Primer

The Data Privacy Vocabulary [[DPV]] enables expressing machine-readable metadata about the use and processing of personal data based on legislative requirements such as the General Data Protection Regulation [[GDPR]]. This document acts as a 'Primer' for the DPV by introducing its fundamental concepts and providing examples of use-cases and applications. It is intended to be a starting point for those wishing to use the DPV and an orientation for people from all disciplines. The canonical URL for DPV is https://www.w3.org/ns/dpv which also contains its documentation and specification.

This primer document aims to ease adoption of DPV by providing:

- A high-level conceptual explanation of the DPV and its modelling of concepts;
- Self-contained examples that illustrate how the concepts and data models provided by DPV can represent information associated with personal data handling; and
- Guidance towards application of DPV in use-cases and technologies.

Related Documents

- This document only serves as the primer. Interested readers and adopters should reference the [[DPV]] Specification for a formal and normative description of DPV and its concepts.
- For a general overview of the Data Protection Vocabularies and Controls Community Group [[DPVCG]], its history, deliverables, and activities - refer to DPVCG Website.
- The peer-reviewed article "Creating A Vocabulary for Data Privacy" presents a historical overview of the DPVCG, and describes the methodology and structure of the DPV along with describing its creation. An open-access version can be accessed here, here, and here.

Contributing to DPV The DPVCG welcomes participation regarding the DPV, including suggestions for primer, expansion or refinement of DPV terms, addressing open issues, and other relevant matters.

While we welcome participation via any and all mediums - e.g., via GitHub pull requests or issues, emails, papers, or reports - the formal resolution of contributions takes place only through the DPVCG meeting calls and mailing

lists. We therefore suggest joining the group to participate in these discussions for formal approval.

For contributions to the DPV, please see the section on GitHub. The current list of open issues and their discussions to date can be found at DPVCG issue tracker as well as GitHub issues.

This document is a 'draft version' and is intended for feedback and comments.

Introduction

The [[[DPVCG]]] was formed in 2018 through [[[SPECIAL]]] with the ambition of providing a machine-readable and interoperable vocabulary for representing information about the use and processing of personal data, whilst inviting perspectives and contributions from a diverse set of stakeholders across computer science, IT, law, soiology, philosophy – representing academia, industry, policy-makers, and activists. It identified the following issues through the W3C Workshop on Privacy and Linked Data:

a. lack of standardised vocabularies to represent information about use and processing of personal data;
b. lack of descriptive taxonomies that describe purposes of processing personal data which are not restricted to a particular domain or use-case; and
c. lack of machine-readable representations of concepts that can be used for technical interoperability of information.

The outcome of addressing these resulted in the creation of the [[[DPV]]], which provides a vocabulary and ontology for expressing information related to processing of personal data, entities involved and their roles, details of technologies utilised, relation to laws and legal justifications permitting its use, and other relevant concepts based on privacy and data protection. It uses the EU's [[[GDPR]]] as a guiding document for the creation and interpretation of concepts, owing to its origin within [[[SPECIAL]]] and the involvement of stakeholders with interest in the GDPR.

People, organisations, laws, and use-cases have different perspectives and interpretations of concepts and requirements which cannot be modelled into a single coherent universal vocabulary. The aim of DPV is to act as a core framework of 'common concepts' that can be extended to represent specific laws, domains, or applications. This lets any two entities agree that a term, for example, `PersonalData`, refers to the same semantic concept, even though they might interpret or model it differently within their own use-cases.

The DPV is serialised using [[RDF]] (along with [[RDFS]] and [[OWL]]) to enable its use as a machine-readable and interoperable vocabulary. The intention of utilising RDF is to have a cohesive and standardised expression of the inherent semantics and relations between concepts (e.g. hierarchies). While DPV is a 'semantic web ontology', it is possible to use the DPV as a simple dictionary (i.e.

as a flat list) or other representations (such as JSON) based on the needs of an use-case.

Where can DPV be used?

DPV is primarily a vocabulary and an ontology for describing information about personal data and its processing. As such, it is intended to be utilised or applied as a vocabulary providing concepts for any application where information regarding processing of personal data is needed to be represented.

The following is an illustrative, but non-exhaustive list of applications possible with the DPV:

- document annotation - identifying and annotating concepts within documents such as privacy policies, legal compliance documentation, web pages
- records or logs – maintaining ex-ante (i.e. plans) or ex-post (i.e. logs) records of how personal data is processed, interactions and activities between entities (e.g. contracts and consent)
- policies – expressing policies for how personal data should be 'handled', policies for describing an use-cases' use of personal data
- rules expression – creating and utilising rules for expressing constraints or obligations regarding the use of personal data, checking conformance with obligations such as for legal compliance

For more concrete uses, see the community maintained [[[DPV-ADOPTION]]] in its wiki.

- Collect known applications, use-cases
  - SPECIAL partners
  - Signatu
- Specify additional ones
  - Represent privacy policies
  - Legal compliance evaluations
  - Provenance of organisation's personal data handling practices
- Link to page with testimonials/documented use-case from adopters

Core Concepts

Taxonomical Approach

DPV can be viewed as a hierarchical taxonomy of concepts where each `core concept` represents the top-most abstract concept in a tree and each of its children

provide a lesser abstract or more concrete concept. For example, consider the concept of `PersonalData` which is the abstract representation of *personal data*. It can be further *refined* or *extended* as `ContactInformation`, which itself can be further extended into concepts related to `Email`, `Telephone` and so on.

From this perspective, the top-most abstract concepts are collectively referred to as the *core vocabulary* within DPV. The goal of the DPV is to provide a rich collection of concepts for each of top concepts so as to enable their application within real-world use-cases. The identification of what constitutes a core concept is based on the need to represent information about it in a modular and independent form, such as that required for legal compliance.

Each core concept is intended to be independent from other core concepts. For example, the `Purpose` (e.g. Optimisation) refers only to the *purpose of why personal data is processed* and is independent as a concept from the `PersonalData` (e.g. `Location`) or the `Processing` activities (e.g. `collect`, `store`) involved to carry out that purpose. Such separation is necessary in order to represent and answer questions such as:

- *Q:* What data is being processed? *Ans:* CurrentLocation
- *Q:* What is being done with the data? *Ans:* collect, store
- *Q:* Why is the data being processed? *Ans:* Optimisation

The separation of concepts creates a *modular* structure for concept hierarchies within DPV, which in turn allows an adopter to use one particular concept taxonomy or module (e.g. list of purposes) independently without reusing the others, or to select only those concepts which are needed for their particular use-case. The separation also permits greater flexibility of representation and usage - such as using different combinations of core concepts as needed in use-cases. For example, a use-case can specify a single concept represent both Purpose and Processing by combining their respective concepts from DPV. The modular design of DPV also makes it possible to define domain and jurisdiction specific concepts in a separate namespace - such as the [[[DPV-NACE]]] purpose taxonomy providing a way for `Purpose` to indicate sectors using NACE taxonomy, and the [[[DPV-GDPR]]] for using `LegalBasis` to represent the legal bases provided by GDPR.

Overview of Core Concepts

`PersonalData`: 'Personal data' refers to any data about a natural person that can be used to identify them directly or, in combination with other information, indirectly. 'Personal data' is also referred to as 'personally identifiable information (PII)', a term that is mainly used in the United States of America. However the terms should not be interchangeably used as 'personal data', as it is defined in the GDPR, is a broader term than PII, as PII is usually used to refer only to information that can directly identify a person. DPV's definition of personal data is therefore based on the GDPR concept as it covers a wider range of information considered 'personal data'. Personal data can be declared as a category, such as

'Email', or an instance, such as 'x@y.z'.

`Purpose`: Representing the *purpose* for which personal data is processed, for e.g. 'Personalisation' as a broad category of purpose

`Processing`: representing *processing* as in the actions or operations over personal data, for e.g. collect, use, share, store.

`LegalBasis`: A legal basis is a law or a clause in a law that justifies or permits the processing of personal data in the specified manner. It is a jurisdictional concept given the scoping of laws to specified countries or regions, as well as a domain-specific concept given the specific laws enacted scoped to particular domains. A law, such as the GDPR, that regulates the use of personal data requires that every processing of personal data must be justified with some legal basis to ensure it is lawful, and to further assess its correctness, accountability, and impact based on the obligations applicable. However, what is considered a legal basis varies greatly across cultures, domains, use-cases, and laws themselves. The aim of DPV is therefore to provide an upper-level abstract taxonomy of categories of legal bases that can be customised and applied as needed.

`LegalEntity`: representing the 'entities' or 'actors' involved in the processing of personal data. DPV provides a broad categorisation of entities, with the following three representing the most common ones:

`DataController`: representing the organisation(s) responsible for processing the personal data

`Data Subject`: representing the categories or groups (e.g. Users of a Service), or instances (e.g. Jane Doe) of individual(s) whose personal data is being processed

`Recipient`: represents the entities that receive personal data, e.g. when it is shared

`Technical and Organisational Measure`: Technical and Organisational measures consist of activities, processes, or procedures used in connection with ensuring data protection, carrying out processing in a secure manner, and complying with legal obligations. Such measures are required by regulations depending on the context of processing involving personal data. For example, GDPR (Article 32) states implementing appropriate measures by taking into account the state of the art, the costs of implementation and the nature, scope, context and purposes of processing, as well as risks, rights and freedoms. Specific examples of measures in the article include:

- the pseudo-anonymisation and encryption of personal data
- the ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services
- the ability to restore the availability and access to personal data in a timely manner in the event of a physical or technical incident

- a process for regularly testing, assessing and evaluating the effectiveness of technical and organisational measures for ensuring the security of the processing

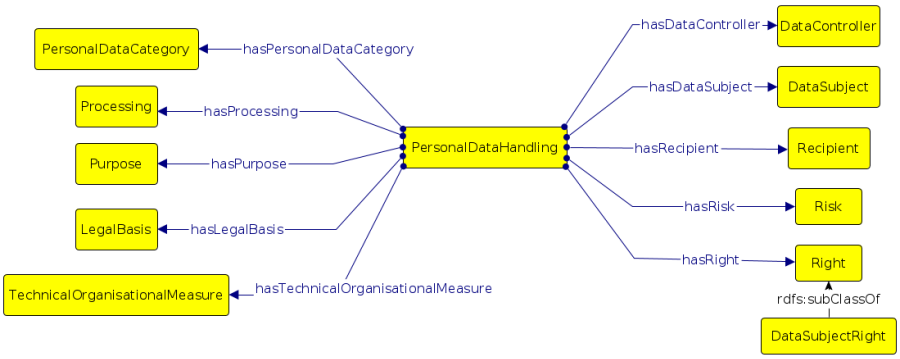
`Right`: representing the rights available, applicable, or afforded by a law or regulation, either to data subjects or data controllers, or other entities

`Risk`: representing risk(s) associated with a concept, for e.g. risk of unauthorised data disclosure related to processing, technical measure, or vulnerability of data subjects

Personal Data Handling

In legal terminology, it is common to refer to all information about how personal data is being processed using the colloquial term *processing*. This results in confusion between the use of *processing* as a concept referring to all information (i.e. purposes, personal data), and [=processing=] as a concept referring to the specific actions or operations (e.g. collect, use).

To avoid this, DPV defines a new concept called `PersonalDataHandling` for representing how the core concepts are related or apply to each other for a particular use-case. To connect or relate each core concept with an instance of `PersonalDataHandling`, relationships or properties are provided. These relationships are expressed using semantic web best practices so as to enable easier comprehension by humans, for e.g. naming the relationship `hasPurpose` to indicate the applicability of a `Purpose`. Generally, properties in DPV follow the convention of `has<SomeConcept>` to indicate applicability of `<SomeConcept>`. The core concepts and their relations are depicted in the following diagram.



For an example of how `PersonalDataHandling` brings together the core concepts, the following diagram and RDF code block indicates `Acme` is a `DataController` that uses `Email` personal data for `Marketing` by performing `Collect` and `Use` processing actions.

```
ex:AcmeMarketing a dpv:PersonalDataHandling ;
  dpv:hasPersonalDataCategory dpv:EmailAddress ;
```

```
dpv:hasProcessing dpv:Collect, dpv:Use ;
dpv:hasPurpose dpv:Marketing ;
dpv:hasDataController ex:Acme .
```

The use of `PersonalDataHandling` can be nested, which means one instance can contain other instances, much like a box with several smaller boxes inside. This permits breaking down complex or dense use-cases into more granular ones and representing them in a more precise and granular fashion. For example, in the above example, consider the following figure outlining its separation into two distinct use-cases where, (i) data is stored using a data processor, (ii) data is used for Marketing. While it is certainly possible to represent all of this information within one single instance of `PersonalDataHandling`, the adopter may decide to create separate instances of `PersonalDataHandling` based on requirements such as reflecting similar separations for legal documentation or accountability purposes.

Note that `PersonalDataHandling` is intended to provide a convenient concept for tying the core concepts together, and DPV does not make its use binding, nor does it constrain the relationships to only be defined between `PersonalDataHandling` and the other core concepts. This is so as to permit using DPV in alternate or differing models. For example, where a central concept already exists, such as when describing relevant information for a smartphone app, the concept for `App` can be a replacement for `PersonalDataHandling` based on statements such as `<App hasPurpose SomePurpose>`. Even in such cases, `PersonalDataHandling` can provide granular expression thereby enabling description of different contexts within which the app uses personal data, such as for registration or complaint resolution.

Another instance where one may not wish to utilise `PersonalDataHandling` is where the adopter or use-case wants to indicate a different method for relating concepts together. For example, instead of expressing the relationship between personal data and purpose through a `PersonalDataHandling` instance, an alternate model could be one where the purpose directly specifies what personal data it uses as: `<SomePurpose hasPersonalData SomePersonalData>`. Similarly, another instance for such alternate use of concepts is to associate a legal basis directly with the purpose by using the `hasLegalBasis` relationship. To support such uses, DPV does not explicitly declare restrictions on the properties in terms of what concepts they can be used with (e.g it does not provide domain assertions). In case an adopter needs such explicit declarations, they can utilise or import the separate file declaring them. The following figures indicate some commonly encountered alternate models which do not use `PersonalDataHandling` as a central concept.

Create a separate file providing property domains and ranges, and for classes declare disjoint for relevant and known disjoint concepts.

Some alternatives, such as using `Purpose` instead of `PersonalDataHandling`, or creating ad-hoc concepts such as `App` or `Policy` as per use-case.

When using alternate models, it is important to note the consequences such models can have on interpretations and interoperability of data defined using DPV. For example, consider a compliance assessment tool that takes data defined using DPV as input. If the tool only operates over `PersonalDataHandling` to perform its assessments, using other alternate models and relationships can produce invalid or incorrect results. To avoid this, we recommend:

1. Documenting alternate models as deviations from the DPV specification to indicate its difference from DPV in terms of semantics and interpretations;
2. Where possible, ensuring and providing mappings between the alternate models and the DPV's specification, so that data can be transformed for interoperability;
3. Consider contributing the alternate model to DPVCG to create a 'library of alternate models' to enable better understanding of adopters needs and influencing future developments within DPVCG.

Semantics of DPV

DPV uses the semantics provided by [[RDF]] (along with [[RDFS]] and [[OWL]]) to define *concepts*. The following section provides a brief overview of how concepts and relationships within DPV should be interpreted.

Classes, Hierarchies, and Instances

A 'class' is a set of information associated with a particular concept, represented by the class label or description. For example, the class `Email` refers to information about *emails*. This information may contain *email addresses, aliases, signatures*, and so on. While an intuitive use of `Email` may be taken to only refer to *email address*, within DPV the concepts are structured as a hierarchy of concepts where each *parent concept* represents a broad set of information and its *children concepts* represent parts of that set. For `Email`, this is represented in the following diagram where `Email` is a larger set and contains several smaller sets for different categories of information within the broad label of *email*.

The relationship between a child concept and its parent concept is defined using the RDFS semantic property `isSubClassOf` to express *subclass* or *subset* relationship. For example, the statement `<EmailID rdfs:subClassOf Email>` indicates that the `EmailID` is a subset of `Email`. This leads to the interpretation where indicating the broader set's (e.g. `Email`) involvement implies its subsets (e.g. `EmailID`) are also involved. Conversely, a smaller set's involvement does not imply the larger set's involvement (i.e. specifying `EmailID` is involved does not entail all of `Email` to be involved). Therefore, when using DPV, it is strongly advised to use the appropriate subclass rather than the broader classes so as to be specific about the involvement of concepts and their scope.

A *class* is a set representing a collection of information represented by a concept. An *instance* is a member of that set representing a specific *instance* of that

concept. For example, `EmailID` is a class of all email identifiers (or addresses), and `x@y.z` is an instance of an email identifier. In this case, `x@y.z` is also an instance or member of the class/set `Email` based on the relationship between `EmailID` and `Email`.

In the real-world, *email* is used as a *catch-all term* or *broad concept* for referring to any and all of: email address, email contents, email exchange timelines and specifics, or even all 'emails'. Concepts in DPV are intended to support such broad usage of terms by modelling them as abstract classes or sets, within which each attribute or contents can be granularly further defined.

To better reflect a use-case and to ensure accuracy of interpretation, it is strongly recommended to either choose a more detailed concept from the DPV taxonomies, and/or to find a suitable concept and extend it to reflect the specifics as accurately as possible.

Extending Concepts for Use-Case

Most of the concepts within DPV are provided as hierarchies of classes representing categories of information, which are generic or abstract or broad so as to permit their application across a diverse and varied landscape of real-world use-cases. In order to accurately reflect the particulars of an use-case, concepts within DPV should (most likely) be extended. The specifics for how this should be done depend on the manner in which DPV is utilised. For example, if using [[RDFS]] or [[OWL]] semantics, extending concepts is straightforward using the subclass relationship or by creating instances of a concept. In these, a *subclass* represents a subset or category which can contain several instances, whereas an instance is an exact and specific instantiation of a concept. For example, `WorkEmail` is a category of work related emails and `office@example.com` is an instance of a work email.

Where an exact concept is not present within the DPV and a broad concept exists for representing the same information, one should subclass or extend that broad concept to define the required information. For example, DPV defines the (broad) concept `Marketing` in its `Purpose` hierarchy to represent information about (purposes related to) marketing activities and topics. For a use-case which requires representing purposes (note: plural) related to *marketing of new products*, the broad `Marketing` concept is extended as a *child* or *subclass* concept for representing the intended purpose as, e.g. `MarketingNewProducts`. Using RDFS, this is expressed as follows:

```
# Case1: Where further categories are required to 'group' related purposes
# creating a new subclass or category of Marketing for use-case
ex:MarketingOfNewProducts rdfs:subClassOf dpv:Marketing ;
    rdfs:label "Marketing of New Products" .

# instances of purposes under purpose group 'Marketing of New Products'
ex:NewslettersOffers a ex:MarketingOfNewProducts ;
```

```
    rdfs:label "Newsletters about Offers" .
ex:EmailsSeasonalOffers a ex:MarketingOfNewProducts ;
    rdfs:label "Emails about Seasonal Offers" .

# Case2: Where only a single purpose needs to be represented without any grouping
ex:MarketingSeasonalOffer a dpv:Marketing ;
    rdfs:label "Sending Email Newsletters with Seasonal Offers" .
```

The mechanism for extending concepts (via both subclasses and instances) is useful to align existing concepts or vocabularies with the DPV taxonomies, such as by declaring them as subclasses of a particular concept. This permits the creation of domain or jurisdiction specific *extensions*, such as [[DPV-GDPR]] for expressing the legal bases provided by GDPR.

DPV is a top-down hierarchy, which means that the concepts at the *top* are broader and more abstract, and as one goes down the *tree* they get more narrow and specific.

Extensions also permit more accurate representations of a use-case by extending from multiple concepts to refine and scope the interpretation. This means each concept can have multiple parents representing the intersection of their respective sets. For example, in the following example use-case, an activity simultaneously uses the data while collecting it. The first representation (`ActivityA`) models them separately - which is not accurate as it is ambiguous in terms of whether collection and usage happen independently or in what order. Instead, by extending the concepts to create a new concept `CollectAndUse`, the use-case can accurately declare that they both occur are to be considered as a single concept along with additional information, such as: technologies involved, implementation details, or agents involved.

```
# Method 1: Ambiguous regarding independence of Collect and Use
ex:ActivityA a dpv:PersonalDataHandling ;
    dpv:hasProcessing ex:Collect, ex:Use .

# Method 2: Accuracy regarding combination of Collect and Use
ex:CollectAndUse a ex:Collect, ex:Use ;
    rdfs:label "Collect and Use data using User Device" .
ex:Activity a dpv:PersonalDataHandling ;
    dpv:hasProcessing ex:CollectAndUse .
```

It is not necessary to extend concepts unless one wishes to depict use-case specific information. For example, if in a use-case it is sufficient to (only) say some information is collected, then `dpv:Collect` can be directly used. However, where more specific information is needed, such as also specifying a method of collection (e.g. `CollectViaWebForm`), then it is recommended to extend the concept, for example as `<CollectViaWebForm a dpv:Collect>`. If there are lots of forms and they need to be 'grouped' together as collection methods, then one would

subclass `Collect` as `CollectViaWebForm` and create instances of it for each form to be represented.

Though this example used a web form as a method of collection by directly mentioning it within the concept as `CollectViaWebForm`, this may not always be desirable. For example, that same web form may also need to be represented separately for logging purposes. DPV is exploring the provision of a `Technology` concept to assist in representing information regarding how concepts are implemented and the use of specific technological artefacts such as web forms, databases, along with their functions such as data storage and retrieval.

Interoperability of Extended Concepts

DPV intends to provide a base or foundational framework for different entities to exchange information and interpret concepts for interoperability. When an adopter (e.g. an organisation using DPV) extends concepts to refine them for their own use-case, the concept is still (weakly) interoperable by relying on DPV's broad taxonomies to provide a common point of reference. For example, two TV companies (`AliceCo` and `BobCo`) extend the concept `Optimisation` to reflect their respective purposes as follows:

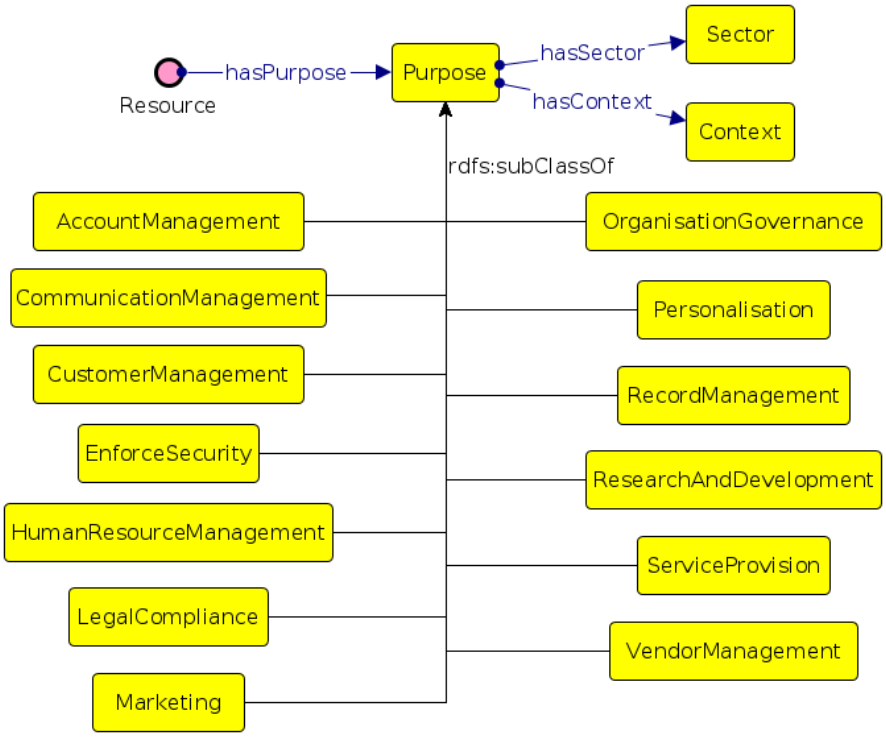
```
# AliceCo's optimisation related to better services for users' infrastructure
ex:TVServiceOptimision a dpv:OptimisationForConsumer ;
    rdfs:label "Optimise Service for Users' Infrastructure" .

# BobCo's optimisation related to more efficient signals for users' TV sets
ex:TVSignalOptimisation a dpv:OptimisationForConsumer ;
    rdfs:label "Optimise Signal for Consumer TV Set" .
```

When exchanging information about their use-cases with each other (or with a third party), by following the chain of use-case specific concepts it is possible to deduce that both `AliceCo` and `BobCo` are doing optimisations for consumers. Thus a common language or interface can be developed based on using DPV as a point of interoperability and commonality which can be used by adopters to define the specifics of their use-case. For example, in the above use-case, a common notice generation algorithm could be created and used to inform users of both services the purposes each company is using data for.

Taxonomies of Key Concepts

The following sections provide an overview of the taxonomies (i.e. hierarchies of concepts) provided by DPV for its core concepts.

Purpose



DPV's taxonomy of purposes is used to represent the *reason* or *justification* for processing of personal data. For this, purposes are organised within DPV based on how they relate to the processing of personal data in terms of several factors, such as: management functions related to information (e.g. records, account, finance), fulfilment of objectives (e.g. delivery of goods), providing goods and services (e.g. service provision), intended benefits (e.g. optimisations for service provider or consumer), and legal compliance.

It is important to note the following in terms of how purposes are discussed within the real-world:

1. There is no universal definition for what constitutes a 'purpose' or what attributes are associated with it.
2. There are several distinct ways to model purposes, e.g. as a 'goal' such as 'Delivery of Ordered Goods'; or as a statement explaining the processing of personal data, e.g. 'Sending newsletters to Email'.
3. The requirements for a 'valid purpose' are defined externally, e.g. [[GDPR]] Article.5-1b requires purposes to be 'explicit and legitimate' based on specific conditions which should be met

4. Purposes have contextual interpretations within their application (i.e. where they are used in an use-case), e.g. purposes related to marketing are quite distinct from purposes related to health and safety even if they use the same wording.

Following from the above, most use-cases would need to extend one of the concepts within DPV's purpose taxonomy to ensure its purpose descriptions are specific and understandable to the context of that use-case. We therefore suggest to adopters, where possible and appropriate, to create their own purposes as required within their use-cases by either declaring the purpose being used as an instance or subclass of one or several purpose categories within DPV and to provide a human readable description to assist in its accurate interpretation (e.g. for RDF, using `rdfs:label` and `rdfs:comment`).

In this example, a new purpose is created as an instance of `dpv:DeliveryOfGoods` and `FraudPreventionAndDetection`, and is accompanied with human-readable information. The interpretation of this purpose is to be taken as being the intersection or combination of the two purpose categories, i.e. the purpose is to achieve both the delivery of goods as well as fraud prevention and detection.

```
ex:CombinedNewPurpose a dpv:DeliveryOfGoods, dpv:FraudPreventionAndDetection ;
  rdfs:label "Deliver Goods and Fraud Prevention" ;
  rdfs:comment "Intended delivery of goods with fraud prevention" .
```

Sector of Purpose Application To further clarify how purposes should be interpreted, DPV provides several concepts such as `Sector` and `Context`. `Sector`, used with the `hasSector` property, can denote *sector* or *domain* of operation, such as Manufacturing. This can be used alongside existing official sector taxonomies such as [[NACE]] (EU), [[NAICS]] (USA), or [[ISIC]] (UN), as well as commercial industry taxonomies such as [[GICS]] maintained by organisations MSCI and S&P. Multiple classifications can be used through mappings between sector codes such as the [[[NACE-NAICS]]] provided by EU.

DPVCG provides an interpretation of the NACE revision 2 codes which uses `rdfs:subClassOf` to specify the hierarchy between sector concepts. It is available as [[DPV-NACE]]. The NACE codes within this extension have the namespace `dpv-nace` and are represented as `dpv-nace:NACE-CODE`.

We are working on further alignments between the NACE codes and DPV's purpose taxonomy, and welcome contributions for the same.

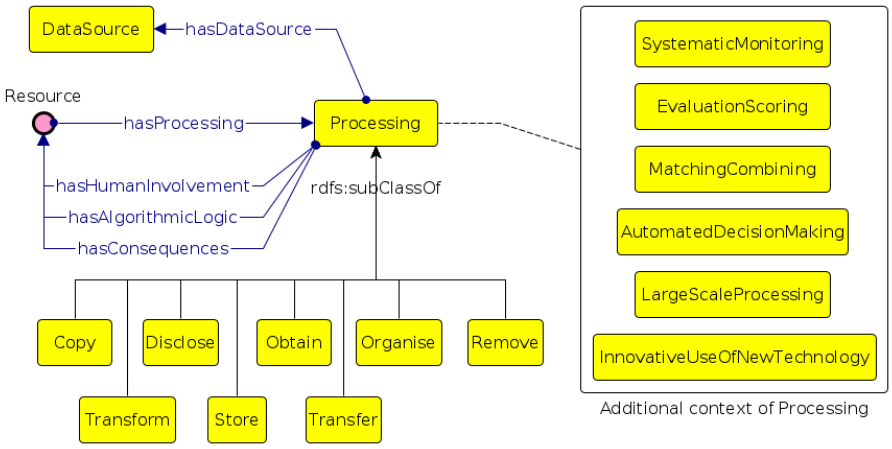
For example, the following purpose concerns implementing access control with the domain specified as scientific research using its corresponding NACE code `M72`:

```
:LimitAccess a dpv:AccessControl ;
  rdfs:label "Limit access to lab" ;
  dpv:hasSector dpv-nace:M72 .
```

Context of Purpose Application While the use of sector codes for restricting (personal data processing) purposes is an uncommon and undocumented practice in terms of legal enforcement, we provide this feature as the use of sector code can assist with identification and interpretation of information as well as legal or organisational obligations and policies. For example, indicating some purpose is to be implemented within manufacturing or scientific research facilities (e.g. medical centres) can assist in ensuring specific types of access control and policies are defined and implemented.

For association additional information for how the purposes can be further restricted or interpreted, the `Context` concept along with the `hasContext` property can be used. `Context` refers to a generic *context* of how that purpose is applied and is intended to be expanded as required and applicable within an use-case or domain. Context enables associating information not possible to be represented or otherwise associated using the other concepts available within DPV. For example, indicating who developed that purpose or documentation about its application.

Processing Categories



DPV's taxonomy of processing concepts reflects the variety of terms used to denote processing activities or operations involving personal data, such as those from [[GDPR]] Article.4-2 definition of *processing*. Real-world use of terms associated with processing rarely uses this same wording or terms, except in cases of specific domains and in legal documentation. On the other hand, common terms associated with processing are generally restricted to: collect, use, store, share, and delete.

DPV provides a taxonomy that aligns both the legal terminologies such as those defined by GDPR with those commonly used. For this, concepts are organised based on whether they subsume other concepts, e.g. `Use` is a broad concept indicating data is used, which DPV extends to define specific processing concepts for `Analyse`, `Consult`, `Profiling`, and `Retrieving`. Through this mechanism, whenever an use-case indicates it *consults* some data, it can be inferred that it also *uses* that data.

The definitions for describing and interpreting each processing concept is based on the following sources: language dictionaries (pre-dominantly Oxford English), use of the term within legal documents (e.g. GDPR case law), and technology-specific interpretations such as for IT systems. Despite these, there may be distinct interpretations for what a term represents based on differences in practices, culture, language, and domains. In case an adopter or a use-case foresees such ambiguity or confusion, it is advisable to extend the relevant concepts and define them as needed, or create a separate extension.

Indicating Processing Location, and Duration For indicating details of processing, such as where and when it is taking place in terms of location and duration, DPV provides the properties `hasLocation` and `hasDuration`. While DPV also provides the concepts `ProcessingLocation` and `ProcessingDuration` for use with these properties, their use is not obligatory as an use-case may want to follow its own conventions, such as use of plain-text strings, e.g. `"EU"` to indicate location, or the use of external vocabularies, e.g. [[[TIME]]] to denote duration. Where possible, it is recommended to utilise common vocabularies and methods of specifications to promote interoperability of information. In the earlier example, the `"EU"` string could be replaced with a vocabulary providing jurisdictional concepts for representing the EU.

Indicating Storage and Source of Data The DPV processing taxonomy provides the concept `Store` to indicate data is being stored. To specify where the data is being stored, the property `hasLocation` can be used alongside `Store` class. Similarly, the `hasDuration` property can be used to denote duration of storage of data. In addition to these, the `StorageDeletion` class complements the duration declaration by explicitly referring to a deletion policy, such as for deletion guarantees.

For declaring the source of data, the `DataSource` class along with `hasDataSource` property can be used. This enables declaring where the data is collected or acquired from. For example, data can be obtained from the data subject directly (e.g. given via forms) or indirectly (e.g observed from activity, or inferred from existing data), or from another entity such as a third party.

Indicating Impact based on Scale, Automation, Consequences For indicating attributes which affect how processing should be considered in terms of impact or carry additional legal obligations, DPV provides concepts based on

[[GDPR]] Article.35 regarding Data Protection Impact Assessments (DPIA) for representing whether processing is carried out at large scale, consists of systematic monitoring (of data subjects), is performed for evaluation or scoring (of data subjects), matching and/or combining existing datasets, utilizes automated decision making, or involves innovative use of new technologies. These concepts provide a way to indicate their applicability for specific processing activities, such as specifying collection happens at large scale or that data is used for/with automated decision making.

For indicating consequences of a processing, the property hasConsequences can be used. Consequences could be non-material or material, and may or may not be impactful to the Data Subjects. While the consequence concept is described here in the processing section, an use-case may choose to utilize it in concepts other than processing, such as for purpose or processing.

Indicating Algorithmic Logic, and Human Involvement For indicating the algorithmic logic or human involvement for a specific processing operation , the properties hasAlgorithmicLogic and hasHumanInvolvement are available respectively. Using these, it is possible to describe the algorithms or point to an external resource or document providing more details, such as how a human moderator was involved in a particular step.

Personal Data

DPV's provides the concept `PersonalData` and the property `hasPersonalData` to indicate what categories or instances of personal data are being processed. As described earlier, common use of personal data concepts in the real-world consists of specifying as concepts both categories (e.g. *Location*) and instances (e.g. your *exact location right now*). DPV's concepts represent categories of personal data, which are expected to be further extended to reflect the actual categories or instances required in an use-case.

Real-world and common usage of *personal data* is at both an abstract level as well as specific level. For example, consider the sentence "We use your Email information...", which uses "*Email*" to represent a reference to what personal data is involved. Here, one may interpret *Email* as representing only the *email address*, or as a broad set of possible information related to emails, such as *email address, email senders and recipients list, email service provider, email usage statistics* and so on.

For ensuring clarity and resolving any potential ambiguity, DPV recommends being as specific as possible. This means where there is ambiguity as to what the information may be associated with or within a concept, it is advisable to resolve that ambiguity - either by choosing a more accurate concept from the taxonomy and/or by creating one through extension of an existing concept.

Prior versions of DPV (i.e. from v0.1 to v0.3) contained a taxonomy of personal data categories (provided under term `PersonalDataCategory`), which were

removed to avoid ambiguity and confusion between concepts which could be considered as personal data and also as other concepts associated with its processing. For example, `Location` was defined as a personal data category, but was also relevant when describing other location-based concepts such as `StorageLocation`.

The taxonomy of personal data categories is now provided as [[[DPV-PD]]], with DPV retaining only a few important concepts necessary for providing the abstract concepts related to personal data. An adopter can choose to use the extension providing categories of personal data, or use another external vocabulary, or define their own categories. In addition to removal of the taxonomy, the class `PersonalDataCategory` and its associated property were changed to `PersonalData` for reflecting the necessity to represent both categories and instances of personal data using DPV.

For indicating personal data which is sensitive, the class `SensitivePersonalData` is provided. For indicating special categories of data, the class `SpecialCategoryPersonalData` is provided. In this, the concept *sensitive* indicates that the data needs additional considerations (and perhaps caution) when processing, such as by increasing its security, reducing usage, or performing impact assessments. *Special categories* are a subset of sensitive personal data which require additional or separate legal justifications in order to process them. For example, [[GDPR]] Article 9 defines several special categories of personal data which are prohibited from being processed unless an appropriate legal basis from Article 9 is (also) used.

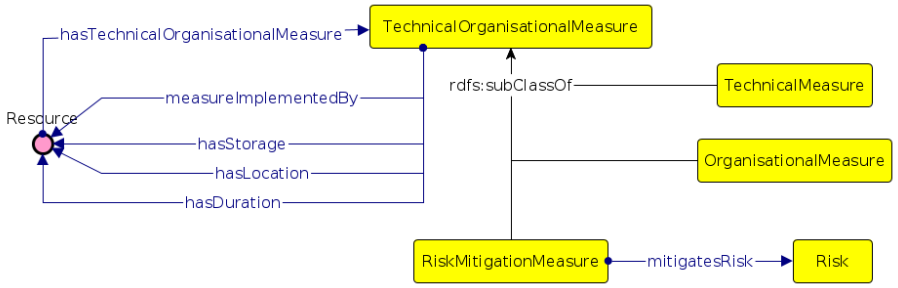
For indicating a category is sensitive or special, it is sub-classed from the appropriate class within DPV. While work is ongoing within DPV to identify and provide a reference list of sensitive and special categories, a use-case can indicate some personal data categories to be sensitive or special by extending the concept and sub-classing it.

For indicating personal data is derived or inferred from other personal data, the concepts `DerivedPersonalData` and `InferredPersonalData` are provided. These allow expressing which personal data has been derived or inferred within a use-case. In DPV, the method for derivation or inference is considered to be part of the processing concept and therefore should be declared using the appropriate processing concepts rather. However, an adopter can choose an alternate model for declaring information about how data was derived or inferred, such as by using PROV to model data as inputs or outputs of activities.

It is challenging to accurately represent involvement of personal data due to the extreme variance in intended interpretations of how concepts function within real-world use. For example, the DPVCG debated the modelling of personal data categories based on the scenario where a picture of passport is initially collected, and from it various information is extracted, such as name, address, and photo. For representing this, merely stating the personal data category as 'passport photograph' might be misleading as there is additional information within that photograph.

A compromise was established whereby the use-case is expected to declare what information it intends to collect or use through the mechanism of extensions or subclasses. For the passport photograph scenario, the use-case would declare the class `PassportPhoto` with subclasses representing `Name`, `Age`, and so on. However, one may wish to use another method for explicitly declaring information, such as by creating additional properties (e.g. `containsData`) or reusing existing ones (e.g. `skos:narrower`) to indicate the photo *contains* information related to name, age, etc. We welcome contributions and proposals regarding both these methods.

Technical and Organisational Measures



DPV provides a taxonomy of technical and organisational measures for representing information about how the processing of personal data is technically and organisationally protected, safeguarded, or otherwise implemented. This is distinct from what technology is used for carrying out processing, and instead refers to what *measures* are in place (i.e. what the technology intends to provide in terms of features). Some examples of common technical measures are use of encryption and access control for security, and for organisational measures include staff training, having guiding policies, and certifications.

DPV provides a taxonomy of concepts which can be structured into two groups of technical and organisational measures respectively. The technical measures, represented by the concept `TechnicalMeasure`, concern measures primarily achieved using some technology. By contrast the organisational measures, represented by OrganisationalMeasure

, are carried out through activities and processes at the management and organisational levels, which may or may not be assisted by technology.

Similar to the other taxonomies, it is advised for an adopter to extend the relevant concept through subclass or instance mechanisms to represent the measure relevant to the use-case. For example, to indicate data is encrypted using the Rivest-Shamir-Adleman (RSA) method, one would extend the `Encryption` concept within DPV to represent `RSA`, and then instantiate it with the specific implementation used (e.g. to indicate key size).

Implementation of Measure To indicate how a measure is implemented, the property `measureImplementedBy` can be used. This permits associating a particular technology, documentation, or entity with the measure to provide information on how it is used, applied, implemented, or deployed within the use-case. The property can be used for both technical as well as organisational measures to indicate the implementation as a document outlining procedures, guidelines, or a software product.

The DPV intends to represent commonly utilised technologies (e.g. database) and their use in processes (e.g. querying) for use as implementation of technical and organisational measures. We welcome contributions for the same.

Restrictions for Storage, Location, and Duration The technical and organisational measures can be annotated with information in terms of their storage, location, and duration using the properties `hasStorage`, `hasLocation`, and `hasDuration`. This is similar to how processing concepts can utilise these concepts to indicate, for e.g. data is collected from a particular location.

In addition to these, additional restrictions for storage, represented under `StorageRestriction`, can be used to define information about duration (as `StorageDuration`), location (as `StorageLocation`), and deletion (as `StorageDeletion`). These classes complement the properties for storage, duration, and location.

Using Existing Standards Technical and organisational measures are the subject of standardisation efforts at national as well as international levels through organisations such as ISO, NIST, CEN/CENELC and ENISA. Such standards can be expressed within DPV by extending the appropriate concepts and describing the application of that standard within the use-case. For example, for indicating the access control measure follows the guidelines of the [[ISO-27017]] standard for security implemented by cloud service providers, the instance of `AccessControlMethod` used in the use-case can be annotated with a reference to the standard by using `measureImplementedBy`.

The DPV intends to represent commonly utilised standards (e.g. ISO) and specifications (e.g. W3C) for use as technical and organisational measures. We welcome contributions for the same.

Risk and Risk Mitigation For risk management, DPV provides the class `Risk` and property `hasRisk` to indicate applicability or existence of a risk, and the class `RiskMitigationMeasure` and property `mitigatesRisk` to indicate mitigation. *Risk*, as a concept, can be associated with any other concept given its broad existence and applicability, whereas its mitigation is defined as a technical and organisational measure. Through this, the implemented or adopted technical and organisational measures can be annotated with the risks they address or mitigate. For example, the implementation of access control methods mitigates the risks of unauthorised access. While these concepts provide a rudimentary

level of association between DPV concepts and risk management, it is important to note that the DPV is not a risk management vocabulary as it does not model several of the concepts necessary for risk assessment and documentation.

Legal Basis

DPV provides the following categories of legal bases based on [[GDPR]] Article 6: consent of the data subject, contract, compliance with legal obligation, protecting vital interests of individuals, legitimate interests, public interest, and official authorities. Though derived from GDPR, these concepts are represented as an abstract legal basis and can be applied for other jurisdictions and use-cases. The legal bases are represented by the concept `LegalBasis` and associated using the property `hasLegalBasis`.

When declaring a legal basis, it is important to denote under what law or jurisdiction that legal basis applies. For instance, using `Consent` as a legal basis has different obligations and requirements in EU (i.e. [[GDPR]]) as compared to other jurisdictions. Therefore, DPV recommends indicating the specific law or legal clause associated with the legal basis so as to scope its interpretation. For GDPR, DPVCG provides the [[[DPV-GDPR]]] which defines the legal bases within [[GDPR]] by extending them from relevant concepts within the DPV. We welcome similar contributions for extending the GDPR extension as well as creating extensions for other laws and domains.

Legal Bases for Cross-Border Data Transfers Given the importance of data transfers or movement across jurisdictional borders, DPV also defines the concept `DataTransferLegalBasis` to represent the category of legal bases applicable for justifying such cross-border data flows. For GDPR, these involve Articles 45, 46, and 49, and are defined within the [[DPV-GDPR]] extension. When representing such cross-border data flows, concepts such as jurisdiction and countries are also required so as to indicate which laws apply, and whether political treaties exist between regions. We welcome contributions for expressing this information within the framework of the DPV.

Consent

`Consent` is a specific (and perhaps special) legal basis given its emphasis on individual empowerment and control, as well as the attention and relevance it receives from being in front of the individuals via ubiquitous consent dialogues throughout the web. DPV provides concepts for representing information about how consent, as a legal basis, was collected (by the Controller) or given (by the Data Subject), how long it is considered to be valid (its duration), and how it can be withdrawn. This information can be utilised in applications associated with consent, such as creating a 'record' of consent for compliance.

Given the reliance of consent as a legal bases whose validity is associated with requirements and obligations based on jurisdictional laws, DPV does not dictate

what constitutes a 'valid' consent and only provides a way to declare information about it. Such information concerning compliance obligations and requirements is within the scope of the DPVCG, and we welcome contributions on how this can be represented in a coherent manner within an extension that is compatible with the rest of DPV.

The class `Consent` represents the concept of *consent*, with the DPV classes and properties used to associate the relevant information it is about. If using `PersonalDataHandling`, the use of consent can be indicated through the `hasLegalBasis` property, and the relevant information associated using properties, such as `hasPurpose` and `Purpose`. Conversely, one can also create alternate forms of *consent as a policy* where the purpose and other information is directly associated with Consent using the generic properties.

To specify additional information, DPV specifies provenance properties of `hasProvisionTime` and `hasProvisionMethod` for how consent was given, and `hasWithdrawalTime` and `hasWithdrawalMethod` to indicate how consent was withdrawn. The expiry of consent can be specified using the property `hasExpiry`, with sub-properties provided to define expiry as a temporal entity using `hasExpiryTime` or as a condition/event using `hasExpiryCondition`.

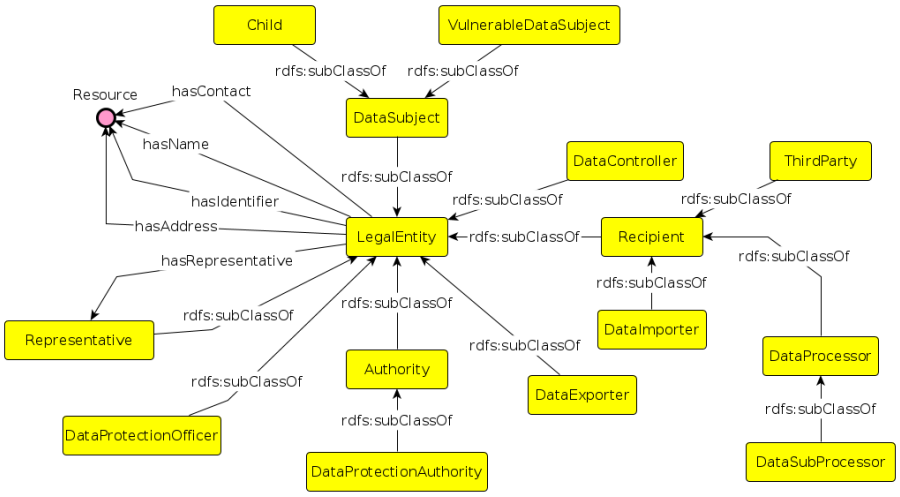
Example

Requirements for *informed* consent require provision of information before the consent is obtained so as to inform the individual. This information is typically provided through a *notice*, which can be specified using the property `hasConsentNotice`. Similarly, *explicit* consent is a specific form of consent where the action of giving consent carries additional obligations of being explicitly performed by the individual. To represent this, the boolean property `isExplicit` is provided.

The specific conditions under which consent can be considered valid, or informed, or explicit as defined under jurisdictional law. The terms provided within the DPV are generic concepts for representing these *types* of consent. For using specific jurisdictional requirements and use of consent as a legal basis, it is advisable to declare such jurisdictional concepts in a separate extension to the DPV. For specific use of consent as a legal basis defined within GDPR, we provide [[DPV-GDPR]].

To specify consent provided by delegation, such as in the case of a parent or guardian providing consent for a child, the properties `hasProvisionByJustification` and `hasWithdrawalByJustification` can be used to capture the nature of such 'delegation', with the fields `hasProvisionBy` and `hasWithdrawalBy` representing the legal entity who provided the consent. By default, the consent can be assumed to be provided by the associated Data Subject.

Legal Entities



A 'legal entity' is an entity whose role is defined legally or within legal norms. DPV provides a taxonomy of entities based on their application within laws and use-cases. Alongside `DataController` and `DataSubject` entities, the DPV also defines other types of entities involved in the processing of personal data. The class of entities that are recipients of personal data is represented by the class `Recipient` and property `hasRecipient`. The `DataProcessor` is a recipient which is contracted by the Data Controller to carry out processing, and the `DataSubProcessor` is a Data Processor which is contracted by (another) Data Processor. A `ThirdParty` is an entity other than the data subject, data controller, and data processor that is a recipient of personal data.

To represent a data subject is a minor or a child, the concept `Child` is defined. Note that the definition of age for what constitutes a minor or a child is based on jurisdiction. To represent data subjects that are considered a vulnerable group, the concept `VulnerableDataSubject` is provided. This permits indicating that a group of data subjects (e.g. minorities) are considered vulnerable in a given use-case.

To indicate the authorities, the concept `Authority` and its subclass `DataProtectionAuthority` are defined. To indicate representatives, the class `Representative` and property `hasRepresentative` are provided. These permit associating a representative, such as a person or a law firm, with another entity. A `DataProtectionOfficer` class is defined as a specific category of representatives responsible for carrying out data protection functions within the organisation.

For indicating how organisations act within the context of data transfer, especially regarding cross-border data flows, the concepts `DataImporter` and `DataExporter` are defined. These are based on the guidelines issued by the

EDPB regarding GDPR's application for cross-border data flows, where a data exporter is an entity that 'exports' or transfers data outside the jurisdiction, whereas a data importer is an entity that 'imports' or receives a data transfer from outside the jurisdiction. Note that the exporter and importer roles are independent from those of controllers and processors, i.e. a controller or a processor can be an exporter or an importer.

Rights

For representing the rights available or applicable in the use-case, the class `Right` is provided. Note that rights are not exclusive to the data subjects, but can also be available to the other entities, such as data controllers and processors. For specifically referring to data subject rights, the class `DataSubjectRight` is defined. Rights can be associated with relevant concepts using the property `hasRight`. Since rights are defined by laws which are bound by the jurisdictions within which they operate, the DPV does not provide a taxonomy of specific rights. The DPV-GDPR, representing the application of DPV concepts for GDPR, specifies the rights provided within GDPR by extending the `DataSubjectRight` concept.

To indicate a right is applicable within a particular use-case, one can use the property `hasRight` with the `PersonalDataHandling` instance. Or for cases where a right is needed to be associated with a particular personal data category or a processing operation, it can be similarly associated with that concept.

Using DPV

The DPV is serialised using [[RDF]] and uses [[RDFS]] and [[OWL]] semantics to define the hierarchy and relationships between its concepts. While this enables its use as a semantic web ontology, the DPV can also be used without semantic web by either utilising a serialisation such as [[JSON-LD]] which retains the semantics but provides convenience of using known notations, or using other methods which strip the semantic information out, such as a CSV or a flat-list of concepts. This section provides an overview of such approaches where DPV can be used both with and without semantic web.

RDFS, OWL, SKOS

DPV, as it is serialised by default, uses both [[RDFS]] and [[OWL]] semantics. An adopter's use-case may be using only one of these, and they may wish to retain this structure. Similarly, another adopter may prefer the semantics of [[SKOS]] which has a simpler mechanism for utilising the concepts as a vocabulary. For such use-cases, DPV provides 'variants' or 'flavours' for semantics associated with [[RDFS]], [[OWL]], and [[SKOS]]. These are available here.

In addition to these, another common requirement for use-cases is to have domain and range assertions, which are absent from the (default) DPV to permit a

wider applicability of its concepts. For use-cases that require such assertions, these are available in a separate file, available here, which can be imported or combined with DPV within the use-case. It defines domain and range assertions for properties, and disjoint declarations for classes.

JSON-LD

For applications which wish to retain the DPV semantics but do not wish to (explicitly) use [[RDF]], the [[JSON-LD]] serialisation offers the best of both worlds. [[JSON-LD]] is a format for expressing [[RDF]] semantics while being compatible with JSON – which is a widely interpreted language for expressing data. When expressed using [[JSON-LD]], the data retains the human-readability and the ability to be extended.

CSV and Flat Lists

In use-cases where DPV is to be used but without the RDF semantics, the concepts are provided in a CSV file as well as flat lists for each of its taxonomies. For example, an use-case may wish to use all of DPV's purpose taxonomies without any of the other concepts and without requiring use of RDF. For such cases, a CSV representation of DPV taxonomies is provided where only the hierarchical relationships (i.e. parent-child) are included. When using DPV in such a manner, it is advised to retain its IRI by either using the entire IRI (e.g. https://w3.org/ns/dpv#Purpose) or the suffixed portion after the namespace (i.e. *Purpose*). Doing this ensures that the data remains compatible and interoperable with the other uses and applications of DPV.

Contributing to DPV

The DPVCG welcomes participation regarding the DPV, including expansion or refinement of its terms, addressing open issues, and welcomes suggestions on their resolution or mitigation.

While we welcome participation via any and all mediums - e.g., via GitHub pull requests or issues, emails, papers, or reports - the formal resolution of contributions takes place only through the DPVCG meeting calls and mailing lists. We therefore suggest joining the group to participate in these discussions for formal approval.

For contributions to the DPV, please see the section on GitHub. The current list of open issues and their discussions to date can be found at GitHub issues. Note, GitHub Issues are preferred for discussion of concepts and proposals.

To suggest a new term, we request following information:

1. term
2. description of the term

3. whether it should be a class or a property
4. relation to existing term(s) in DPV e.g. through sub-classes
5. source (where applicable)
6. justification or relevance of why this term should be added (where not obvious)

Notes

This document is based on inspiration from the following:

- RDF 1.1 Primer https://www.w3.org/TR/rdf11-primer/
- OWL 2 Primer https://www.w3.org/TR/owl2-primer/
- PROV Model Primer https://www.w3.org/TR/prov-primer/