <http://data.mycity.example.com/transport/dataset/bus/stops
rel="original",
 <http://data.mycity.example.com/transport/dataset/bus/timegate/s
rel="timegate"
```

This is how a client determines which dataset version was operational on June 20 2015:

```
curl -I -H "Accept-Datetime: Sat, 20 Jun 2015  12:00:00 GMT" http

HTTP/1.1 302 Found
Vary: accept-datetime
Location: http://data.mycity.example.com/transport/dataset/bus/st
Link: <http://data.mycity.example.com/transport/dataset/bus/stops
rel="original"
```

Human-readable

Example page with human-readable data versioning information.

How to Test

Check if the metadata for the dataset/distribution provides a unique version number or date in a human-readable format.

Check if a computer application can automatically detect/discover and process the unique version number or date of a dataset or distribution.

Evidence

Relevant requirements: R-DataVersion

Benefits

Best Practice 8: Provide version history

Provide a complete version history that explains the changes made in each version.

Why

In creating applications that use data, it can be helpful to understand the variability of that data over time. Interpreting the data is also enhanced by an understanding of its dynamics. Determining how the various versions of a dataset differ from each other is typically very laborious unless a summary of the differences is provided.

Intended Outcome

Humans and software agents will be able to understand how the dataset typically changes from version to version and how any two specific versions differ.

Possible Approach to Implementation

Provide a list of published versions and a description for each version that explains how it differs from the previous version. An API can expose a version history with a single dedicated URL that retrieves the latest version of the complete history.

EXAMPLE 8

Machine-readable

Suppose that a new bus stop was created and a new dataset (`stops-2015-12-17`) is published to keep the data up to date. The new dataset is a version of `stops-2015-05-05`. The machine-readable metadata of the new dataset is shown below with the corresponding versioning history information.

```
:stops-2015-12-17
    a dcat:Dataset ;
    dct:title "Bus stops of MyCity" ;
    dcat:keyword "transport","mobility","bus" ;
    dct:issued "2015-12-17"^^xsd:date ;
    dcat:contactPoint <http://data.mycity.example.com/transport
    dct:temporal <http://reference.data.gov.uk/id/year/2015> ;
    dct:spatial <http://sws.geonames.org/3399415> ;
    dct:publisher :transport-agency-mycity ;
    dct:accrualPperiodicity <http://purl.org/linked-data/sdmx/2
    dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
    dct:creator :adrian ;
     ...
    dct:isVersionOf :stops-2015-05-05 ;
    pav:previousVersion :stops-2015-05-05 ;
    rdfs:comment "The bus stops dataset was updated to reflect
    owl:versionInfo "1.1" ;
    pav:version "1.1"
    .
```

Using Memento:

Assume:

- `http://data.mycity.example.com/transport/dataset/bus/stops` is the "generic URI" at which the current version of a dataset is always available
- `http://data.mycity.example.com/transport/dataset/bus/stops-2015-12-17` is the versioned URI for the current dataset
- `http://data.mycity.example.com/transport/dataset/bus/stops-2015-05-05` is the versioned URI of the prior version of the dataset
- http://example.org/stops-2015-01-01 is the versioned URI of the first version of the dataset

The versioned URIs, the generic URI, and the TimeGate can provide a link to a TimeMap that provides an overview of all temporal versions of the dataset:

```
curl -I http://data.mycity.example.com/transport/dataset/bus/stop

HTTP/1.1 200 OK
Memento-Datetime: Tue, 05 May 2015 00:00:00 GMT
Link: <http://data.mycity.example.com/transport/dataset/bus/stops
rel="original",
 <http://data.mycity.example.com/transport/dataset/bus/timemap/st
rel="timemap";
 type="application/link-format"
```

This is how the TimeMap is retrieved:

```
curl -I http://data.mycity.example.com/transport/dataset/bus/time

HTTP/1.1 200 OK
Content-Type: application/link-format

<http://data.mycity.example.com/transport/dataset/bus/stops>;rel=
<http://data.mycity.example.com/transport/dataset/bus/timegate/st
<http://data.mycity.example.com/transport/dataset/bus/timemap/stc
 type="application/link-format",
<http://data.mycity.example.com/transport/dataset/bus/stops-2015-
rel="first memento"; datetime="Thu, 01 Jan 2015 00:00:00 GMT",
<http://data.mycity.example.com/transport/dataset/bus/stops-2015-
rel="memento"; datetime="Tue, 05 May 2015 00:00:00 GMT"
<http://data.mycity.example.com/transport/dataset/bus/stops-2015-
rel="last memento"; datetime="Thu, 17 Dec 2015 00:00:00 GMT"
```

The versioned URI can provide information regarding relations with other dataset versions:

```
curl -I http://data.mycity.example.com/transport/dataset/bus/stop

HTTP/1.1 200 OK
Memento-Datetime: Tue, 05 May 2015 00:00:00 GMT
Link: <http://data.mycity.example.com/transport/dataset/bus/stops
rel="original",
 <http://data.mycity.example.com/transport/dataset/bus/stops-2015
rel="prev first memento";
 datetime="Thu, 01 Jan 2015 00:00:00 GMT",
 <http://data.mycity.example.com/transport/dataset/bus/stops-2015
rel="next last memento";
 datetime="Thu, 17 Dec 2015 00:00:00 GMT"
```

Human-readable

Example page with human-readable data versioning history information.

How to Test

Check that a list of published versions is available as well as a change log describing precisely how each version differs from the previous one.

Evidence

Relevant requirements: R-DataVersion

Benefits

8.7 Data Identifiers

Identifiers take many forms and are used extensively in every information system. Data discovery, usage and citation on the Web depends fundamentally on the use of HTTP (or HTTPS) URIs: globally unique identifiers that can be looked up by dereferencing them over the Internet [RFC3986]. It is perhaps worth emphasizing some key points about URIs in the current context.

1. URIs are 'dumb strings', that is, they carry no semantics. Their function is purely to identify a resource.
2. Although the previous point is accurate, it would be perverse for a URI such as http://example.com/dataset.csv to return anything other than a CSV file. Human readability is helpful.
3. When de-referenced (looked up), a single URI may offer the same resource in more than one format. http://example.com/dataset may offer the same data in, say, CSV, JSON and XML. The server returns the most appropriate format based on content negotiation .
4. One URI may redirect to another.
5. De-referencing a URI triggers a computer program to run on a server that may do something as simple as return a single, static file, or it may carry out complex processing. Precisely what processing is carried out, i.e. the software on the server, is completely independent of the URI itself.

Best Practice 9: Use persistent URIs as identifiers of datasets

Identify each dataset by a carefully chosen, persistent URI.

Why

Adopting a common identification system enables basic data identification and comparison processes by any stakeholder in a reliable way. They are an essential pre-condition for proper data management and reuse.

Developers may build URIs into their code and so it is important that those URIs persist and that they dereference to the same resource over time without the need for human intervention.

Intended Outcome

Datasets or information about datasets will be discoverable and citable through time, regardless of the status, availability or format of the data.

Possible Approach to Implementation

To be persistent, URIs must be designed as such. A lot has been written on this topic, see, for example, the European Commission's Study on Persistent URIs [PURI] which in turn links to many other resources.

Where a data publisher is unable or unwilling to manage a URI space directly for persistence, an alternative approach is to use a redirection service such as Permanent Identifiers for the Web or purl.org. These provide persistent URIs that can be redirected as required so that the eventual location can be ephemeral. The software behind such services is freely available so that it can be installed and managed locally if required.

Digital Object Identifiers (DOIs) offer a similar alternative. These identifiers are defined independently of any Web technology but can be appended to a 'URI stub.' DOIs are an important part of the digital infrastructure for research data and libraries.

EXAMPLE 9

The URI `http://data.mycity.example.com/transport/dataset/bus/stops` has several features that support persistence:

- All names are subject to change over time but in choosing a domain name, it is reasonable for Adrian to assume that MyCity will continue to exist and that it will continue to have a government. Therefore, while cases like Yugoslavia prove that even country names change and top level domains disappear (like .yu), a domain name based on the city's name is as persistent as any domain name can be.
- By putting data on the `data.mycity.example.com` subdomain, Adrian is creating a specific domain that can be managed independently of any particular department.
- It is *not* safe to assume that a specific *department* will persist. The authorities in MyCity might very well decide that the Transport Agency should be merged with another to create the Transport and Environment Agency. It is right, therefore, not to include the name of the Transport Agency in the URI, but to include the task from which the data comes, in this case that of providing public transport.
- The `/dataset` path segment is an indication that the URI identifies a dataset, rather than, say, a specific bus route.
- Likewise, the path segment of `/bus` take us further towards the specific dataset for which Adrian is responsible.
- Finally, `/stops` leads us to the dataset concerning bus stops in MyCity.
- In DCAT terms, this would be the identifier for the dataset. Specific distributions of the dataset are likely to be identified by adding the relevant file extension to the URI, such as `http://data.mycity.example.com/transport/dataset/bus/stops.csv`, `http://data.mycity.example.com/transport/dataset/bus/stops.json`, `http://data.mycity.example.com/transport/dataset/bus/stops.ttl` etc.

How to Test

Check that each dataset is identified using a URI that has been designed for persistence. Ideally the relevant Web site includes a description of the design scheme and a credible pledge of persistence should the publisher no longer be able to maintain the URI space themselves.

Evidence

Relevant requirements: R-UniqueIdentifier, R-Citable

Benefits

Best Practice 10: Use persistent URIs as identifiers within datasets

Reuse other people's URIs as identifiers within datasets where possible.

Why

The power of the Web lies in the *Network effect*. The first telephone only became useful when the second telephone meant there was someone to call; the third telephone made both of them more useful yet. Data becomes more valuable if it refers to other people's data about the same thing, the same place, the same concept, the same event, the same person, and so on. That means using the same identifiers across datasets and making sure that your identifiers can be referred to by other datasets. When those identifiers are HTTP URIs, they can be looked up and more data discovered.

These ideas are at the heart of the 5 Stars of Linked Data where one data point links to another, and of Hypermedia where links may be to further data or to services that can act on or relate to the data in some way.

That's the Web of Data.

Intended Outcome

Data items will be related across the Web creating a global information space accessible to humans and machines alike.

Possible Approach to Implementation

This is a topic in itself and a general document such as this can only include superficial detail.

Developers know that very often the problem they are trying to solve will have already been solved by other people. In the same way, if you are looking for a set of identifiers for obvious things like countries, currencies, subjects, species, proteins, cities and regions, Nobel prize winners and products – someone's done it already. The steps described for discovering existing vocabularies [LD-BP] can readily be adapted.

- ensure URI sets you use are published by a trusted group or organization;

- ensure URI sets have persistent URIs.

If you can not find an existing set of identifiers that meet your needs then you will need to create your own, following the patterns for URI persistence so that others will add value to your data by linking to it.

> EXAMPLE 10
>
> The URI given as an example in the previous Best Practice (`http://data.mycity.example.com/transport/dataset/bus/stops`) identifies a dataset. Much of the URI can be reused to identify bus stops, routes and the type of bus used on a given service. For example, a suitable persistent URI for the 'Airport - Bullfrog' route would be:
>
> `http://data.mycity.example.com/transport/route/bus/id/AB`
>
> This has the same initial structure as for the dataset but rather than `/dataset` it now includes the path segment `/route` so that humans can see that the type of thing identified is a bus route. The `/id` segment indicates that the URI identifies something that is not an information resource, i.e. something you cannot retrieve over the Internet, and `/AB` is the local identifier for the actual bus route. This is consistent with advice from GS1's SmartSearch Implementation Guideline [GS1] which says that where standard identifiers are used for a product, location etc., it is recommended that the URI includes the type of identifier being used. For example, if a GTIN is being used to identify a product then the URI should be of the form: `http://data.myproduct.example.com/gtin/05011476100885`. Dereferencing URIs for non-information resources should result in an HTTP 303 redirect to a similar URL such as `http://data.mycity.example.com/transport/route/bus/doc/AB` that *describes*, i.e. gives information about, the AB bus route (note the substitution of `/doc` for `/id`). Jeni Tennison's work on URLs in Data has more to say on this topic [URLs-in-data].
>
> In offering this advice, it is recognized that URIs can be long. In a dataset of even moderate size, storing each URI is likely to be repetitive and obviously wasteful. Instead, define locally unique identifiers for each element (such as `AB` in this example) and provide data that allows them to be converted to globally unique URIs programmatically. The Metadata Vocabulary for Tabular Data [Tabular-Metadata] provides mechanisms for doing this within tabular data such as CSV files, in particular using URI template properties such as the about URL property.

How to Test

Check that within the dataset, references to things that do not change or that change slowly, such as countries, regions, organizations and people, are referred to by URIs or by short identifiers that can be appended to a URI stub. Ideally the URIs should resolve, however, they have value as globally scoped variables whether they resolve or not.

Evidence

Relevant requirements: R-UniqueIdentifier

Benefits

Best Practice 11: Assign URIs to dataset versions and series

Assign URIs to individual versions of datasets as well as to the overall series.

Why

Like documents, many datasets fall into natural series or groups. For example:

- bus stops in MyCity (that change over time);
- a list of elected officials in MyCity
- evolving versions of a document through to completion.

In different circumstances, it will be appropriate to refer to the current situation (the current set of bus stops, the current elected officials etc.). In others, it may be appropriate to refer to the situation as it existed at a specific time.

Intended Outcome

Humans and software agents will be able to refer to specific versions of a dataset and to concepts such as a 'dataset series' and 'the latest version'.

Possible Approach to Implementation

The W3C provides a good example of how to do this. The (persistent) URI for this document is `https://www.w3.org/TR/2016/PR-dwbp-20161215/`. That identifier points to an immutable snapshot of the document on the day of its publication. The URI for the 'latest version' of this document is `https://www.w3.org/TR/dwbp/` which is an identifier for a series of closely related documents that are subject to change over time. At the time of publication, these two URIs both resolve to this document. However, when the next ver-

sion of this document is published, the 'latest version' URI will be changed to point to that, but the dated URI remains unchanged.

> EXAMPLE 11
>
> Suppose that a new bus stop is created. To keep `stops-2015-05-05` up to date, a new version of the dataset (`stops-2015-12-17`) is created. `stops-2015-12-17` includes all the data from `stops-2015-05-05` plus the data about the new bus stop. The two versions can be identified by the following URIs:
>
> `http://data.mycity.example.com/transport/dataset/bus/stops-2015-05-05` is the versioned URI of the first version of the dataset
>
> `http://data.mycity.example.com/transport/dataset/bus/stops-2015-12-17` is the version URI of the updated version of the dataset
>
> `http://data.mycity.example.com/transport/dataset/bus/stops` always resolves to the latest version so it resolved to `stops-2015-05-05` *until* 17 December 2015 when the server configuration was updated to point that URL to `stops-2015-12-17`.

How to Test

Check that each version of a dataset has its own URI, and that there is also a "latest version" URI.

Evidence

Relevant requirements: R-UniqueIdentifier, R-Citable

Benefits

8.8 Data Formats

The format in which data is made available to consumers is a key aspect of making that data usable. The best, most flexible access mechanism in the world is pointless unless it serves data in formats that enable use and reuse. Below we detail Best Practices in selecting formats for your data, both at the level of files and that of individual fields. W3C encourages the use of formats that can be used by the widest possible audience and processed most readily by computing systems. Source formats, such as

database dumps or spreadsheets, used to generate the final published format, are out of scope. This document is concerned with what is actually published rather than internal systems used to generate the published data.

Best Practice 12: Use machine-readable standardized data formats

Make data available in a machine-readable, standardized data format that is well suited to its intended or potential use.

Why

As data becomes more ubiquitous, and datasets become larger and more complex, processing by computers becomes ever more crucial. Posting data in a format that is not machine-readable places severe limitations on the continuing usefulness of the data. Data becomes useful when it has been processed and transformed into information. Note that there is an important distinction between formats that can be read and edited by humans using a computer and formats that are machine-readable. The latter term implies that the data is readily extracted, transformed and processed by a computer.

Using non-standard data formats is costly and inefficient, and the data may lose meaning as it is transformed. By contrast, standardized data formats enable interoperability as well as future uses, such as remixing or visualization, many of which cannot be anticipated when the data is first published. It is also important to note that most machine-readable standardized formats are also locale-neutral.

Intended Outcome

Machines will easily be able to read and process data published on the Web and humans will be able to use computational tools typically available in the relevant domain to work with the data.

Possible Approach to Implementation

Make data available in a machine-readable standardized data format that is easily parseable including but not limited to CSV, XML, HDF5, JSON and RDF serialization syntaxes like RDF/XML, JSON-LD, or Turtle.

EXAMPLE 12

Adrian knows that tabular data is commonly used on the Web and he decides to use CSV as the data format for one of the distributions of the bus stops dataset. To facilitate data processing, he uses the Model for Tabular Data and Metadata on the Web for publishing the CSV distribution (`stops-2015-05-05.csv`). The example below presents a fragment of the CSV distribution which complies with the structural metadata defined in Example 4.

```
Identifier,Name,Description,Latitude,Longitude,ZONE,URL
345,Castle Avenue,Sunset Drive,-3.731862,-38.526670,x20,http://da
483,Main Street,Lily Park,-3.731541,-38.535157,x20,http://data.my
```

How to Test

Check if the data format conforms to a known machine-readable data format specification.

Evidence

Relevant requirements: R-FormatMachineRead, R-FormatStandardized R-FormatOpen

Benefits

Best Practice 13: Use locale-neutral data representations

Use locale-neutral data structures and values, or, where that is not possible, provide metadata about the locale used by data values.

Why

Data values that are machine-readable and not specific to any particular language or culture are more durable and less open to misinterpretation than values that use one of the many different cultural representations. Things like dates, currencies and numbers may look similar but have different meanings in different locales. For example, the 'date' 4/7 can be read as 7th of April or the 4th of July depending on where the data was created. Similarly, €2,000 is either two thousand Euros or an over-precise representation of two Euros. By using a locale-neutral format, systems avoid the need to establish specific interchange rules that vary according to the language or location of the user. When the data is already in a locale-specific format, making the locale and language explicit by providing locale parame-

ters allows users to determine how readily they can work with the data and may enable automated translation services.

Intended Outcome

Humans and software agents will be able to interpret the meaning of strings representing dates, times, currencies and numbers etc. accurately.

Possible Approach to Implementation

Most common data serialization formats are locale-neutral. For example, XML Schema types such as `xsd:integer` and `xsd:date` are intended for locale-neutral data interchange. Using locale-neutral representations allows the data values to be processed accurately without complex parsing or misinterpretation and also allows the data to be presented in the format most comfortable for the consumer of the data in any locale. For example, rather than storing "€2000,00" as a string, it is strongly preferred to exchange a data structure such as:

```
…
"price" {
    "value": 2000.00,
    "currency": "EUR"
}
…
```

Some datasets contain values that are not or cannot be rendered into a locale-neutral format. This is particularly true of any natural language text values. For each data field that can contain locale-affected or natural-language text, there should be an associated language tag used to indicate the language and locale of the data. This locale information can be used in parsing the data or to ensure proper presentation and processing of the value by the consumer. BCP47 [BCP47] provides the standard for language and locale identification and, informatively, CLDR [CLDR] is the source for both representing specific localized formats and as a reference for specific locale data values.

EXAMPLE 13

Machine-readable

The example below shows the machine-readable metadata for the bus stops dataset (`stops-2015-05-05`) with the inclusion of the **locale parameters** metadata. The property `dct:language` is used to declare the languages the dataset is published in. If the dataset is available in multiple languages, use multiple values for this property. The property `dct:conformsTo` is used to specify the standard adopted for date and time formats.

```
:stops-2015-05-05
    a dcat:Dataset ;
    dct:title "Bus stops of MyCity" ;
    dcat:keyword "transport","mobility","bus" ;
    dct:issued "2015-05-05"^^xsd:date ;
    dcat:contactPoint <http://data.mycity.example.com/transport
    dct:temporal <http://reference.data.gov.uk/id/year/2015> ;
    dct:spatial <http://sws.geonames.org/3399415> ;
    dct:publisher :transport-agency-mycity ;
    dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/20
    dcat:theme :mobility ;
    dcat:distribution :stops-2015-05-05.csv ;
    dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ,
                 <http://id.loc.gov/vocabulary/iso639-1/pt> ;
    dct:conformsTo <http://www.iso.org/iso/home/standards/iso8
    .
```

The example below shows a **locale-neutral representation** of bus fare data.

```
fare_id,price,currency_type,payment_method,transfers,transfer_d
p,1.25,USD,0,0,0
a,5.25,USD,0,0,0
```

Human-readable

Example page with human-readable description of dataset is available.

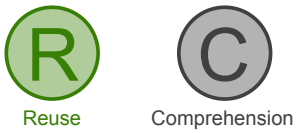
How to Test

Check that locale-sensitive data values are represented in a locale-neutral format or that, if this is not possible, relevant locale metadata is provided.

Evidence

Relevant requirements: R-FormatLocalize, R-MetadataAvailable, R-GeographicalContext, R-FormatMachineRead

Benefits



Best Practice 14: Provide data in multiple formats

Make data available in multiple formats when more than one format suits its intended or potential use.

Why

Providing data in more than one format reduces costs incurred in data transformation. It also minimizes the possibility of introducing errors in the process of transformation. If many users need to transform the data into a specific data format, publishing the data in that format from the beginning saves time and money and prevents errors many times over. Lastly it increases the number of tools and applications that can process the data.

Intended Outcome

As many users as possible will be able to use the data without first having to transform it into their preferred format.

Possible Approach to Implementation

Consider the data formats most likely to be needed and consider alternatives that are likely to be useful in the future. Data publishers must balance the effort required to make the data available in many formats against the cost of doing so, but providing at least one alternative will greatly increase the usability of the data. In order to serve data in more than one format you can use content negotiation as described in Best Practice Use content negotiation for serving data available in multiple formats.

A word of warning: local identifiers within the dataset, which may be exposed as fragment identifiers in URIs, must be consistent across the various formats.

EXAMPLE 14

In order to reach a larger number of data consumers, Adrian decides to also provide a JSON distribution of the bus stops dataset. In the following example, the property `dcat:distribution` is used to associate the dataset `stops-2015-05-05` with its two distributions: `stops-2015-05-05.csv` and `stops-2015-05-05.json`.

```
:stops-2015-05-05
    a dcat:Dataset ;
    dcat:distribution :stops-2015-05-05.csv ;
    dcat:distribution :stops-2015-05-05.json
    .
:stops-2015-05-05.csv
    a dcat:Distribution ;
    dcat:downloadURL <http://data.mycity.example.com/transport/
    dct:title "CSV distribution of stops-2015-05-05 dataset" ;
    dct:description "CSV distribution of the bus stops dataset
    dcat:mediaType "text/csv;charset=UTF-8" ;
    dct:license <http://creativecommons.org/licenses/by-sa/3.0/
    .
:stops-2015-05-05.json
    a dcat:Distribution ;
    dcat:downloadURL <http://data.mycity.example.com/transport/
    dct:title "JSON distribution of stops-2015-05-05 dataset" ;
    dct:description "JSON distribution of the bus stops dataset
    dcat:mediaType "application/json" ;
    dct:license <http://creativecommons.org/licenses/by-sa/3.0/
    .
```

How to Test

Check if the complete dataset is available in more than one data format.

Evidence

Relevant requirements: R-FormatMultiple

Benefits

8.9 Data Vocabularies

Vocabularies define the concepts and relationships (also referred to as “terms” or “attributes”) used to describe and represent an area of interest. They are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. Several near-synonyms for 'vocabulary' have been coined, for example, ontology, controlled vocabulary, thesaurus, taxonomy, code list, semantic network.

There is no strict division between the artifacts referred to by these names. “Ontology” tends however to denote the vocabularies of classes and properties that structure the descriptions of resources in (linked) datasets. In relational databases, these correspond to the names of tables and columns; in XML, they correspond to the elements defined by an XML Schema. Ontologies are the key building blocks for inference techniques on the Semantic Web. The first means offered by W3C for creating ontologies is the RDF Schema [RDF-SCHEMA] language. It is possible to define more expressive ontologies with additional axioms using languages such as those in The Web Ontology Language [OWL2-OVERVIEW].

On the other hand, “controlled vocabularies”, “concept schemes” and “knowledge organization systems” enumerate and define resources that can be employed in the descriptions made with the former kind of vocabulary, i.e. vocabularies that structure the descriptions of resources in (linked) datasets. A concept from a thesaurus, say, “architecture”, will for example be used in the subject field for a book description (where “subject” has been defined in an ontology for books). For defining the terms in these vocabularies, complex formalisms are most often not needed. Simpler models have thus been proposed to represent and exchange them, such as the ISO 25964 data model [ISO-25964] or W3C's Simple Knowledge Organization System [SKOS-PRIMER].

Best Practice 15: Reuse vocabularies, preferably standardized ones

Use terms from shared vocabularies, preferably standardized ones, to encode data and metadata.

Why

Use of vocabularies already in use by others captures and facilitates consensus in communities. It increases interoperability and reduces redundancies, thereby encouraging reuse of your own data. In particular, the use of shared vocabularies for metadata (especially structural, provenance, quality and versioning metadata) helps the comparison and automatic processing of both data and metadata. In addition, referring to codes and terms from standards helps to avoid ambiguity and clashes between similar elements or values.

Intended Outcome

Interoperability and consensus among data publishers and consumers will be enhanced.

Possible Approach to Implementation

The Vocabularies section of the W3C Best Practices for Publishing Linked Data [LD-BP] provides guidance on the discovery, evaluation and selection of existing vocabularies.

Organizations such as the Open Geospatial Consortium (OGC), ISO, W3C, WMO, libraries and research data services, etc. provide lists of codes, terminologies and Linked Data vocabularies that can be used by everyone. A key point is to make sure the dataset, or its documentation, provides enough (human- and machine-readable) context so that data consumers can retrieve and exploit the standardized meaning of the values. In the context of the Web, using unambiguous, Web-based identifiers (URIs) for standardized vocabulary resources is an efficient way to do this, noting that the same URI may have multilingual labels attached for greater cross-border interoperability. The European Union's multilingual thesaurus, Eurovoc, provides a prime example.

EXAMPLE 15

1. The DCAT vocabulary expresses metadata concerning datasets [VOCAB-DCAT] and re-uses elements from several pre-existing vocabularies: Dublin Core, FOAF, SKOS and vCard. Reusing Dublin Core properties like `dct:title` instead of creating new ones (say, `dcat:title`) enables DCAT-based metadata to be consumed by any application that can read and manipulate Dublin Core statements.

2. In the digital culture sector, the data model for Europeana (EDM) also makes extensive uses of existing shared vocabularies like Dublin Core, FOAF, SKOS, etc. This has facilitated adoption of EDM by Europeana's data providers and helped position it as a Best Practice for similar initiatives in the same sector. For instance, the metadata application profile from the Digital Public Library of America reuses EDM and thus the various vocabularies that EDM builds on. As a result, large amounts of digital culture data have become more interoperable within the sector. That data is also easier to reuse by consumers from other communities, who are not familiar with the traditional models and terminologies used by library, archives and museums.

3. The Library of Congress publishes lists of ISO 639 languages as Linked Data (see [ISO639-1-LOC] for two-letter codes):

```
:stops
    dct:language <http://id.loc.gov/vocabulary/iso639-1/en> .
```

4. Australia's Solid Earth and Environment Grid publishes a reference list of URIs for geologic timescale elements from the International Commission on Stratigraphy's Chronostratigraphic Chart, such as `http://resource.geosciml.org/classifier/ics/ischart/Paleozoic` for the Paleozoic Era:

```
:dataset-005 a dcat:Dataset ;
    dct:temporal <http://resource.geosciml.org/classifier/ics
```

5. Google maintains the General Transit Feed Specification that defines a format for publishing public transportation data. This format relies on a set of fields like `route_short_name` or `route_type` that are carefully defined and exposed to constant community feedback in order to facilitate consensus. Definitions include specifications of coded values, as the ones used with `route_type`:

```
0 - Tram, Streetcar, Light rail. Any light rail or street lev
1 - Subway, Metro. Any underground rail system within a metro
2 - Rail. Used for intercity or long-distance travel.
```

> Note that in a non-Linked Data fashion, these fields and codes have no individual Web identifiers nor machine-readable semantics. Exploiting them thus requires implementers to parse the documentation and encode interpretations in each individual application consuming the data.

How to Test

Using vocabulary repositories like the Linked Open Vocabularies repository or lists of services mentioned in technology-specific Best Practices such as the Best Practices for Publishing Linked Data [LD-BP], or the Core Initial Context for RDFa and JSON-LD, check that classes, properties, terms, elements or attributes used to represent a dataset do not replicate those defined by vocabularies used for other datasets.

Check if the terms or codes in the vocabulary to be used are defined in a standards development organization such as IETF, OGC & W3C etc., or are published by a suitable authority, such as a government agency.

Evidence

Relevant requirements: R-MetadataStandardized, R-MetadataDocum, R-QualityComparable, R-VocabOpen, R-VocabReference

Benefits

Best Practice 16: Choose the right formalization level

Opt for a level of formal semantics that fits both data and the most likely applications.

Why

As Albert Einstein may or may not have said: everything should be made as simple as possible, but not simpler.

Formal semantics help to establish precise specifications that convey detailed meaning and using a complex vocabulary (ontology) may serve as a basis for tasks such as automated reasoning. On the other hand, such complex vocabularies require more effort to produce and understand, which could hamper their reuse, comparison and linking of datasets that use them.

If the data is sufficiently rich to support detailed research questions (the fact that A, B and C are true, and that D is not true, leads to the conclusion E) then something like an OWL Profile would clearly be appropriate [OWL2-PROFILES].

But there is nothing complicated about a list of bus stops.

Choosing a very simple vocabulary is always attractive but there is a danger: the drive for simplicity might lead the publisher to omit some data that provides important information, such as the geographical location of the bus stops that would prevent showing them on a map. Therefore, a balance has to be struck, remembering that the goal is not simply to share your data, but for others to reuse it.

Intended Outcome

The most likely application cases will be supported with no more complexity than necessary.

Possible Approach to Implementation

Look at what your peers do already. It is likely you will see that there is a commonly used vocabulary that matches, or nearly matches, your current needs. That is probably the one to use.

You may find a vocabulary that you'd like to use but you notice a semantic constraint that makes it difficult to do so, such as a domain or range restriction that does not apply to your case. In that scenario, it is often worth contacting the vocabulary publisher and talking to them about it. They may well be able to lift that restriction and provide further guidance on how the vocabulary is used more broadly.

W3C operates a mailing list at public-vocabs@w3.org [archive] where issues around vocabulary usage and development can be discussed.

If you are creating a vocabulary of your own, keep the semantic restrictions to the minimum that works for you, again, so as to increase the possibility of reuse by others. As an example, the designers of the (very widely used) SKOS ontology itself have minimized its ontological commitment by questioning all formal axioms that were suggested for its classes and properties. Often they were rejected because their use, while beneficial to many applications, would have created formal inconsistencies for the data from other applications, making SKOS not usable at all for these. As an example, the property `skos:broader` was not defined as a transitive property, even though it would have fitted the way hierarchical links between concepts are created for many thesauri [SKOS-DESIGN]. Look for evidence of that kind of "design for wide use" when selecting a vocabulary.

Another example of this "design for wide use" can be seen in schema.org. Launched in June 2011, schema.org was massively adopted in a very short time in part because of its informative rather than normative approach for defining the types of objects that properties can be

used with. For instance, the values of the property `author` are only "expected" to be of type `Organization` or `Person`. `author` "can be used" on the type `CreativeWork` but this is not a strict constraint. Again, that approach to design makes schema.org a good choice as a vocabulary to use when encoding data for sharing.

> EXAMPLE 16
>
> Adrian encodes the bus stop data using GTFS [GTFS] because:
>
> - it is in widespread use;
> - it offers a level of detail that matches his data;
> - a motivation for publishing bus stop data is to support the development of applications to help bus users and GTFS is designed for just that purpose.

How to Test

This is almost always a matter of subjective judgment with no objective test. As a general guideline:

- Are common vocabularies used such as Dublin Core and schema.org?
- Are simple facts stated simply and retrieved easily?
- For formal knowledge representation languages, applying an inference engine on top of the data that uses a given vocabulary does not produce too many statements that are unnecessary for target applications.

Evidence

Relevant requirements: R-VocabReference, R-QualityComparable

Benefits

8.10 Data Access

Providing easy access to data on the Web enables both humans and machines to take advantage of the benefits of sharing data using the Web infrastructure. By default, the Web offers access using Hypertext Transfer Protocol (HTTP) methods. This provides access to data at an atomic transaction level. This might be through the simple bulk download of a file or, where data is distributed across multiple

files or requires more sophisticated retrieval methods, through an API. The two basic methods, bulk download and API, are not mutually exclusive.

In the bulk download approach, data is generally pre-processed server side where multiple files or directory trees of files are provided as one downloadable file. When bulk data is being retrieved from non-file system solutions, depending on the data user communities, the data publisher can offer APIs to support a series of retrieval operations representing a single transaction.

For data that is generated in real time or near real time, data publishers should use an automated system to enable immediate access to time-sensitive data, such as emergency information, weather forecasting data, or system monitoring metrics. In general, APIs should be available to allow third parties to automatically search and retrieve such data.

Aside from helping to automate real-time data pipelines, APIs are suitable for all kinds of data on the Web. Though they generally require more work than posting files for download, publishers are increasingly finding that delivering a well documented, standards-based, stable API is worth the effort.

For some data publishers, it is important to know who has downloaded the data and how they have used it. There are two possible approaches to gathering this information. First, publishers can *invite* users to provide it, the user's motivation for doing so being that it encourages the continued publication of the data and promotes their own work. A second and less user-friendly approach is to require registration before data is accessed. In both cases, the Dataset Usage Vocabulary [VOCAB-DUV] provides a structure for representing such information. When collecting data from users, the publisher should explain why and how information gathered from users (either explicitly or implicitly) will be used. Without a clear policy users might be fearful of providing information and thus the value of the dataset is reduced.

Best Practice 17: Provide bulk download

Enable consumers to retrieve the full dataset with a single request.

Why

When Web data is distributed across many URIs but might logically be organized as one container, accessing the data in bulk can be useful. Bulk access provides a consistent means to handle the data as one dataset. Individually accessing data over many retrievals can be cumbersome and, if used to reassemble the complete dataset, can lead to inconsistent approaches to handling the data.

Intended Outcome

Large file transfers that would require more time than a typical user would consider reasonable will be possible via dedicated file-transfer protocols.

Possible Approach to Implementation

Depending on the nature of the data and consumer needs, possible approaches could include the following:

- For datasets that exist initially as multiple files, preprocessing a copy of the data into a single file and making the data accessible for download from one URI. For larger datasets, the file can also be compressed.
- Hosting an API that includes the ability to retrieve a bulk download in addition to dynamic queries. This approach is useful for capturing a complete snapshot of dynamic data.
- For very large datasets, bulk file transfers can be enabled via means other than http, such as bbcp or GridFTP.

The bulk download should include the metadata describing the dataset. Discovery metadata [VOCAB-DCAT] should also be available outside the bulk download.

> EXAMPLE 17
>
> The MyCity transit agency may have a large dataset with arrival times for the various transit modes that was collected over an entire year. The data might be stored as a CSV file for each month. Suppose the agency wants to make that data available as a bulk download containing all the CSV files, for a hackathon. Since all the arrival data for all the transit services would be a lot of data, and they want to provide all the months together as one dataset, they might offer it as a single-file, compressed archive (tarred and gzipped).

How to Test

Check if the full dataset can be retrieved with a single request.

Evidence

Relevant requirements: R-AccessBulk

Benefits

Best Practice 18: Provide Subsets for Large Datasets

If your dataset is large, enable users and applications to readily work with useful

subsets of your data.

Why

Large datasets can be difficult to move from place to place. It can also be inconvenient for users to store or parse a large dataset. Users should not have to download a complete dataset if they only need a subset of it. Moreover, Web applications that tap into large datasets will perform better if their developers can take advantage of "lazy loading", working with smaller pieces of a whole and pulling in new pieces only as needed. The ability to work with subsets of the data also enables offline processing to work more efficiently. Real-time applications benefit in particular, as they can update more quickly.

Intended Outcome

Humans and applications will be able to access subsets of a dataset, rather than the entire thing, with a high ratio of needed to unneeded data for the largest number of users. Static datasets that users in the domain would consider to be too large will be downloadable in smaller pieces. APIs will make slices or filtered subsets of the data available, the granularity depending on the needs of the domain and the demands of performance in a Web application.

Possible Approaches to Implementation

Consider the expected use cases for your dataset and determine what types of subsets are likely to be most useful. An API is usually the most flexible approach to serving subsets of data, as it allows customization of what data is transferred, making the available subsets much more likely to provide the needed data – and little unneeded data – for any given situation. The granularity should be suitable for Web application access speeds. (An API call that returns within one second enables an application to deliver interactivity that feels natural. Data that takes more than ten seconds to deliver will likely cause users to suspect failure.)

Another way to subset a dataset is to simply split it into smaller units and make those units individually available for download or viewing.

It can also be helpful to mark up a dataset so that individual sections through the data (or even smaller pieces, if expected use cases warrant it) can be processed separately. One way to do that is by indicating "slices" with the RDF Data Cube Vocabulary.

EXAMPLE 18

The MyCity transit agency has been collecting detailed data about passenger usage for several years. This is a very large dataset, containing values for numbers of passengers by transit type, route, vehicle, driver, entry stop, exit stop, transit pass type, entry time, etc. They have found that a wide variety of stakeholders are interested in downloading various subsets of the data. The folks who run each transit system want only the data for their transit mode, the city planners only want the numbers of entries and exits at each stop, the city budget office wants only the numbers for the various types of passes sold, and others want still different subsets. The agency created a Web site where users can select which variables are of interest to them, set ranges on some variables, and download only the subset they need.

How to Test

Check that the entire dataset can be recovered by making multiple requests that retrieve smaller units.

Evidence

Relevant requirements: R-Citable, R-GranularityLevels, R-UniqueIdentifier, R-AccessRealTime

Benefits

Best Practice 19: Use content negotiation for serving data available in multiple formats

Use content negotiation in addition to file extensions for serving data available in multiple formats.

Why

It is possible to serve data in an HTML page mixed with human-readable and machine-readable data, using RDFa for example. However, as the Architecture of the Web [WEBARCH] and DCAT [VOCAB-DCAT] make clear, a resource, such as a dataset, can have many representations. The same data might be available as JSON, XML, RDF, CSV and HTML. These multiple representations can be made available via an API, but should be made available from *the same* URL using content negotiation to return the appropriate rep-

resentation (what DCAT calls a distribution). Specific URIs can be used to identify individual representations of the data directly, by-passing content negotiation.

Intended Outcome

Content negotiation will enable different resources or different representations of the same resource to be served according to the request made by the client.

Possible Approach to Implementation

A possible approach to implementation is to configure the Web server to deal with content negotiation of the requested resource.

The specific format of the resource's representation can be accessed by the URI or by the Content-type of the HTTP Request.

> EXAMPLE 19
>
> Different representations of the bus stops dataset can be served according to the specified content type of the HTTP Request:
> Using `cURL` to get the content of `http://data.mycity.example.com/transport/dataset/bus/stops` represented in CSV and in JSON-LD format.
>
> ```
> curl -L -H "Accept: text/csv" http://data.mycity.example.com/tra
>
> curl -L -H "Accept: application/ld+json" http://data.mycity.examp
> ```

How to Test

Check the available representations of the resource and try to get them specifying the accepted content on the HTTP Request header.

Evidence

Relevant requirements: R-FormatMachineRead, R-FormatMultiple

Benefits

Best Practice 20: Provide real-time access

When data is produced in real time, make it available on the Web in real time or near real-time.

Why

The presence of real-time data on the Web enables access to critical time sensitive data, and encourages the development of real-time Web applications. Real-time access is dependent on real-time data producers making their data readily available to the data publisher. The necessity of providing real-time access for a given application will need to be evaluated on a case by case basis considering refresh rates, latency introduced by data post processing steps, infrastructure availability, and the data needed by consumers. In addition to making data accessible, data publishers may provide additional information describing data gaps, data errors and anomalies, and publication delays.

Intended Outcome

Applications will be able to access time-critical data in real time or near real time, where real-time means a range from milliseconds to a few seconds after the data creation.

Possible Approach to Implementation

A possible approach to implementation is for publishers to configure a Web Service that provides a connection so as real-time data is received by the Web Service it can be instantly made available to consumers by polling or streaming.

If data is checked infrequently by consumers, real-time data can be polled upon consumer request for the most recent data through an API. The data publishers will provide an API to facilitate these read-only requests.

If data is checked frequently by consumers, a streaming data implementation may be more appropriate where data is pushed through an API. While streaming techniques are beyond the scope of this best practice, there are many standard protocols and technologies available (for example Server-sent Events, WebSocket, EventSourceAPI) for clients receiving automatic updates from the server.

EXAMPLE 20

In this example the Transport Agency of MyCity keeps track of all bus GPS data. The API provides consumers real-time status information using a REST API. The API allows the consumer to select:

- Current position of the bus
- Bus arrival time
- Bus status

API Description

Description	API	Parameters
Bus position	`{root}/bus/position/current`	bus_id
Bus arrival time to some stop	`{root}/bus/arrival_time`	bus_id, stop_id
Bus status (Possible return: "on-schedule", "delay", "out-of-service")	`{root}/bus/status`	bus_id

How to Test

To adequately test real time data access, data will need to be tracked from the time it is initially collected to the time it is published and accessed. [PROV-O] can be used to describe these activities. Caution should be used when analyzing real-time access for systems that consist of multiple computer systems. For example, tests that rely on wall clock time stamps may reflect inconsistencies between the individual computer systems as opposed to data publication time latency.

Evidence

Relevant requirements: R-AccessRealTime

Benefits

Best Practice 21: Provide data up to date

Make data available in an up-to-date manner, and make the update frequency explicit.

Why

The availability of data on the Web should closely match the data creation or collection time, perhaps after it has been processed or changed. Carefully synchronizing data publication to the update frequency encourages consumer confidence and data reuse.

Intended Outcome

Data on the Web will be updated in a timely manner so that the most recent data available online generally reflects the most recent data released via any other channel. When new data becomes available, it will be published on the Web as soon as practical thereafter.

Possible Approach to Implementation

New versions of the dataset can be posted to the Web on a regular schedule, following the Best Practices for Data Versioning. Posting to the Web can be made a part of the release process for new versions of the data. Making Web publication a deliverable item in the process and assigning an individual person as responsible for the task can help prevent data becoming out of date. To set consumer expectations for updates going forward, you can include human-readable text stating the expected publication frequency, and you can provide machine-readable metadata indicating the frequency as well.

EXAMPLE 21

Suppose that the update frequency of the bus stops dataset is annual. In order to describe the frequency with which new data is added to the dataset, the property `dct:accrualPeriodicity` can be used. A new version of the dataset (`stops-2016-05-05`) is created to reflect the update schedule of the data. It is important to note that new versions can be created sooner than the schedule calls for, but the publisher should ensure that extra versions are published to the Web as quickly as their scheduled counterparts.

```
:stops-2016-05-05
   a dcat:Dataset ;
   dct:title "Bus stops of MyCity" ;
   dcat:keyword "transport","mobility","bus" ;
   dct:issued "2016-05-05"^^xsd:date ;
   ...
   dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/2(
   ...
   dct:isVersionOf :stops-2015-05-05 ;
   pav:previousVersion stops-2015-12-17 ;
   rdfs:comment "The bus stops dataset was updated to reflect
   owl:versionInfo "1.2" ;
   pav:version "1.2"
   .
```

How to Test

Check that the update frequency is stated and that the most recently published copy on the Web is no older than the date predicted by the stated update frequency.

Evidence

Relevant requirements: R-AccessUptodate

Benefits

Best Practice 22: Provide an explanation for data that is not available

For data that is not available, provide an explanation about how the data can be

accessed and who can access it.

Why

Publishing online documentation about unavailable data provides a means for publishers to explicitly identify knowledge gaps. This provides a contextual explanation for consumer communities thus encouraging use of the data that *is* available.

Intended Outcome

Consumers will know that data that is referred to from the current dataset is unavailable or only available under different conditions.

Possible Approach to Implementation

Depending on the machine/human context there are a variety of ways to indicate data unavailability. Data publishers may publish an HTML document that gives a human-readable explanation for data unavailability. From a machine application interface perspective, appropriate HTTP status codes with customized human-readable messages can be used. Examples of status codes include: 303 (see other), 410 (permanently removed), 503 (service *providing data* unavailable).

> EXAMPLE 22
>
> The dataset created for the bus stops can contain sensitive data, for instance, information about the bus driver. In this case, the publisher provides an explanation informing potential users that the personal data about the bus driver is not available.

How to Test

Where the dataset includes references to data that is no longer available or is not available to all users, check that an explanation of what is missing and instructions for obtaining access (if possible) are given. Check if a legitimate http response code in the 400 or 500 range is returned when trying to get unavailable data.

Evidence

Relevant requirements: R-AccessLevel, R-SensitivePrivacy, R-SensitiveSecurity

Benefits

8.10.1 Data Access APIs

Best Practice 23: Make data available through an API

Offer an API to serve data if you have the resources to do so.

Why

An API offers the greatest flexibility and processability for consumers of your data. It can enable real-time data usage, filtering on request, and the ability to work with the data at an atomic level. If your dataset is large, frequently updated, or highly complex, an API is likely to be the best option for publishing your data.

Intended Outcome

Developers will have programmatic access to the data for use in their own applications, with data updated without requiring effort on the part of consumers. Web applications will be able to obtain specific data by querying a programmatic interface.

Possible Approach to Implementation

Creating an API is a little more involved than posting data for download. It requires some understanding of how to build a Web application. One need not necessarily to build one from scratch, however. If you use a data management platform, such as CKAN, you may be able to enable an existing API. Many Web development frameworks include support for APIs, and there are also frameworks written specifically for building custom APIs.

Rails (Ruby), Django (Python), and Express (NodeJS) are some example Web development frameworks that offer support for building APIs. Examples of API frameworks include Swagger, Apigility, Restify, and Restlet.

> EXAMPLE 23
>
> Besides providing bulk downloads of data about public transport, Adrian decides to offer a more flexible data access mechanism. For this, he develops an API to offer access to bus stops, bus routes and real-time information about bus stops. See the examples of its use.

How to Test

Check if a test client can simulate calls and the API returns the expected responses.

Evidence

Relevant requirements: R-AccessRealTime, R-AccessUpToDate

Benefits

Best Practice 24: Use Web Standards as the foundation of APIs

When designing APIs, use an architectural style that is founded on the technologies of the Web itself.

Why

APIs that are built on Web standards leverage the strengths of the Web. For example, using HTTP verbs as methods and URIs that map directly to individual resources helps to avoid tight coupling between requests and responses, making for an API that is easy to maintain and can readily be understood and used by many developers. The statelessness of the Web can be a strength in enabling quick scaling, and using hypermedia enables rich interactions with your API.

Intended Outcome

Developers who have some experience with APIs based on Web standards, such as REST, will have an initial understanding of how to use the API. The API will also be easier to maintain.

Possible Approaches to Implementation

REST (REpresentational State Transfer)[Fielding][Richardson] is an architectural style that, when used in a Web API, takes advantage of the architecture of the Web itself. A full discussion of how to build a RESTful API is beyond the scope of this document, but there are many resources and a strong community that can help in getting started. There are also many RESTful development frameworks available. If you are already using a Web development framework that supports building REST APIs, consider using that. If not, consider an API-only framework that uses REST.

Another aspect of implementation to consider is making a hypermedia API, one that responds with links as well as data. Links are what make the Web a web, and data APIs can be more useful and usable by including links in their responses. The links can offer additional resources, documentation, and navigation. Even for an API that does not meet all the constraints of REST, returning links in responses can make for a service that is rich and self-documenting.

EXAMPLE 24

An example response for information about a certain bus route from a hypermedia API might look like the following:

```
{
        "code": "200",
        "text": "OK",
        "data": {
                "update_time": "2013-01-01T03:00:02Z",
                "route_id": "52",
                "route_name": "Lexington South",
                "route_description": "Lexington corridor south of
                "route_type": "3"
        },
        "links": [{
    "href": "http://data.mycity.example.com/transport/api/v2/rout
                "rel": "self",
                "type": "application/json",
                "method": "GET"
        }, {
  "href": "http://data.mycity.example.com/transport/api/v2/routes
                "rel": "collection",
                "type": "application/json",
                "method": "GET"
        }, {
    "href": "http://data.mycity.example.com/transport/api/v2/sche
                "rel": "describedby",
                "type": "application/json",
                "method": "GET"
        }, {
    "href": "http://data.mycity.example.com/transport/api/v2/maps
                "rel": "describedby",
                "type": "application/json",
                "method": "GET"
        }]
}
```

How to Test

Check that the service avoids using http as a tunnel for calls to custom methods, and check that URIs do not contain method names.

Evidence

Relevant requirements: R-APIDocumented, R-UniqueIdentifier

Benefits

Best Practice 25: Provide complete documentation for your API

Provide complete information on the Web about your API. Update documentation as you add features or make changes.

Why

Developers are the primary consumers of an API and the documentation is the first clue about its quality and usefulness. When API documentation is complete and easy to understand, developers are probably more willing to continue their journey to use it. Providing comprehensive documentation in one place allows developers to code efficiently. Highlighting changes enables your users to take advantage of new features and adapt their code if needed.

Intended Outcome

Developers will be able to obtain detailed information about each call to the API, including the parameters it takes and what it is expected to return, i.e., the whole set of information related to the API. The set of values — how to use it, notices of recent changes, contact information, and so on — should be described and easily browsable on the Web. It will also enables machines to access the API documentation in order to help developers build API client software.

Possible Approach to Implementation

A typical API reference provides a comprehensive list of the calls the API can handle, describing the purpose of each one, detailing the parameters it allows and what it returns, and giving one or more examples of its use. One nice trend in API documentation is to provide a form in which developers can enter specific calls for testing, to see what the API returns for their use case. There are now tools available for quickly creating this type of documentation, such as Swagger, io-docs, OpenApis, and others. It is important to say that the API should be self-documenting as well, so that calls return helpful information about errors and usage. API users should be able to contact the maintainers with questions, suggestions, or bug reports.

The quality of documentation is also related to usage and feedback from developers. Try to get constant feedback from your users about the documentation.

> EXAMPLE 25
> In order to help developers, the transport agency offers a complete documentation about the API that provides access to data about bus stops and routes. The API documentation includes a list of the calls handled by the API, the corresponding parameters and some examples.

How to Test

Check that every call enabled by your API is described in your documentation. Make sure you provide details of what parameters are required or optional and what each call returns.

Check the Time To First Successful Call (i.e. being capable of doing a successful request to the API within a few minutes will increase the chances that the developer will stick to your API).

Evidence

Relevant requirements: R-APIDocumented

Benefits

Best Practice 26: Avoid Breaking Changes to Your API

Avoid changes to your API that break client code, and communicate any changes in your API to your developers when evolution happens.

Why

When developers implement a client for your API, they may rely on specific characteristics that you have built into it, such as the schema or the format of a response. Avoiding breaking changes in your API minimizes breakage to client code. Communicating changes when they do occur enables developers to take advantage of new features and, in the rare case of a breaking change, take action.

Intended Outcome

Developer code will continue to work. Developers will know of improvements you make and be able to make use of them. Breaking changes to your API will be rare, and if they occur, developers will have sufficient time and information to adapt their code. That will en-

able them to avoid breakage, enhancing trust. Changes to the API will be announced on the API's documentation site.

Possible Approach to Implementation

When improving your API, focus on adding new calls or new options rather than changing how existing calls work. Existing clients can ignore such changes and will continue functioning.

If using a fully RESTful style, you should be able to avoid changes that affect developers by keeping resource URIs constant and changing only elements that your users do not code to directly. If you need to change your data in ways that are not compatible with the extension points that you initially designed, then a completely new design is called for, and that means changes that break client code. In that case, it’s best to implement the changes as a new REST API, with a different resource URI.

If using an architectural style that does not allow you to make moderately significant changes without breaking client code, use versioning. Indicate the version in the response header. Version numbers should be reflected in your URIs or in request "accept" headers (using content negotiation). When versioning in URIs, include the version number as far to the left as possible. Keep the previous version available for developers whose code has not yet been adapted to the new version.

EXAMPLE 26

Some examples of breaking changes to an API include:

- Removing a call;
- Changing the method used to make a call;
- Changing the URI of a resource used in a call;
- Adding a required parameter for a call;
- Changing the data type of a parameter;
- Changing the name of a key in a key-value response;
- Changing the structure of an XML response
- Changing the data type of a value in a response, such as changing a string to an array;

Suppose the MyCity transit agency's API responds to a request for a certain bus's arrival time at a single station as `http://data.mycity.example.com/transport/api/arrivals/buses/53/stop/12` , but the agency decides it wants to make it possible to query for a range of stops at once. Rather than change the form of the request to require a range, like `http://data.mycity.example.com/transport/api/arrivals/buses/53/stop/12-12` , the agency can keep the old API call and add a new one for multiple arrivals, like `http://data.mycity.example.com/transport/api/arrivals/buses/53/stops/1-12` .

To notify users directly of changes, it is a good idea to create a mailing list and encourage developers to join. You can then announce changes there, and this provides a nice mechanism for feedback as well. It also allows your users to help each other.

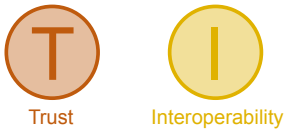
How to Test

Release changes initially to a test version of your API before applying them to the production version. Invite developers to test their applications on the test version and provide feedback.

Evidence

Relevant requirements: R-PersistentIdentification, R-APIDocumented

Benefits



8.11 Data Preservation

The working group recognizes that it is unrealistic to assume that all data on the Web will be available on demand at all times into the indefinite future. For a wide variety of reasons, data publishers are likely to want or need to remove data from the live Web, at which point it moves out of scope for the current work and into the scope of data archivists. What *is* in scope here, however, is what is left behind, that is, what steps should publishers take to indicate that data has been removed or archived. Simply deleting a resource from the Web is bad practice. In that circumstance, dereferencing the URI would lead to an HTTP Response code of 404 that tells the user nothing other than that the resource was not found. The following Best Practices offer more productive approaches.

Best Practice 27: Preserve identifiers

When removing data from the Web, preserve the identifier and provide information about the archived resource.

Why

URI dereferencing is the primary interface to data on the Web. If dereferencing a URI leads to the infamous 404 response code (Not Found), the user will not know whether the lack of availability is permanent or temporary, planned or accidental. If the publisher, or a third party, has archived the data, that archived copy is much less likely to be found if the original URI is effectively broken.

Intended Outcome

The URI of a resource will always dereference to the resource or redirect to information about it.

Possible Approach to Implementation

There are two scenarios to consider:

1. the resource has been deleted entirely and is no longer available via any route;
2. the resource has been archived and is only available through a request to the archive.

In the first of these cases, the server should be configured to respond with an HTTP Response code of 410 (Gone). From the specification:

> The 410 response is primarily intended to assist the task of Web maintenance by notifying the recipient that the resource is intentionally unavailable and that the server owners desire that remote links to that resource be removed.

In the second case, where data has been archived, it is more appropriate to redirect requests to a Web page giving information about the archive that holds the data and how a potential user can access it.

In both cases, the original URI continues to identify the resource and leads to useful information, even though the data is no longer directly available.

EXAMPLE 27

Adrian decides to archive versions of the bus stop data that have been superseded by more recent versions for at least a year, such as the version from 2012-03-30. The server is configured such that requests for the March 2012 dataset are redirected, using HTTP Code 303, to a Web page that includes the following notice.

> **Archived**
>
> The data you requested has been archived. This is inline with the MyCity policy of archiving data that was superseded more than 12 months ago. However, a copy can be requested at any time via the contact page.

How to Test

Check that dereferencing the URI of a dataset that is no longer available returns information about its current status and availability, using either a 410 or 303 Response Code as appropriate.

Evidence

Relevant requirements:R-AccessLevel, R-PersistentIdentification

Benefits

Best Practice 28: Assess dataset coverage

Assess the coverage of a dataset prior to its preservation.

Why

A chunk of Web data is by definition dependent on the rest of the global graph. This global context influences the meaning of the description of the resources found in the dataset. Ideally, the preservation of a particular dataset would involve preserving all its context. That is the entire Web of Data.

At the time of archiving, an evaluation of the linkage of the dataset dump to already preserved resources, and the vocabularies used, needs to be assessed. Datasets for which very few of the vocabularies used and/or resources pointed to are already preserved somewhere should be flagged as being at risk.

Intended Outcome

Users will be able to make use of archived data well into the future.

Possible Approach to Implementation

Check whether all the resources used are either already preserved somewhere or need to be provided along with the dataset being considered for preservation.

EXAMPLE 28

An RDF dataset targeted for preservation is made of the following triples:

```
<http://data.mycity.example.com/transport/route/bus/ABtimetable
    a gtfs:Route ;
    gtfs:color "ff0000" ;
    gtfs:shortname "10" ;
    gtfs:longName "Airport - Bullfrog" ;
    gtfs:agency <http://data.mycity.example.com/transport-agenc
    gtfs:routeType ex:three ;
    ex:usualVehicleType dbpedia:Routemaster ;
    foaf:isPrimaryTopicOf ex:Airport_Bullfrog
    .

<http://data.mycity.example.com/ransport/route/bus/BFC>
    a gtfs:Route ;
    gtfs:color "ffff00" ;
    gtfs:shortname "20" ;
    gtfs:longName "Bullfrog - Furnace Creek Resort" ;
    gtfs:agency <http://data.mycity.example.com/transport-agenc
    gtfs:routeType ex:three ;
    ex:usualVehicleType dbpedia:Articulated_bus ;
    foaf:isPrimaryTopicOf ex:Bullfrog_Furnace_Creek_Resort
    .
…
```

Those triples make use of the "gtfs" vocabulary and a custom one defined in the testing domain name "ex". It also uses entities defined in "foaf", "dbpedia" and "ex". Although not formal standards, FOAF and GTFS [GTFS] are well established ontologies that are archived in several places on the Web (see, for instance, the LOV repository). Entities defined in DBpedia are also preserved through their Memento gateway and archived dumps of the dataset also exist. The risks associated to preserving the triple making use of those external resource is thus minimal. A bigger concern arises from the usage made of resources defined in "ex" which is a namespace that, by design, does not exist outside of the dataset. Unless the data describing "ex:usualVehicleType", "ex:Airport_Bullfrog" and "ex:Bullfrog_Furnace_Creek_Resort" is preserved alongside those triples their contextual meaning will be lost. This is particularly critical for "ex:usualVehicleType" as without it the relationship between the described route and the dbpedia resources will be unknown to a consuming application (however obvious it may be to a human).

Considering this assessment, a revised dataset including the definition of "ex:usualVehicleType" can be considered for preservation:

```
<http://data.mycity.example.com/transport/route/bus/AB> a gtfs:R
     gtfs:color "ff0000" ;
     gtfs:shortname "10" ;
     gtfs:longName "Airport - Bullfrog" ;
     gtfs:agency <http://data.mycity.example.com/transport-agenc
     gtfs:routeType ex:three ;
     ex:usualVehicleType dbpedia:Routemaster ;
     foaf:isPrimaryTopicOf ex:Airport_Bullfrog
     .

 <http://data.mycity.example.com/transport/route/bus/BFC>
     a gtfs:Route;
     gtfs:color "ffff00";
     gtfs:shortname "20";
     gtfs:longName "Bullfrog - Furnace Creek Resort";
     gtfs:agency <http://data.mycity.example.com/transport-agenc
     gtfs:routeType ex:three;
     ex:usualVehicleType dbpedia:Articulated_bus;
     foaf:isPrimaryTopicOf ex:Bullfrog_Furnace_Creek_Resort
     .
 …

 # Custom vocabulary element
 ex:usualVehicleType
     a rdf:Property ;
     rdfs:subPropertyOf gtfs:routeType ;
     rdfs:range gtfs:Bus
     .
```

This second, more complete, dataset is better suited for preservation as it is more self-describing and only makes use of external entities whose preservation is trusted.

How to Test

It is impossible to determine what will be available in, say, 50 years' time. However, one can check that an archived dataset depends only on widely used external resources and vocabularies. Check that unique or lesser-used dependencies are preserved as part of the archive.

Evidence

Relevant requirements:R-VocabReference

Benefits

8.12 Feedback

Publishing on the Web enables data sharing on a large scale to a wide range of audiences with different levels of expertise. Data publishers want to ensure that the data published is meeting the data consumer needs and for this purpose, user feedback is crucial. Feedback has benefits for both publishers and consumers, helping data publishers to improve the integrity of their published data, as well as encouraging the publication of new data. Feedback allows data consumers to have a voice describing usage experiences (e.g. applications using data), preferences and needs. When possible, feedback should also be publicly available for other data consumers to examine. Making feedback publicly available allows users to become aware of other data consumers, supports a collaborative environment, and allows user community experiences, concerns or questions are currently being addressed.

From a user interface perspective there are different ways to gather feedback from data consumers, including site registration, contact forms, quality ratings selection, surveys and comment boxes for blogging. From a machine perspective the data publisher can also record metrics on data usage or information about specific applications that use the data. Feedback such as this establishes a communication channel between data publishers and data consumers. Publicly available feedback should be displayed in a human-readable form.

This section provides some Best Practices to be followed by data publishers in order to enable consumers to provide feedback. This feedback can be for humans or machines.

Best Practice 29: Gather feedback from data consumers

Provide a readily discoverable means for consumers to offer feedback.

Why

Obtaining feedback helps publishers understand the needs of their data consumers and can help them improve the quality of their published data. It also enhances trust by showing consumers that the publisher cares about addressing their needs. Specifying a clear feedback mechanism removes the barrier of having to search for a way to provide feedback.

Intended Outcome

Data consumers will be able to provide feedback and ratings about datasets and distributions.

Possible Approach to Implementation

Provide data consumers with one or more feedback mechanisms including, but not limited to, a contact form, point and click data quality rating buttons, or a comment box. In order to make the most of feedback received from consumers, it is a good idea to collect the feedback with a tracking system that captures each item in a database, enabling quantification and analysis. It is also a good idea to capture the type of each item of feedback, i.e., its motivation (editing, classifying [rating], commenting or questioning), so that each item can be expressed using the Dataset Usage Vocabulary [VOCAB-DUV].

> EXAMPLE 29
>
> Example feedback form

How to Test

Check that at least one feedback mechanism is provided and readily discoverable by data consumers.

Evidence

Relevant requirements: R-UsageFeedback, R-QualityOpinions

Benefits

Best Practice 30: Make feedback available

Make consumer feedback about datasets and distributions publicly available.

Why

By sharing feedback with consumers, publishers can demonstrate to users that their concerns are being addressed, and they can avoid submission of duplicate bug reports. Sharing feedback also helps consumers understand any issues that may affect their ability to use the data, and it can foster a sense of community among them.

Intended Outcome

Consumers will be able to assess the kinds of errors that affect the dataset, review other users' experiences with it, and be reassured that the publisher is actively addressing issues as needed. Consumers will also be able to determine whether other users have already pro-

vided similar feedback, saving them the trouble of submitting unnecessary bug reports and sparing the maintainers from having to deal with duplicates.

Possible Approach to Implementation

Feedback can be available as part of an HTML Web page, but it can also be provided in a machine-readable format using the Dataset Usage Vocabulary [VOCAB-DUV].

EXAMPLE 30

```
:stops-2015-05-05
    a dcat:Dataset ;
    dct:title "Bus stops of MyCity" ;
    dcat:keyword "transport","mobility","bus" ;
    dct:issued "2015-05-05"^^xsd:date ;
    dcat:contactPoint <http://data.mycity.example.com/transport
    dct:temporal <http://reference.data.gov.uk/id/year/2015> ;
    dct:spatial <http://sws.geonames.org/3399415> ;
    dct:publisher :transport-agency-mycity ;
    dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/20
    dcat:theme :mobility ;
    dcat:distribution :stops-2015-05-05.csv
    .

:stops-2015-05-05.csv
    a dcat:Distribution ;
    dct:title "CSV distribution of stops-2015-05-05 dataset" ;
    dct:description "CSV distribution of the bus stops dataset
    dcat:mediaType "text/csv;charset=UTF-8"
    .
:comment1Content
    a oa:TextualBody ;
    rdf:value "This dataset is missing stop 3"
    .

:comment1
    a oa:Annotation ;
    a duv:UserFeedback ;
    oa:hasBody :comment1Content ;
    oa:hasTarget :stops-2015-05-05 ;
    dct:creator :localresident ;
    oa:motivatedBy oa:assessing
    .

:comment2Content
    a oa:TextualBody ;
    rdf:value "Are tab delimited formats also available?"
    .

:comment2
    a oa:Annotation ;
    a duv:UserFeedback ;
    oa:hasBody :comment2Content ;
    oa:hasTarget :stops-2015-05-05.csv ;
    dct:creator :localresident ;
    oa:motivatedBy oa:assessing
    .
```

```
:localresident
    a foaf:Person ;
    foaf:Name "Alan Law"
    .
```

How to Test

Check that any feedback given by data consumers for a specific dataset or distribution is publicly available.

Evidence

Relevant requirements: R-UsageFeedback, R-QualityOpinions

Benefits

8.13 Data Enrichment

Data enrichment refers to a set of processes that can be used to enhance, refine or otherwise improve raw or previously processed data. This idea and other similar concepts contribute to making data a valuable asset for almost any modern business or enterprise. It is a diverse topic in itself, details of which are beyond the scope of the current document. However, it is worth noting that some of these techniques should be approached with caution, as ethical concerns may arise. In scientific research, care must be taken to avoid enrichment that distorts results or statistical outcomes. For data about individuals, privacy issues may arise when combining datasets. That is, enriching one dataset with another, when neither contains sufficient information about any individual to identify them, may yield a combined dataset that compromises privacy. Furthermore, these techniques can be carried out at scale, which in turn highlights the need for caution.

This section provides some advice to be followed by data publishers in order to enrich data.

Best Practice 31: Enrich data by generating new data

Enrich your data by generating new data when doing so will enhance its value.

Why

Enrichment can greatly enhance processability, particularly for unstructured data. Under some circumstances, missing values can be filled in, and new attributes and measures can

be added from the existing raw data. Datasets can also be enriched by gathering additional results in the same fashion as the original data, or by combining the original data with other datasets. Publishing more complete datasets can enhance trust, if done properly and ethically. Deriving additional values that are of general utility saves users time and encourages more kinds of reuse. There are many intelligent techniques that can be used to enrich data, making the dataset an even more valuable asset.

Intended Outcome

Datasets with missing values will be enhanced by filling in those values. Structure will be conferred and utility enhanced if relevant measures or attributes are added, but only if the addition does not distort analytical results, significance, or statistical power.

Possible Approaches to Implementation

Techniques for data enrichment are complex and go well beyond the scope of this document, which can only highlight the possibilities.

Machine learning can readily be applied to the enrichment of data. Methods include those focused on data categorization, disambiguation, entity recognition, sentiment analysis and topification, among others. New data values may be derived as simply as performing a mathematical calculation across existing columns. Other examples include visual inspection to identify features in spatial data and cross-reference to external databases for demographic information. Lastly, generation of new data may be demand-driven, where missing values are calculated or otherwise determined by direct means.

Values generated by inference-based techniques should be labeled as such, and it should be possible to retrieve any original values replaced by enrichment.

Whenever licensing permits, the code used to enrich the data should be made available along with the dataset. Sharing such code is particularly important for scientific data.

Prioritization of enrichment activities should be based on value to the data consumer as well as the effort required. Value to the consumer can be gauged by measurement of demand (e.g., through surveys or tracking direct requests). Documenting how you measure demand can make the increased value demonstrable.

If you make enrichments to someone else’s data, it’s a good idea to offer those enrichments back to the original publisher.

EXAMPLE 31

1. The MyCity transport agency has street addresses for each of its transit stops. It wants to make it easier for consumers of its data to combine the data with maps, so it adds latitude and longitude information for each stop by utilizing a geographic database.
2. The transit agency has a large collection of email correspondence from transit riders. Some of the correspondence is complimentary, some emails are complaints, and some are requests for information. The agency conducts a combination of sentiment analysis and categorization to extract metadata for each of the messages, such as transit mode, route number, and rider positivity, to create a semi-structured dataset.

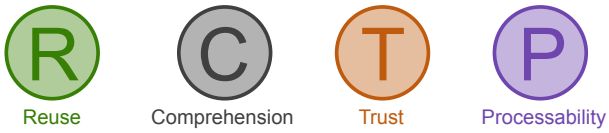
How to Test

Verify that there are no missing values in the dataset, or additional fields likely to be needed by others, that could readily be provided. Check that any data added by inferential enrichment techniques is identified as such and that any replaced data is still available.

Evidence

Relevant requirements: R-DataEnrichment, R-FormatMachineRead, R-ProvAvailable

Benefits



Best Practice 32: Provide Complementary Presentations

Enrich data by presenting it in complementary, immediately informative ways, such as visualizations, tables, Web applications, or summaries.

Why

Data published online is meant to inform others about its subject. But only posting datasets for download or API access puts the burden on consumers to interpret it. The Web offers unparalleled opportunities for presenting data in ways that let users learn and explore without having to create their own tools.

Intended Outcome

Complementary data presentations will enable human consumers to have immediate insight into the data by presenting it in ways that are readily understood.

Possible Approaches to Implementation

One very simple way to provide immediate insight is to publish an analytical summary in an HTML page. Including summative data in graphs or tables can help users scan the summary and quickly understand the meaning of the data.

If you have the means to create interactive visualizations or Web applications that use the data, you can give consumers of your data greater ability to understand it and discover patterns in it. These approaches also demonstrate its suitability for processing and encourage reuse.

> EXAMPLE 32
>
> The MyCity transit agency publishes detailed data about all its transit lines through an API, but it also has many users who are not Web developers and who want to know how to use the system to move about the city. The transit agency could build a Web application that allows users to enter a departure address and a destination and receive step-by-step directions for making their journey via public transit.

How to Test

Check that the dataset is accompanied by some additional interpretive content that can be perceived without downloading the data or invoking an API.

Evidence

Relevant requirements: R-DataEnrichment

Benefits

8.14 Republication

Reusing data is another way of publishing data; it is simply republishing. It can take the form of combining existing data with other datasets, creating Web applications or visualizations, or repackaging the data in a new form, such as a translation. Data republishers have some responsibilities that are

unique to that form of publishing on the Web. This section provides advice to be followed when re-publishing data.

Best Practice 33: Provide Feedback to the Original Publisher

Let the original publisher know when you are reusing their data. If you find an error or have suggestions or compliments, let them know.

Why

Publishers generally want to know whether the data they publish has been useful. Moreover, they may be required to report usage statistics in order to allocate resources to data publishing activities. Reporting your usage helps them justify putting effort toward data releases. Providing feedback repays the publishers for their efforts by directly helping them to improve their dataset for future users.

Intended Outcome

Better communication will make it easier for original publishers to determine how the data they post is being used, which in turn helps them justify publishing the data. Publishers will also be made aware of steps they can take to improve their data. This leads to more and better data for everyone.

Possible Approach to Implementation

When you begin using a dataset in a new product, make a note of the publisher's contact information, the URI of the dataset you used, and the date on which you contacted them. This can be done in comments within your code where the dataset is used. Follow the publisher's preferred route to provide feedback. If they do not provide a route, look for contact information for the Web site hosting the data.

EXAMPLE 33

```
# Calling the MyCity transit API, http://data.mycity.example.com/
# Published by MyCity Transit Agency,
# notified of our reuse by email to opendata@mycitytransit.exampl
# by Newton Calegari on 3/24/2016.
```

How to Test

Check that you have a record of at least one communication informing the publisher of your use of the data.

Evidence

Relevant requirements: R-TrackDataUsage, R-UsageFeedback, R-QualityOpinions

Benefits

Best Practice 34: Follow Licensing Terms

Find and follow the licensing requirements from the original publisher of the dataset.

Why

Licensing provides a legal framework for using someone else's work. By adhering to the original publisher's requirements, you keep the relationship between yourself and the publisher friendly. You don't need to worry about legal action from the original publisher if you are following their wishes. Understanding the initial license will help you determine what license to select for your reuse.

Intended Outcome

Data publishers will be able to trust that their work is being reused in accordance with their licensing requirements, which will make them more likely to continue to publish data. Reusers of data will themselves be able to properly license their derivative works.

Possible Approach to Implementation

Read the original license and adhere to its requirements. If the license calls for specific licensing of derivative works, choose your license to be compatible with that requirement. If no license is given, contact the original publisher and ask what the license is.

> EXAMPLE 34
>
> If a dataset you are using is licensed under the Creative Commons Attribution 3.0 License, you will need to meet the terms specified in that license agreement.

How to Test

Read through the original license and check that your use of the data does not violate any of the terms.

Evidence

Relevant requirements: R-LicenseAvailable, R-LicenseLiability,

Benefits

Best Practice 35: Cite the Original Publication

Acknowledge the source of your data in metadata. If you provide a user interface, include the citation visibly in the interface.

Why

Data is only useful when it is trustworthy. Identifying the source is a major indicator of trustworthiness in two ways: first, the user can judge the trustworthiness of the data from the reputation of the source, and second, citing the source suggests that you yourself are trustworthy as a republisher. In addition to informing the end user, citing helps publishers by crediting their work. Publishers who make data available on the Web deserve acknowledgment and are more likely to continue to share data if they find they are credited. Citation also maintains provenance and helps still others to work with the data.

Intended Outcome

End users will be able to assess the trustworthiness of the data they see and the efforts of the original publishers will be recognized. The chain of provenance for data on the Web will be traceable back to its original publisher.

Possible Approach to Implementation

You can present the citation to the original source in a user interface by providing bibliographic text and a working link.

> EXAMPLE 35
>
> Data source: MyCity Transport Agency. "Bus timetable of MyCity" (series 1.2). MyCity. May 5, 2015. Available from: http://data.mycity.example.com/transport/dataset/bus/stops.

How to Test

Check that the original source of any reused data is cited in the metadata provided. Check that a human-readable citation is readily visible in any user interface.

Evidence

Relevant requirements: R-Citable, R-ProvAvailable, R-MetadataAvailable, R-TrackDataUsage

Benefits

9. Glossary

This section is non-normative.

Citation

A Citation may be either direct and explicit (as in the reference list of a journal article), indirect (e.g. a citation to a more recent paper by the same research group on the same topic), or implicit (e.g. as in artistic quotations or parodies, or in cases of plagiarism).

From: CiTO, the Citation Typing Ontology.

Data archiving

Data Archiving is the set of practices around the storage and monitoring of the state of digital material over the years.

These tasks are the responsibility of a Trusted Digital Repository (TDR), also sometimes referred to as Long-Term Archive Service (LTA). Often such services follow the Open Archival Information System [OAIS] which defines the archival process in terms of ingest, monitoring and reuse of data.

Data consumer

For the purposes of this WG, a Data Consumer is a person or group accessing, using, and potentially performing post-processing steps on data.

From: Strong, Diane M., Yang W. Lee, and Richard Y. Wang. "Data quality in context." Communications of the ACM 40.5 (1997): 103-110.

Data format

Data Format defined as a specific convention for data representation i.e. the way that information is encoded and stored for use in a computer system, possibly constrained by a formal data type or set of standards."

From: Digital Humanities Curation Guide

Data preservation

Data Preservation is defined by the Alliance for Permanent Access Network as "The processes and operations in ensuring the technical and intellectual survival of objects through time". This is part of a data management plan focusing on preservation planning and meta-data. Whether it is worthwhile to put effort into preservation depends on the (future) value of the data, the resources available and the opinion of the designated community of stakeholders.

Data producer

Data Producer is a person or group responsible for generating and maintaining data.

From: Strong, Diane M., Yang W. Lee, and Richard Y. Wang. "Data quality in context." Communications of the ACM 40.5 (1997): 103-110.

Data provenance

Provenance originates from the French term "provenir" (to come from), which is used to describe the curation process of artwork as art is passed from owner to owner. Data provenance, in a similar way, is metadata that allows data providers to pass details about the data history to data users.

Data quality

Data quality is commonly defined as “fitness for use” for a specific application or use case.

Dataset

A dataset is defined as a collection of data, published or curated by a single agent, and available for access or download in one or more formats. A dataset does not have to be available as a downloadable file.

From: Data Catalog Vocabulary (DCAT) [VOCAB-DCAT]

Distribution

A distribution represents a specific available form of a dataset. Each dataset might be available in different forms; these forms might represent different formats of the dataset or different endpoints. Examples of distributions include a downloadable CSV file, an API or an RSS feed

From: Data Catalog Vocabulary (DCAT) [VOCAB-DCAT]

Feedback

A feedback forum is used to collect messages posted by consumers about a particular topic. Messages can include replies to other consumers. Datetime stamps are associated with each message and the messages can be associated with a person or submitted anonymously.

From: Semantically-Interlinked Online Communities (SIOC) and the Annotation Model [Annotation-Model]

To better understand why an annotation was created, a SKOS Concept Scheme [SKOS-PRIMER] is used to show inter-related annotations between communities with more meaningful distinctions than a simple class/subclass tree.

File format

File Format is a standard way that information is encoded for storage in a computer file. It specifies how bits are used to encode information in a digital storage medium. File formats may be either proprietary or free and may be either unpublished or open.

Examples of file formats include: plain text (in a specified character encoding, ideally UTF-8), Comma Separated Variable (CSV) [RFC4180], Portable Document Format [PDF], XML, JSON [RFC4627], Turtle [Turtle] and HDF5.

License

A license is a legal document giving official permission to do something with the data with which it is associated.

From: DCTERMS [DCTERMS]

Locale

A collection of international preferences, generally related to a language and geographic region that a (certain category) of users require. These are usually identified by a shorthand identifier or token, such as a language tag, that is passed from the environment to various processes to get culturally affected behavior.

From Language Tags and Locale Identifiers for the World Wide Web [LTLI].

Machine-readable data

Machine-readable data is data in a standard format that can be read and processed automatically by a computing system. Traditional word processing documents and portable document format (PDF) files are easily read by humans but typically are difficult for machines to interpret and manipulate. Formats such as XML, JSON, HDF5, RDF and CSV are machine-readable data formats

Adapted from Wikipedia.

Near real-time

The term "near real-time" or "nearly real-time" (NRT), in telecommunications and computing, refers to the time delay introduced, by automated data processing or network transmission, between the occurrence of an event and the use of the processed data, such as for display or feedback and control purposes. For example, a near-real-time display depicts an event or situation as it existed at the current time minus the processing time, as nearly the time of the live event.

From: Wikipedia

Sensitive data

Sensitive data is any designated data or metadata that is used in limited ways and/or intended for limited audiences. Sensitive data may include personal data, corporate or government data, and mishandling of published sensitive data may lead to damages to individuals or organizations.

Standard

A technical standard is an established norm or requirement in regard to technical systems. It is usually a formal document that establishes uniform engineering or technical criteria, methods,

processes and practices. In contrast, a custom, convention, company product, corporate standard, etc. that becomes generally accepted and dominant is often called a de facto standard.

From: Wikipedia

Structured data

Structured Data refers to data that conforms to a fixed schema. Relational databases and spreadsheets are examples of structured data.

Vocabulary

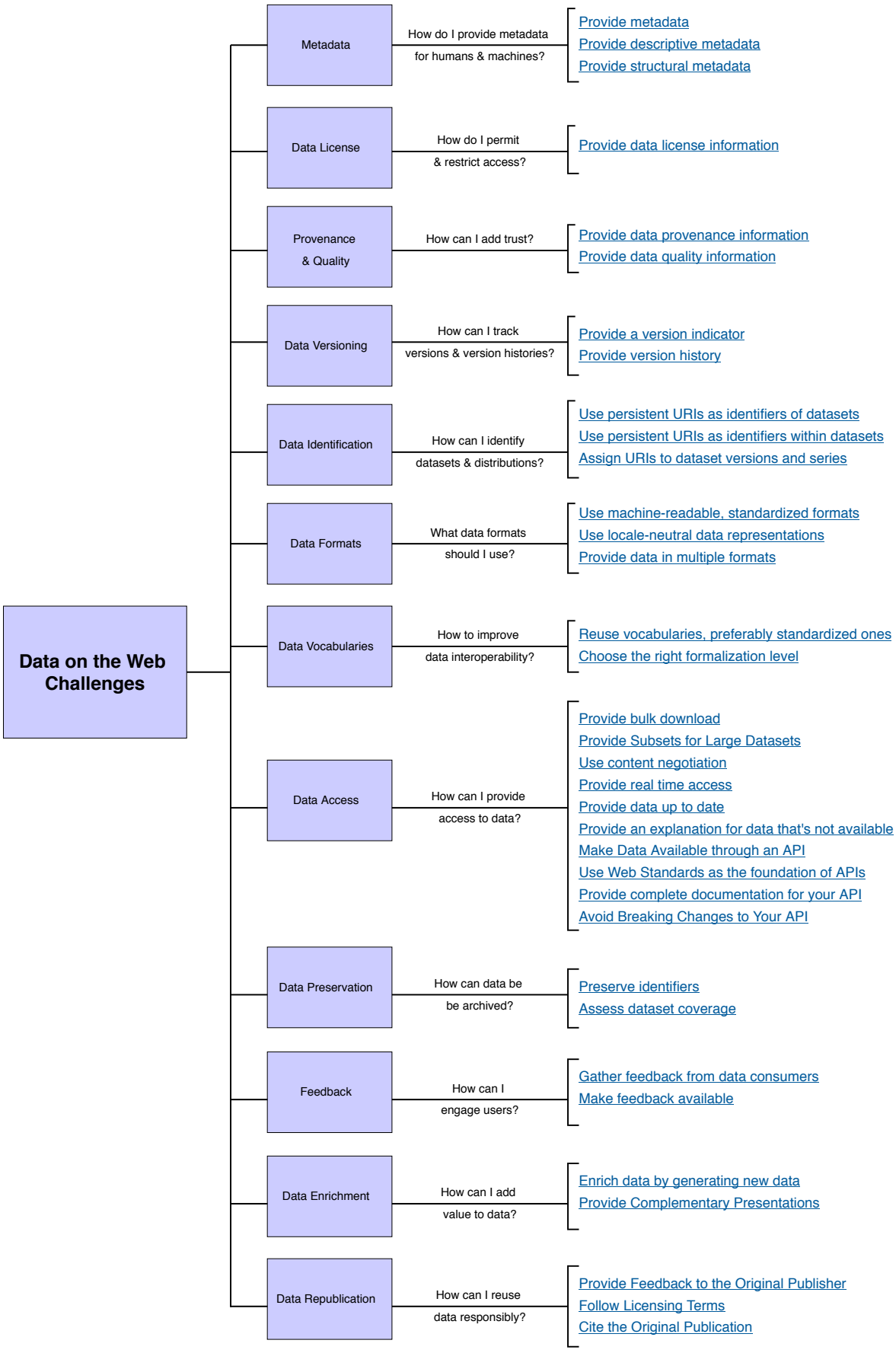
A vocabulary is a collection of "terms" for a particular purpose. Vocabularies can range from simple such as the widely used RDF Schema [RDF-SCHEMA], FOAF [FOAF] and Dublin Core [DCTERMS] to complex vocabularies with thousands of terms, such as those used in healthcare to describe symptoms, diseases and treatments. Vocabularies play a very important role in Linked Data, specifically to help with data integration. The use of this term overlaps with Ontology.

From: Linked Data Glossary

10. Data on the Web Challenges

This section is non-normative.

The following diagram summarizes some of the main challenges faced when publishing or consuming data on the Web. These challenges were identified from the DWBP Use Cases and Requirements [DWBP-UCR] and, as presented in the diagram, is addressed by one or more Best Practices.


Data on the Web Challenges
Metadata
How do I provide metadata for humans & machines?
Provide metadata
Provide descriptive metadata
Provide structural metadata
Data License
How do I permit & restrict access?
Provide data license information
Provenance & Quality
How can I add trust?
Provide data provenance information
Provide data quality information
Data Versioning
How can I track versions & version histories?
Provide a version indicator
Provide version history
Data Identification
How can I identify datasets & distributions?
Use persistent URIs as identifiers of datasets
Use persistent URIs as identifiers within datasets
Assign URIs to dataset versions and series
Data Formats
What data formats should I use?
Use machine-readable, standardized formats
Use locale-neutral data representations
Provide data in multiple formats
Data Vocabularies
How to improve data interoperability?
Reuse vocabularies, preferably standardized ones
Choose the right formalization level
Data Access
How can I provide access to data?
Provide bulk download
Provide Subsets for Large Datasets
Use content negotiation
Provide real time access
Provide data up to date
Provide an explanation for data that's not available
Make Data Available through an API
Use Web Standards as the foundation of APIs
Provide complete documentation for your API
Avoid Breaking Changes to Your API
Data Preservation
How can data be be archived?
Preserve identifiers
Assess dataset coverage
Feedback
How can I engage users?
Gather feedback from data consumers
Make feedback available
Data Enrichment
How can I add value to data?
Enrich data by generating new data
Provide Complementary Presentations
Data Republication
How can I reuse data responsibly?
Provide Feedback to the Original Publisher
Follow Licensing Terms
Cite the Original Publication

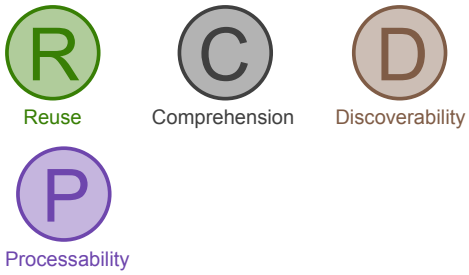
11. Best Practices Benefits

This section is non-normative.

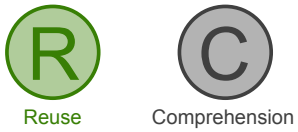
The list below describes the main benefits of applying the DWBP. Each benefit represents an improvement in the way how datasets are available on the Web.

- Comprehension: humans will have a better understanding about the data structure, the data meaning, the metadata and the nature of the dataset.
- Processability: machines will be able to automatically process and manipulate the data within a dataset.
- Discoverability machines will be able to automatically discover a dataset or data within a dataset.
- Reuse: the chances of dataset reuse by different groups of data consumers will increase.
- Trust: the confidence that consumers have in the dataset will improve.
- Linkability: it will be possible to create links between data resources (datasets and data items).
- Access: humans and machines will be able to access up to date data in a variety of forms.
- Interoperability: it will be easier to reach consensus among data publishers and consumers.

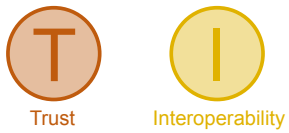
The following table relates Best Practices and Benefits.

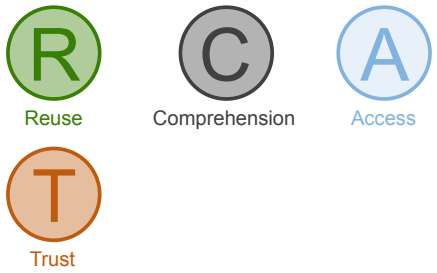
Best Practice	**Benefits**
Provide metadata	
Provide descriptive metadata	
Provide structural metadata	

Best Practice	Benefits
Provide data license information	
Provide data provenance information	
Provide data quality information	
Provide a version indicator	
Provide version history	
Use persistent URIs as identifiers of datasets	
Use persistent URIs as identifiers within datasets	

Best Practice	Benefits
Assign URIs to dataset versions and series	
Use machine-readable standardized data formats	
Use locale-neutral data representations	
Provide data in multiple formats	
Reuse vocabularies, preferably standardized ones	
Choose the right formalization level	
Provide bulk download	

Best Practice	Benefits
Provide Subsets for Large Datasets	
Use content negotiation for serving data available in multiple formats	
Provide real-time access	
Provide data up to date	
Provide an explanation for data that is not available	
Make data available through an API	

Best Practice	Benefits
Use Web Standards as the foundation of APIs	
Provide complete documentation for your API	
Avoid Breaking Changes to Your API	
Preserve identifiers	
Assess dataset coverage	
Gather feedback from data consumers	
Make feedback available	

Best Practice	Benefits
Enrich data by generating new data	
Provide Complementary Presentations	
Provide Feedback to the Original Publisher	
Follow Licensing Terms	
Cite the Original Publication	

Best Practices and Benefits

The figure below shows the benefits that data publishers will gain with adoption of the Best Practices.

REUSE

All Best Practices

ACCESS

Provide bulk download

Provide Subsets for Large Datasets

Use content negotiation for serving data available in multiple formats

Provide real-time access

Provide data up to date

Make data available through an API

Use Web Standards as the foundation of APIs

Provide Complementary Presentations

DISCOVERABILITY

Provide metadata

Provide descriptive metadata

Use persistent URIs as identifiers of datasets

Use persistent URIs as identifiers within datasets

Assign URIs to dataset versions and series

Use Web Standards as the foundation of APIs

Cite the Original Publication

PROCESSABILITY

Provide metadata

Provide structural metadata

Use machine-readable standardized data formats

Provide data in multiple formats

Reuse vocabularies, preferably standardized ones

Provide Subsets for Large Datasets

Make data available through an API

Use Web Standards as the foundation of APIs

Enrich data by generating new data

TRUST

Provide data license information

Provide data provenance information

Provide data quality information

Provide a version indicator

Provide version history

Assign URIs to dataset versions and series

Reuse vocabularies, preferably standardized ones

Provide an explanation for data that is not available

Provide complete documentation for your API

Avoid Breaking Changes to Your API

Preserve identifiers

Assess dataset coverage

Gather feedback from data consumers

Make feedback available

Enrich data by generating new data

Provide Complementary Presentations

Provide Feedback to the Original Publisher

Follow Licensing Terms

Cite the Original Publication

INTEROPERABILITY

Use persistent URIs as identifiers of datasets

Use persistent URIs as identifiers within datasets

Reuse vocabularies, preferably standardized ones

Choose the right formalization level

Make data available through an API

Use Web Standards as the foundation of APIs

Avoid Breaking Changes to Your API

Provide Feedback to the Original Publisher

LINKABILITY

Use persistent URIs as identifiers of datasets

Use persistent URIs as identifiers within datasets

Provide Subsets for Large Datasets

Use Web Standards as the foundation of APIs

COMPREHENSION

Provide metadata

Provide descriptive metadata

Provide structural metadata

Provide data provenance information

Use locale-neutral data representations

Reuse vocabularies, preferably standardized ones

Choose the right formalization level

Gather feedback from data consumers

Enrich data by generating new data

Provide Complementary Presentations

12. Use Cases Requirements x Best Practices

This section is non-normative.

Requirement	Best Practices
R-MetadataAvailable	Provide metadata Provide descriptive metadata Provide structural metadata Provide data provenance information Use locale-neutral data representations Cite the Original Publication
R-MetadataDocum	Provide metadata Reuse vocabularies, preferably standardized ones
R-MetadataMachineRead	Provide metadata Provide descriptive metadata Provide data license information
R-MetadataStandardized	Provide descriptive metadata Reuse vocabularies, preferably standardized ones
R-LicenseAvailable	Provide data license information Follow Licensing Terms
R-LicenseLiability	Provide data license information Follow Licensing Terms

Requirement	Best Practices
R-ProvAvailable	Provide data provenance information Enrich data by generating new data Cite the Original Publication
R-QualityMetrics	Provide data quality information
R-DataMissingIncomplete	Provide data quality information
R-QualityOpinions	Provide data quality information Gather feedback from data consumers Make feedback available Provide Feedback to the Original Publisher
R-DataVersion	Provide a version indicator Provide version history
R-UniqueIdentifier	Use persistent URIs as identifiers of datasets Use persistent URIs as identifiers within datasets Assign URIs to dataset versions and series Provide Subsets for Large Datasets Use Web Standards as the foundation of APIs
R-Citable	Use persistent URIs as identifiers of datasets Assign URIs to dataset versions and series Provide Subsets for Large Datasets Cite the Original Publication

Requirement	Best Practices
R-FormatMachineRead	Use machine-readable standardized data formats Use locale-neutral data representations Use content negotiation for serving data available in multiple formats Enrich data by generating new data
R-FormatStandardized	Use machine-readable standardized data formats
R-FormatOpen	Use machine-readable standardized data formats
R-FormatLocalize	Use locale-neutral data representations
R-GeographicalContext	Use locale-neutral data representations
R-FormatMultiple	Provide data in multiple formats Use content negotiation for serving data available in multiple formats
R-QualityComparable	Reuse vocabularies, preferably standardized ones Choose the right formalization level
R-VocabOpen	Reuse vocabularies, preferably standardized ones
R-VocabReference	Reuse vocabularies, preferably standardized ones Choose the right formalization level Assess dataset coverage
R-AccessBulk	Provide bulk download
R-GranularityLevels	Provide Subsets for Large Datasets

Requirement	Best Practices
R-AccessRealTime	Provide Subsets for Large Datasets Provide real-time access Make data available through an API
R-AccessUpToDate	Provide data up to date Make data available through an API
R-AccessLevel	Provide an explanation for data that is not available Preserve identifiers
R-SensitivePrivacy	Provide an explanation for data that is not available
R-SensitiveSecurity	Provide an explanation for data that is not available
R-APIDocumented	Use Web Standards as the foundation of APIs Provide complete documentation for your API Avoid Breaking Changes to Your API
R-PersistentIdentification	Avoid Breaking Changes to Your API Preserve identifiers
R-UsageFeedback	Gather feedback from data consumers Make feedback available Provide Feedback to the Original Publisher
R-DataEnrichment	Enrich data by generating new data Provide Complementary Presentations

Requirement	Best Practices
R-TrackDataUsage	Provide Feedback to the Original Publisher Cite the Original Publication

Requirements x Best Practices

A. Acknowledgements

The editors gratefully acknowledge the contributions made to this document by all members of the working group. Especially Annette Greiner's great effort and the contributions received from Antoine Isaac, Eric Stephan and Phil Archer.

This document has benefited from inputs from many members of the Spatial Data on the Web Working Group. Specific thanks are due to Andrea Perego, Dan Brickley, Linda van den Brink and Jeremy Tandy.

The editors would also like to thank comments received from Addison Phillips, Adriano Machado, Adriano Veloso, Andreas Kuckartz, Augusto Herrmann, Bart van Leeuwen, Christophe Gueret, Erik Wilde, Giancarlo Guizzardi, Gisele Pappa, Gregg Kellogg, Herbert Van de Sompel, Ivan Herman, Leigh Dodds, Lewis John McGibbney, Makx Dekkers, Manuel Tomas Carrasco-Benitez, Maurino Andrea, Michel Dumontier, Nandana Mihindukulasooriya, Nathalia Sautchuk Patrício, Peter Winstanley, Renato Iannella, Steven Adler, Vagner Diniz, and Wagner Meira.

The editors also gratefully acknowledge the chairs of this Working Group: Deirdre Lee, Hadley Beeman, Yaso Córdova and the staff contact Phil Archer.

B. Change history

Changes since the previous version:

- The introduction was rearranged and slightly extended to make it clear that DWBP is a general document that is complemented by more detailed work in specific areas.
- Additional attributes were added to the SVG diagrams to increase their accessibility.
- Descriptive names added to the namespaces table
- The name used in the running example, John, changed for the mre gender-neutral Adrian.

C. References

C.1 Informative references

[Annotation-Model]

Robert Sanderson; Paolo Ciccarese; Benjamin Young. W3C. *Web Annotation Data Model*. 17 January 2017. W3C Proposed Recommendation. URL: https://www.w3.org/TR/annotation-model/

[BCP47]

A. Phillips; M. Davis. IETF. *Tags for Identifying Languages*. September 2009. IETF Best Current Practice. URL: https://tools.ietf.org/html/bcp47

[BNF]

Bibliothèque nationale de France. *Reference information about authors, works, topics*. URL: http://data.bnf.fr/

[CCREL]

Hal Abelson; Ben Adida; Mike Linksvayer; Nathan Yergler. W3C/Creative Commons. *ccREL: The Creative Commons Rights Expression Language*. 1 May 2008. W3C Member Submission. URL: http://www.w3.org/Submission/ccREL/

[CLDR]

Unicode Consortium. *Unicode Common Locale Data Repository*. URL: http://cldr.unicode.org/

[DCTERMS]

Dublin Core metadata initiative. *DCMI Metadata Terms*. 14 June 2012. DCMI Recommendation. URL: http://dublincore.org/documents/dcmi-terms/

[DWBP-UCR]

Deirdre Lee; Bernadette Farias Loscio; Phil Archer. W3C. *Data on the Web Best Practices Use Cases & Requirements*. 24 February 2015. W3C Note. URL: https://www.w3.org/TR/dwbp-ucr/

[FOAF]

Dan Brickley; Libby Miller. FOAF project. *FOAF Vocabulary Specification 0.99 (Paddington Edition)*. 14 January 2014. URL: http://xmlns.com/foaf/spec/

[Fielding]

Roy Thomas Fielding. University of California, Irvine. *Representational State Transfer (REST), Chapter 5 of Architectural Styles and the Design of Network-based Software Architectures*. 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[GS1]

Mark Harrison; Ken Traub. GS1. *SmartSearch Implementation Guideline*. November 2015. URL: http://www.gs1.org/gs1-smartsearch/guideline/gtin-web-implementation-guideline

[GTFS]

Pieter Colpaert; Andrew Byrd. *General Transit Feed Specification*. URL: http://vocab.gtfs.org/terms#

[HTML-RDFA]

Manu Sporny. W3C. *HTML+RDFa 1.1 - Second Edition*. 17 March 2015. W3C Recommendation. URL: https://www.w3.org/TR/html-rdfa/

[ISO-25964]

Stella Dextre Clarke et al. ISO/NISO. *ISO 25964 – the international standard for thesauri and interoperability with other vocabularies*. URL: http://www.niso.org/schemas/iso25964/

[ISO639-1-LOC]

Library of Congress. *Ontology for ISO 639-1 Languages*. URL: http://id.loc.gov/ontologies/iso639-1_Languages

[JSON-LD]

Manu Sporny; Gregg Kellogg; Markus Lanthaler. W3C. *JSON-LD 1.0*. 16 January 2014. W3C Recommendation. URL: https://www.w3.org/TR/json-ld/

[LD-BP]

Bernadette Hyland; Ghislain Auguste Atemezing; Boris Villazón-Terrazas. W3C. *Best Practices for Publishing Linked Data*. 9 January 2014. W3C Note. URL: https://www.w3.org/TR/ld-bp/

[LODC]

Max Schmachtenberg; Christian Bizer; Anja Jentzsch; Richard Cyganiak. *The Linking Open Data Cloud Diagram*. URL: http://lod-cloud.net/

[LTLI]

Felix Sasaki; Addison Phillips. W3C. *Language Tags and Locale Identifiers for the World Wide Web*. 23 April 2015. W3C Working Draft. URL: https://www.w3.org/TR/ltli/

[Navathe]

Ramez Elmasri; Shamkant B. Navathe. Addison Wesley. *Fundamentals of Database Systems*. 2010.

[OAIS]

ISO/TC 20/SC 13. ISO. *Space data and information transfer systems -- Open archival information system (OAIS) -- Reference model*. 21 August 2012. ISO Standard. URL: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=57284

[ODB]

World Wide Web Foundation. *Open Data Barometer*. URL: http://opendatabarometer.org

[ODRL-model]

Renato Iannella; Serena Villata. W3C. *ODRL Information Model*. 21 July 2016. W3C Working Draft. URL: https://www.w3.org/TR/odrl-model/

[ODRS]

Leigh Dodds. The Open Data Institute. *Open Data Rights Statement Vocabulary*. 29 July 2013. URL: http://schema.theodi.org/odrs/

[OKFN-INDEX]

Open Knowledge Foundation. *Global Open Data Index*. URL: http://index.okfn.org/

[OWL2-OVERVIEW]

W3C OWL Working Group. W3C. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. 11 December 2012. W3C Recommendation. URL: https://www.w3.org/TR/owl2-overview/

[OWL2-PROFILES]

Boris Motik; Bernardo Cuenca Grau; Ian Horrocks; Zhe Wu; Achille Fokoue. W3C. *OWL 2 Web Ontology Language Profiles (Second Edition)*. 11 December 2012. W3C Recommendation.

URL: https://www.w3.org/TR/owl2-profiles/

[OWL2-QUICK-REFERENCE]

Jie Bao; Elisa Kendall; Deborah McGuinness; Peter Patel-Schneider. W3C. *OWL 2 Web Ontology Language Quick Reference Guide (Second Edition)*. 11 December 2012. W3C Recommendation. URL: https://www.w3.org/TR/owl2-quick-reference/

[PAV]

Paolo Ciccarese; Stian Soiland-Reyes. *PAV - Provenance, Authoring and Versioning*. 28 August 2014. URL: http://purl.org/pav/

[PDF]

Document management — Portable document format — Part 1: PDF. ISO.

[PROV-O]

Timothy Lebo; Satya Sahoo; Deborah McGuinness. W3C. *PROV-O: The PROV Ontology*. 30 April 2013. W3C Recommendation. URL: https://www.w3.org/TR/prov-o/

[PROV-Overview]

Paul Groth; Luc Moreau. W3C. *PROV-Overview*. 30 April 2013. W3C Note. URL: https://www.w3.org/TR/prov-overview/

[PURI]

Phil Archer; Nikos Loutas; Stijn Goedertier; Saky Kourtidis. *Study On Persistent URIs*. 17 December 2012. URL: http://philarcher.org/diary/2013/uripersistence/

[RDA]

Research Data Alliance. URL: http://rd-alliance.org

[RDF-SCHEMA]

Dan Brickley; Ramanathan Guha. W3C. *RDF Schema 1.1*. 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf-schema/

[RFC3986]

T. Berners-Lee; R. Fielding; L. Masinter. IETF. *Uniform Resource Identifier (URI): Generic Syntax*. January 2005. Internet Standard. URL: https://tools.ietf.org/html/rfc3986

[RFC4180]

Y. Shafranovich. IETF. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. October 2005. Informational. URL: https://tools.ietf.org/html/rfc4180

[RFC4627]

D. Crockford. IETF. *The application/json Media Type for JavaScript Object Notation (JSON)*. July 2006. Informational. URL: https://tools.ietf.org/html/rfc4627

[RFC7089]

H. Van de Sompel; M. Nelson; R. Sanderson. IETF. *HTTP Framework for Time-Based Access to Resource States -- Memento*. December 2013. Informational. URL: https://tools.ietf.org/html/rfc7089

[Richardson]

Richardson L.; Sam Ruby. O'Reilly. *RESTful Web Services*. 2007. URL: http://restfulwebapis.org/rws.html

[SCHEMA-ORG]

Schema.org. URL: http://schema.org/

[SDW-BP]

Jeremy Tandy; Payam Barnaghi; Linda van den Brink. W3C. *Spatial Data on the Web Best Practices*. 5 January 2017. W3C Note. URL: https://www.w3.org/TR/sdw-bp/

[SIRI]

CEN. *Service Interface for Real Time Information CEN/TS 15531 (prCEN/TS-OO278181)*. October 2006. URL: http://user47094.vs.easily.co.uk/siri/

[SKOS-DESIGN]

Tom Baker; Sean Bechhofer; Antoine Isaac; Alistair Miles; Guus Schreiber; Ed Summers. Elsevier. *Key Choices in the Design of Simple Knowledge Organization System (SKOS)*. May 2013. Journal of Web Semanics 20: 35-49. URL: http://dx.doi.org/10.1016/j.websem.2013.05.001

[SKOS-PRIMER]

Antoine Isaac; Ed Summers. W3C. *SKOS Simple Knowledge Organization System Primer*. 18 August 2009. W3C Note. URL: https://www.w3.org/TR/skos-primer/

[SchemaVer]

Alex Dean. *Introducing SchemaVer for semantic versioning of schemas*. 2014. URL: http://snowplowanalytics.com/blog/2014/05/13/introducing-schemaver-for-semantic-versioning-of-schemas/

[Tabular-Data-Primer]

Jeni Tennison. W3C. *CSV on the Web: A Primer*. 25 February 2016. W3C Note. URL: https://www.w3.org/TR/tabular-data-primer/

[Tabular-Metadata]

Jeni Tennison; Gregg Kellogg. W3C. *Metadata Vocabulary for Tabular Data*. 17 December 2015. W3C Recommendation. URL: https://www.w3.org/TR/tabular-metadata/

[Turtle]

Eric Prud'hommeaux; Gavin Carothers. W3C. *RDF 1.1 Turtle*. 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/turtle/

[URLs-in-data]

Jeni Tennison. W3C. *URLs in Data Primer*. 4 June 2013. W3C Working Draft. URL: https://www.w3.org/TR/urls-in-data/

[VOCAB-DCAT]

Fadi Maali; John Erickson. W3C. *Data Catalog Vocabulary (DCAT)*. 16 January 2014. W3C Recommendation. URL: https://www.w3.org/TR/vocab-dcat/

[VOCAB-DQV]

Riccardo Albertoni; Antoine Isaac. W3C. *Data on the Web Best Practices: Data Quality Vocabulary*. 15 December 2016. W3C Note. URL: https://www.w3.org/TR/vocab-dqv/

[VOCAB-DUV]

Bernadette Farias Loscio; Eric Stephan; Sumit Purohit. W3C. *Data on the Web Best Practices: Dataset Usage Vocabulary*. 15 December 2016. W3C Note. URL: https://www.w3.org/TR/vocab-duv/

[WEBARCH]

Ian Jacobs; Norman Walsh. W3C. *Architecture of the World Wide Web, Volume One*. 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

[XHTML-VOCAB]

XHTML 2 Working Group. W3C. *XHTML Vocabulary*. 27 October 2010. URL: https://www.w3.org/1999/xhtml/vocab