Object Management Group

Data Product Ontology (DPROD) — OMG submission preview

OMG Submission Preview 1 June 2026

Version:
1.1 draft

This version:
https://ekgf.github.io/grapharch/showcases/artifacts/dprod/pdf/omg-submission-dprod.pdf

Latest published version:
https://www.omg.org/spec/DPROD/

Latest editor's draft:
https://github.com/EKGF/dprod/tree/develop

Editors:
Matthias Autrata
Marcel Fröhlich
Jacobus Geluk, Initiator
Pete Rivett
Tony Seale, Chair
Natasa Varytimou

Authors:
Oli Bage
Ben Clinch
Josh Cornejo
Steve Fisher
Andrea Gioia
Charles Ivie
Carl Mattocks
Carlos Tubbax
Ben Whittam Smith

Document license

Preamble

Copyright © 2024-2026, London Stock Exchange Group plc
Copyright © 2024-2026, Federated Knowledge, LLC
Copyright © 2024-2026, agnos.ai UK Ltd.
Copyright © 2024-2026, Quantyca S.p.A.
Copyright © 2024-2026, eccenca GmbH
Copyright © 2024-2026, EDM Council, Inc.

USE OF SPECIFICATION – TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

Contributions to this specification are made under the terms of the Contributor License Agreement given at https://cla-assistant.io/EKGF/dprod. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that:

1. both the copyright notice identified above and this permission notice appear on any copies of this specification;
2. the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and
3. no modifications are made to this specification.

This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will immediately destroy any copies of the specifications in your possession or control.

PATENTS

This specification is made available under the OMG's Copyright and Non-Assertion Covenant (see https://www.omg.org/cgi-bin/doc.cgi?ipr for details). The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 9C Medway Rd, PMB 274, Milford, MA 01757, U.S.A.

TRADEMARKS

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: https://www.omg.org/legal/tm_list.htm. All other products or company names mentioned are used for identification purposes only and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials. Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

Contents

Abstract . . . viii
Status of This Document . . . viii
Publisher preface . . . ix
About the Object Management Group . . . ix
OMG Specifications . . . ix
Issues . . . ix
1 Scope . . . 1
2 Conformance . . . 2
3 Normative references . . . 3
3.1 data mesh . . . 3
3.2 dcterms . . . 3
3.3 dprod . . . 3
3.4 iri . . . 3
3.5 json-ld11 . . . 3
3.6 odrl-model . . . 3
3.7 owl2-quick-reference . . . 3
3.8 prov-overview . . . 3
3.9 rdf-schema . . . 3
3.10 rdf11-concepts . . . 4
3.11 rdf11-primer . . . 4
3.12 rdf12-turtle . . . 4
3.13 rfc5545 . . . 4
3.14 shacl . . . 4
3.15 skos-reference . . . 4
3.16 swbp-xsch-datatypes . . . 4
3.17 turtle . . . 4
3.18 uri . . . 4
3.19 vocab-dcat-3 . . . 4
3.20 xmlschema-2 . . . 5
4 Terms and Definitions . . . 6
5 Symbols . . . 7
6 Normative namespaces . . . 8
7 Data Product (DPROD) Model . . . 9
7.1 UK Bonds in practice . . . 10
7.2 Working with JSON-LD contexts . . . 11
8 Data Contracts . . . 12
8.1 DPROD extensions . . . 12
8.2 Optional adoption . . . 12
9 Data Contract Model . . . 13
10 Vocabulary and constraints reference . . . 16
10.1 OWL classes . . . 16
10.1.1 dprod:DataContract . . . 16
10.1.2 dprod:DataOffer . . . 16
10.1.3 dprod:DataProduct . . . 17
10.1.4 dprod:Enumeration . . . 18
10.1.5 dprod:EvaluationContext . . . 18
10.1.6 dprod:InformationSensitivityClassification . . . 19
10.1.7 dprod:OperandSource . . . 19
10.1.8 dprod:Schedule . . . 20
10.1.9 dprod:ScheduleFormat . . . 21

10.1.10 dprod:SecuritySchemaType . . . 21
10.2 Object properties . . . 22
10.2.1 dprod:acceptsOffer . . . 22
10.2.2 dprod:agent . . . 22
10.2.3 dprod:contractLifecycleStatus . . . 22
10.2.4 dprod:dataProductLifecycleStatus . . . 23
10.2.5 dprod:dataProductOwner . . . 23
10.2.6 dprod:domain . . . 23
10.2.7 dprod:dutyState . . . 24
10.2.8 dprod:informationSensitivityClassification . . . 24
10.2.9 dprod:inputDataset . . . 25
10.2.10 dprod:inputPort . . . 25
10.2.11 dprod:isAccessServiceOf . . . 25
10.2.12 dprod:isDistributionOf . . . 26
10.2.13 dprod:lifecycleStatus . . . 26
10.2.14 dprod:not . . . 26
10.2.15 dprod:objectOfDuty . . . 27
10.2.16 dprod:offerLifecycleStatus . . . 27
10.2.17 dprod:operandSource . . . 28
10.2.18 dprod:outputDataset . . . 28
10.2.19 dprod:outputPort . . . 28
10.2.20 dprod:protocol . . . 29
10.2.21 dprod:request . . . 29
10.2.22 dprod:schedule . . . 29
10.2.23 dprod:securitySchemaType . . . 30
10.2.24 dprod:state . . . 30
10.2.25 dprod:subjectOfDuty . . . 30
10.3 Datatype properties . . . 31
10.3.1 dprod:carriesTimeZone . . . 31
10.3.2 dprod:clock . . . 31
10.3.3 dprod:deadline . . . 31
10.3.4 dprod:effectiveDate . . . 32
10.3.5 dprod:expirationDate . . . 32
10.3.6 dprod:purpose . . . 32
10.3.7 dprod:scheduleExpression . . . 33
10.3.8 dprod:scheduleTimeZone . . . 33
10.4 Named individuals . . . 33
10.4.1 Named individual: context source . . . 33
10.4.2 Named individual: request source . . . 34
10.4.3 Named individual: state source . . . 34
10.5 Concept schemes . . . 34
10.5.1 dprod:DataContractLifecycleStatus . . . 34
10.5.2 dprod:DataProductLifecycleStatus . . . 36
10.6 SHACL node shapes . . . 37
10.6.1 dprod-contracts-shapes:AgreementShape . . . 37
10.6.2 dprod-contracts-shapes:AssetCollectionShape . . . 37
10.6.3 dprod-contracts-shapes:ConstraintShape . . . 38
10.6.4 dprod-contracts-shapes:DataContractShape . . . 38
10.6.5 dprod-contracts-shapes:DataOfferShape . . . 39
10.6.6 dprod-contracts-shapes:DutyOnlyPropertiesShape . . . 40
10.6.7 dprod-contracts-shapes:DutyShape . . . 40
10.6.8 dprod-contracts-shapes:EvaluationContextShape . . . 41

10.6.9 dprod-contracts-shapes:LeftOperandShape . . . 42
10.6.10 dprod-contracts-shapes:LogicalConstraintShape . . . 42
10.6.11 dprod-contracts-shapes:NonEmptyClausesConstraint . . . 43
10.6.12 dprod-contracts-shapes:OfferShape . . . 43
10.6.13 dprod-contracts-shapes:PartOfShape . . . 44
10.6.14 dprod-contracts-shapes:PartyCollectionShape . . . 44
10.6.15 dprod-contracts-shapes:PermissionShape . . . 45
10.6.16 dprod-contracts-shapes:PolicyShape . . . 45
10.6.17 dprod-contracts-shapes:ProhibitionShape . . . 46
10.6.18 dprod-contracts-shapes:RejectAnonymousScheduleShape . . . 46
10.6.19 dprod-contracts-shapes:RejectConsequenceShape . . . 46
10.6.20 dprod-contracts-shapes:RejectInheritAllowedShape . . . 47
10.6.21 dprod-contracts-shapes:RejectInheritFromShape . . . 47
10.6.22 dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataContractShape . . . 48
10.6.23 dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataOfferShape . . . 48
10.6.24 dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataProductShape . . . 48
10.6.25 dprod-contracts-shapes:RejectObsoleteContractLifecycleStatusSpellingShape . . . 49
10.6.26 dprod-contracts-shapes:RejectObsoleteDprodMemberOfShape . . . 49
10.6.27 dprod-contracts-shapes:RejectObsoleteDprodPartOfShape . . . 50
10.6.28 dprod-contracts-shapes:RejectObsoleteOfferLifecycleStatusSpellingShape . . . 50
10.6.29 dprod-contracts-shapes:RejectObsoleteRecurrenceShape . . . 50
10.6.30 dprod-contracts-shapes:RejectOperandPathShape . . . 51
10.6.31 dprod-contracts-shapes:RejectRemedyShape . . . 51
10.6.32 dprod-contracts-shapes:RejectRequestShape . . . 52
10.6.33 dprod-contracts-shapes:RejectRuntimeReferenceShape . . . 52
10.6.34 dprod-contracts-shapes:RejectTicketShape . . . 53
10.6.35 dprod-contracts-shapes:RejectXoneShape . . . 53
10.6.36 dprod-contracts-shapes:ScheduleFormatShape . . . 53
10.6.37 dprod-contracts-shapes:ScheduleReferenceShape . . . 54
10.6.38 dprod-contracts-shapes:ScheduleShape . . . 54
10.6.39 dprod-contracts-shapes:SetShape . . . 56
10.6.40 dprod-shapes:DataProductShape . . . 57
10.6.41 dprod-shapes:DataServiceShape . . . 57
10.6.42 dprod-shapes:DatasetShape . . . 58
10.6.43 dprod-shapes:DistributionShape . . . 58
10.6.44 dprod-shapes:InformationSensitivityClassificationShape . . . 59
10.6.45 dprod-shapes:SecuritySchemaTypeShape . . . 59
10.7 SHACL property shapes . . . 60
10.7.1 dcat:Dataset-informationSensitivityClassification . . . 60
10.7.2 dprod-contracts-shapes:ContractLifecycleStatusPropertyGroup . . . 60
10.7.3 dprod-contracts-shapes:DutyStatePropertyGroup . . . 60
10.7.4 dprod-contracts-shapes:OfferLifecycleStatusPropertyGroup . . . 61
10.7.5 dprod-contracts-shapes:ProfilePropertyGroup . . . 61
10.7.6 dprod-shapes:DataProduct-dataProductLifecycleStatus . . . 62
10.7.7 dprod-shapes:DataProduct-dataProductOwner . . . 62
10.7.8 dprod-shapes:DataProduct-description . . . 62
10.7.9 dprod-shapes:DataProduct-domain . . . 63
10.7.10 dprod-shapes:DataProduct-hasPolicy . . . 63
10.7.11 dprod-shapes:DataProduct-inputDataset . . . 63
10.7.12 dprod-shapes:DataProduct-inputPort . . . 64
10.7.13 dprod-shapes:DataProduct-label . . . 64
10.7.14 dprod-shapes:DataProduct-outputDataset . . . 65

10.7.15 dprod-shapes:DataProduct-outputPort . . . 65
10.7.16 dprod-shapes:DataProduct-purpose . . . 65
10.7.17 dprod-shapes:DataProduct-type . . . 66
10.7.18 dprod-shapes:DataService-endpointDescription . . . 66
10.7.19 dprod-shapes:DataService-endpointURL . . . 66
10.7.20 dprod-shapes:DataService-isAccessServiceOf . . . 66
10.7.21 dprod-shapes:DataService-protocol . . . 67
10.7.22 dprod-shapes:DataService-securitySchemaType . . . 67
10.7.23 dprod-shapes:Dataset-conformsTo . . . 67
10.7.24 dprod-shapes:Dataset-description . . . 68
10.7.25 dprod-shapes:Dataset-distribution . . . 68
10.7.26 dprod-shapes:Dataset-hasPolicy . . . 68
10.7.27 dprod-shapes:Dataset-label . . . 69
10.7.28 dprod-shapes:Dataset-type . . . 69
10.7.29 dprod-shapes:Distribution-accessService . . . 69
10.7.30 dprod-shapes:Distribution-conformsTo . . . 70
10.7.31 dprod-shapes:Distribution-format . . . 70
10.7.32 dprod-shapes:Distribution-isDistributionOf . . . 70
11 SBA Pool Rates . . . 72
12 Core Data Product Extensions . . . 74
13 Data Schema . . . 76
14 Data Quality . . . 78
15 Equity Trade . . . 80
16 Data Rights . . . 82
17 Observability Ports . . . 84
17.1 Defining Observability Ports in DPROD . . . 84
17.2 Example Data Product with Observability Port . . . 84
18 Data Lineage . . . 86
18.1 High Level Lineage: Between Data Products . . . 86
18.2 Detailed Level: Between Datasets . . . 88
19 Collections and Target Inheritance . . . 89
20 Data Offer and Contract . . . 91
21 Data Use Policy . . . 93
22 Evaluation Context . . . 96
23 Lifecycle and Versioning . . . 99
24 ODCS Mapping . . . 101
25 Provider Duties . . . 103
A Acknowledgements . . . 105
B Document provenance . . . 106
Recent commits . . . 106
Diagram interpretation . . . 106

Abstract

The concept of Data Products has emerged as organizations increasingly recognize the value of data as an asset to be managed and distributed like any other product. As more companies adopt decentralized data architectures, such as Data Mesh, the need for standardized methods to describe and manage data products consistently across platforms has become critical. This is where the dprod specification, built on W3C Linked Data standards, becomes essential. Without such a standard, organizations face significant challenges: inconsistent metadata across diverse data products, limited discoverability, and interoperability issues that hinder data integration from various sources. As data ecosystems grow, the lack of a common framework also impedes scalability, increases vendor lock-in, and makes it difficult to manage these products effectively.

DPROD offers a solution by providing a clear schema for describing data products, ensuring they are discoverable, interoperable, and treated with the same level of accountability as traditional products.

The W3C tech stack is perfectly suited to address these challenges because it was designed to foster interconnected, decentralized systems, providing a robust framework for creating metadata that is both machine-readable and human-understandable. DPROD enables consistent terminology across different platforms, domains, and organizations, allowing advanced users to semantically enrich their data products and connect them into distributed knowledge graphs.

As more organizations strive to build and scale data products, DPROD provides the standardization needed to ensure interoperability and unlock the full potential of decentralized data ecosystems in a controlled and mature way.

Status of This Document

This is the **DPROD 1.1 OMG submission preview** prepared from the EKGF community source repository for review through the OMG Enterprise Knowledge Graph Platform Task Force. It is not an approved OMG specification and does not assert that a formal submission has been accepted.

OMG document number: not assigned for this preview. No approval status, formal release date or official 1.1 publication URL is claimed. Those values must be supplied by the publisher before the separately configured formal release target can resolve.

The official baseline remains DPROD 1.0 beta 2, OMG document ptc/26-05-40. This edition preserves the source legal preamble and includes the formal scope, conformance, references, terminology, symbols, namespaces, models and worked examples. The publisher preface and bibliographic editions carried over from that baseline remain subject to confirmation for 1.1.

The EKGF community draft and this submission preview share the same technical placements and source commit. The closing provenance section records that commit and the generator; it does not confer standards approval. External ontology imports are not fetched implicitly.

Publisher preface

> Draft provenance note: The publisher background below is carried over from OMG document ptc/26-05-40 (1.0 beta 2). It needs confirmation for 1.1 and does not establish approval of this draft.

About the Object Management Group

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia. OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG’s specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Meta-model); and industry-specific standards for dozens of vertical markets. More information on the OMG is available at https://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL: https://www.omg.org/spec All of OMG‟s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at: OMG Headquarters 9C Medway Road, PMB 274 Milford, MA 01757 USA Tel: +1-781-444-0404 Fax: +1-781-444-0320 Email: pubs@omg.org Certain OMG specifications are also available as ISO/IEC standards. Please consult: http://www.iso.org

Issues

The reader is encouraged to report and technical or editing issues/problems with this specification to: https://www.omg.org/report_issue.htm

1 Scope

The Data Product (DPROD) specification is a profile of the Data Catalog (DCAT) Vocabulary, designed to describe Data Products. This document defines the schema and provides examples of its use.

DPROD extends DCAT to enable publishers to describe Data Products and data services in a decentralized way. By using a standard model and ontology, DPROD facilitates the consumption and aggregation of metadata from multiple Data Marketplaces. This approach increases the discoverability of products and services, supports decentralized data publishing, and enables federated search across multiple sites using a uniform query mechanism and structure.

DPROD also defines an optional Data Contracts profile of the W3C Open Digital Rights Language odrl-model, with which the provider of a data product can publish the terms on which it is offered, and a consumer can record the contract under which it is used.

The namespace for DPROD terms is `https://ekgf.org/dprod/spec/develop/`

The suggested prefix for the DPROD namespace is **`dprod`**

DPROD follows two basic principles:

- Decentralize Data Ownership: To make data integration more efficient, tasks should be shared among multiple teams. DCAT helps by offering a standard way to publish datasets in a decentralized manner.
- Harmonize Data Schemas: Using shared schemas helps unify different data formats. For instance, the DPROD specification provides a common set of rules for defining a Data Product. Users of DPROD can extend this schema as needed.

The dprod specification builds on DCAT vocab-dcat-3 by connecting DCAT Data Services to DPROD Data Products using input and output ports. These ports are used to publish and consume data from a Data Product. DPROD treats ports as DCAT data services, so the data exchanged can be described using DCAT's highly expressive metadata around distributions and datasets. This approach also allows publishers to create their own descriptions for the data they are sharing. They can use a special property called conformsTo from DCAT to link to their own set of rules or guidelines for their data.

The DPROD specification has four main aims:

- To provide unambiguous and sharable semantics to answer the question: 'What is a data product?'
- Be simple for anyone to use, but expressive enough to power large data marketplaces
- Allow organisations to reuse their existing data catalogues and dataset infrastructure
- To share common semantics across different Data Products and promote harmonization

2 Conformance

A data product is conformant with this specification if it satisfies the SHACL constraints provided in the file dprod-shapes.ttl.

Support for the Data Contracts profile is optional. A data offer, data contract or data use policy is conformant with this specification if it satisfies the SHACL constraints provided in the file dprod-contracts-shapes.ttl, and an evaluator of such policies is conformant if it produces the results defined by the profile's formal semantics.

3 Normative references

> Draft provenance note: Bibliographic editions were checked against the 1.0 beta 2 reference list, with repository-local citation records taken from this source revision. Confirm the selected editions for 1.1.

3.1 data mesh

Data Mesh Architectures. Wikipedia. https://en.wikipedia.org/wiki/Data_mesh

3.2 dcterms

DCMI Metadata Terms. DCMI Usage Board. DCMI. 20 January 2020. DCMI Recommendation. URL: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

3.3 dprod

Data Product Ontology. Tony Seale; Natasa Varytimou; Matthias Autrata; Pete Rivett; Marcel Fröhlich; Andrea Gioia; Jacobus Geluk; Ben Whittam Smith; Steve Fisher; Oli Bage; Ben Clinch; Carl Mattocks; Carlos Tubbax; Charles Ivie; Josh Cornejo. EKGF. https://ekgf.org/dprod/spec/develop/

3.4 iri

Internationalized Resource Identifiers (IRIs). M. Duerst; M. Suignard. IETF. January 2005. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc3987

3.5 json-ld11

JSON-LD 1.1. Gregg Kellogg; Pierre-Antoine Champin; Dave Longley. W3C. 16 July 2020. W3C Recommendation. URL: https://www.w3.org/TR/json-ld11/

3.6 odrl-model

ODRL Information Model 2.2. Renato Iannella; Serena Villata. W3C. 15 February 2018. W3C Recommendation. URL: https://www.w3.org/TR/odrl-model/

3.7 owl2-quick-reference

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

3.8 prov-overview

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

3.9 rdf-schema

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

3.10 rdf11-concepts

RDF 1.1 Concepts and Abstract Syntax. Richard Cyganiak; David Wood; Markus Lanthaler. W3C. 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf11-concepts/

3.11 rdf11-primer

RDF 1.1 Primer. Guus Schreiber; Yves Raimond. W3C. 24 June 2014. W3C Working Group Note. URL: https://www.w3.org/TR/rdf11-primer/

3.12 rdf12-turtle

RDF 1.2 Turtle. Gregg Kellogg; Dominik Tomaszuk. W3C. 29 August 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-turtle/

3.13 rfc5545

Internet Calendaring and Scheduling Core Object Specification (iCalendar). B. Desruisseaux, editor. IETF. September 2009. RFC 5545. https://www.rfc-editor.org/rfc/rfc5545

3.14 shacl

Shapes Constraint Language (SHACL). Holger Knublauch; Dimitris Kontokostas. W3C. 20 July 2017. W3C Recommendation. URL: https://www.w3.org/TR/shacl/

3.15 skos-reference

SKOS Simple Knowledge Organization System Reference. Alistair Miles; Sean Bechhofer. W3C. 18 August 2009. W3C Recommendation. https://www.w3.org/TR/2009/REC-skos-reference-20090818/

3.16 swbp-xsch-datatypes

XML Schema Datatypes in RDF and OWL. Jeremy Carroll; Jeff Pan. W3C. 14 March 2006. W3C Working Group Note. URL: https://www.w3.org/TR/swbp-xsch-datatypes/

3.17 turtle

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

3.18 uri

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

3.19 vocab-dcat-3

Data Catalog Vocabulary (DCAT) - Version 3. Simon Cox; Andrea Perego; Alejandra Gonzalez Beltran; Peter Winstanley; Riccardo Albertoni; David Browning. W3C. 22 August 2024. W3C Recommendation. URL: https://www.w3.org/TR/vocab-dcat-3/

3.20 xmlschema-2

XML Schema Part 2: Datatypes Second Edition. Paul V. Biron; Ashok Malhotra. W3C. 28 October 2004. W3C Recommendation. URL: https://www.w3.org/TR/xmlschema-2/

4 Terms and Definitions

All terms introduced in this specification are given definitions in the Data Product Model defined later.

5 Symbols

The following acronyms are used in this specification.

- **DPROD** - Data Product Ontology dprod
- **DCAT** - Data Catalog Vocabulary vocab-dcat-3
- **JSON-LD** - JavaScript Object Notation for Linked Data json-ld11
- **IRI** - Internationalized Resource Identifier IRI
- **ODRL** - Open Digital Rights Language odrl-model
- **OWL** - Web Ontology Language owl2-quick-reference
- **PROV** - Provenance Ontology prov-overview
- **RDF** - Resource Description Format rdf11-concepts
- **RDFS** - RDF Schema rdf-schema, swbp-xsch-datatypes
- **SHACL** - Shapes Constraint Language (shacl)
- **TTL** - Turtle (commonly used serialization for RDF) turtle rdf12-turtle
- **URI** - Uniform Resource Identifier URI
- **W3C** - World Wide Web Consortium

6 Normative namespaces

Namespaces and prefixes used in normative parts of this Profile are shown in the following table.

Prefix	Namespace IRI	Source
dprod	https://ekgf.org/dprod/spec/develop/	dprod
dprod-shapes	https://ekgf.org/dprod/spec/develop/shapes/	dprod
dprod-contracts-shapes	https://ekgf.org/dprod/spec/develop/contracts/shapes/	dprod
dcat	http://www.w3.org/ns/dcat#	vocab-dcat-3
dct	http://purl.org/dc/terms/	dcterms
odrl	http://www.w3.org/ns/odrl/2/	odrl-model
owl	http://www.w3.org/2002/07/owl#	owl2-quick-reference
prov	http://www.w3.org/ns/prov#	prov-overview
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	rdf11-primer
rdfs	http://www.w3.org/2000/01/rdf-schema#	rdf-schema
sh	http://www.w3.org/ns/shacl#	shacl
skos	http://www.w3.org/2004/02/skos/core#	skos-reference
xsd	http://www.w3.org/2001/XMLSchema#	xmlschema-2

7 Data Product (DPROD) Model

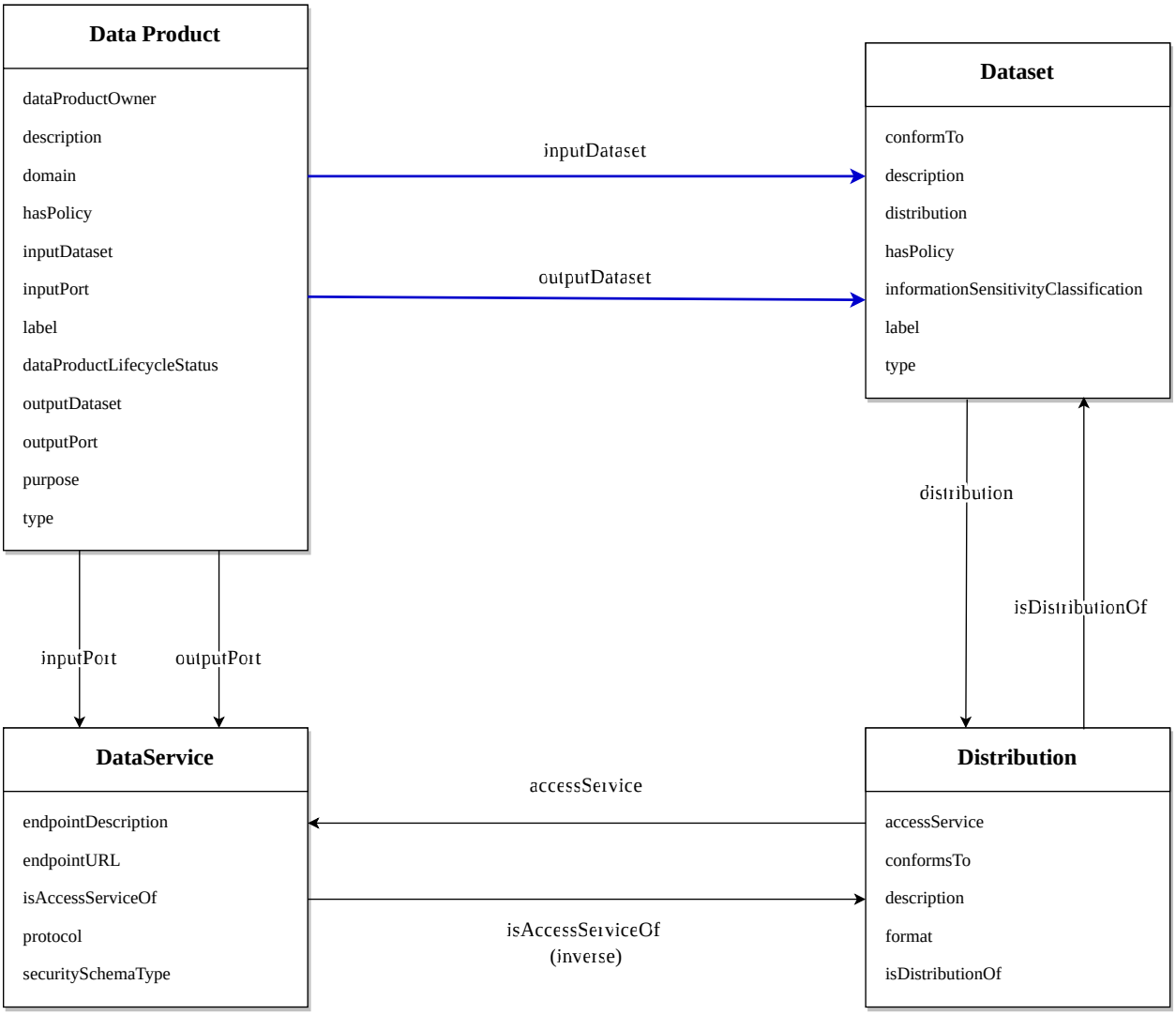
Data Mesh Architectures Data Mesh use input and output ports to manage how data enters and leaves a Data Product. These ports can handle different formats, schemas, and protocols. Input ports bring in data, while output ports send data to other Data Products for aggregation, reuse, analysis or reporting, etc.

In the vocab-dcat-3 framework, a Data Service is a way to describe services that provide access to data. Data Services give standardized, machine-readable descriptions of how to access one or more datasets or data processing functions.

Data Services specify how to access and download the data. In DPROD, Data Services are connected to Distributions by a property called isAccessServiceOf; on the Distribution one can specify formats (like CSV or JSON etc.) and provide metadata about the "physical model" of the data. Distributions link to Datasets and DCAT has a very rich vocabulary for describing every aspect of a dataset. Finally, Datasets use the conformsTo property to link to the "logical model" where one can specify one's own rich semantic metadata.

By linking Data Product ports to DCAT DataServices (vocab-dcat-3), DPROD can describe Data Products in a way that machines can read across the organization. This makes it easier for data teams to build and manage their own data products independently, while still working well with the rest of the organization's data.

Using standards like DCAT helps create a strong and clear way to define Data Products. It ensures that as data becomes more complex, the methods for describing, sharing, and using data stay consistent and reliable. It also allows different organizations to share data securely and in a standardized way.



Overview of the DPROD model and its relationship with DCAT classes

The Profile consists of the following classes:

- Data Mesh (`dcat:Catalog`) - The collection of Data Products
- Data Product (`dprod:DataProduct`) - A data product may have input and output ports, code and metadata
- Port (`dcat:DataService`) - A digital interface that provides access to a Dataset. This can be an HTTP URL, a Database or a FileShare, etc.
- Distribution (`dcat:Distribution`) - A specific representation of a dataset (CSV, JSON, ADLS etc.) which can conform to a physical model
- Dataset (`dcat:Dataset`) - A collection of related data that can conform to a logical model

7.1 UK Bonds in practice

As DCAT Data Services, the DPROD input and output ports can specify connection details, they have distributions that define formats, and link to datasets that conform to shared schemas. In this example, the UK Bonds Data Product includes an output port, which is a RESTful API. This API delivers JSON data conforming to the shared FIBO specification for callable bonds.

Example 1

```
  {
    "@context": "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    "id": "https://y.com/products/uk-bonds",
    "type": "DataProduct",
    "title": "UK Bonds",
    "description": "UK Bonds is your one-stop-shop for all your bonds!",
    "dataProductOwner": "https://www.linkedin.com/in/tonyseale/",
    "dataProductLifecycleStatus": "https://ekgf.org/dprod/spec/develop/data/lifecycle-s
tatus/Consume",
    "outputPort": {
      "id": "https://y.com/products/uk-bonds/ports/10-year-api",
      "type": "DataService",
      "endpointURL": "https://y.com/uk-10-year-bonds",
      "isAccessServiceOf": {
        "id": "https://y.com/products/uk-bonds/distributions/10-year-json",
        "type": "Distribution",
        "format": "https://www.iana.org/assignments/media-types/application/json",
        "isDistributionOf": {
          "id": "https://y.com/products/uk-bonds/datasets/10-year",
          "type": "Dataset",
          "conformsTo": "https://spec.edmcouncil.org/fibo/ontology/SEC/Debt/Bonds/Calla
bleBond"
        }
      }
    }
  }
```

7.2 Working with JSON-LD contexts

Note
DPROD publishes two JSON-LD contexts that denote the same graph. The examples in this specification use the simple context, `dprod-simple.jsonld`, which lets a document use plain terms such as `outputPort`, `title`, `id` and `type`. When DPROD JSON is combined with other JSON-LD contexts, use `dprod-context.jsonld` and prefixed terms such as `dprod:outputPort` and `dct:title` with the keywords `@id` and `@type`, so that neither side claims generic names such as `title` or `format`. Neither context has an `@vocab`, so an undefined term is visibly undefined rather than silently coined in the DPROD namespace.

The JSON above can be pasted into https://json-ld.org/playground to see the RDF it denotes.

Note
The following sections are driven by the Shapes definitions for DPROD, which represent the properties expected to be used for instances of the above classes. As such, they include properties defined by the DCAT specification which DPROD extends. The prefix in the *identifier* for each entry indicates the ontology defining the property.

8 Data Contracts

A data product is only useful once someone can rely on it. The provider of a product commits to deliver data on a schedule, to keep it conformant to a schema, and to give notice before it changes; the consumer, in turn, agrees to use the data only in permitted ways and to meet obligations of its own, such as reporting usage. In a Data Mesh these commitments are made between many autonomous teams, so they need to be recorded in a form that machines can validate and evaluate, not just in prose.

DPROD Data Contracts is a profile of the W3C Open Digital Rights Language (ODRL) 2.2 odrl-model for exactly this purpose. It uses ODRL's own terms for everything ODRL already defines: a policy, its permissions, prohibitions and duties, the constraints on them, and the parties and assets they refer to.

8.1 DPROD extensions

DPROD adds vocabulary only where ODRL leaves behaviour undefined:

- **Two policy types.** A `dprod:DataOffer` is an `odrl:Offer` published by a data provider. A `dprod:DataContract` is an `odrl:Agreement` that records a consumer's acceptance of exactly one offer, through `dprod:acceptsOffer`, and binds both parties.
- **Bilateral duties.** ODRL duties fall on the assignee. In a data contract the provider has duties too, so DPROD names the party that bears each duty (`dprod:subjectOfDuty`) and the party it is owed to (`dprod:objectOfDuty`).
- **Schedules and deadlines.** A duty may recur on a reusable, identified `dprod:Schedule` (for example an RFC5545 recurrence rule with its start time and timezone), referenced through `dprod:schedule`, and each instance has a fulfilment window (`dprod:deadline`).
- **Authored status and computed state.** A publisher records where an offer or contract is in its lifecycle. An evaluator computes, from observable facts, whether each duty is pending, active, fulfilled or violated (`dprod:dutyState`). The two are kept apart on purpose.
- **Deterministic evaluation.** Every operand a constraint refers to is bound to one property on one normalised input (the authorisation request, a state-of-the-world snapshot, or the evaluation context itself), prohibitions always win over permissions, and the same inputs always produce the same decision.

Because it is a proper ODRL profile, every DPROD data contract is a valid ODRL 2.2 policy. An ODRL processor that knows nothing of DPROD can still read the permissions and prohibitions; a DPROD-aware processor additionally enforces duty state, bilateral duties and the evaluation rules.

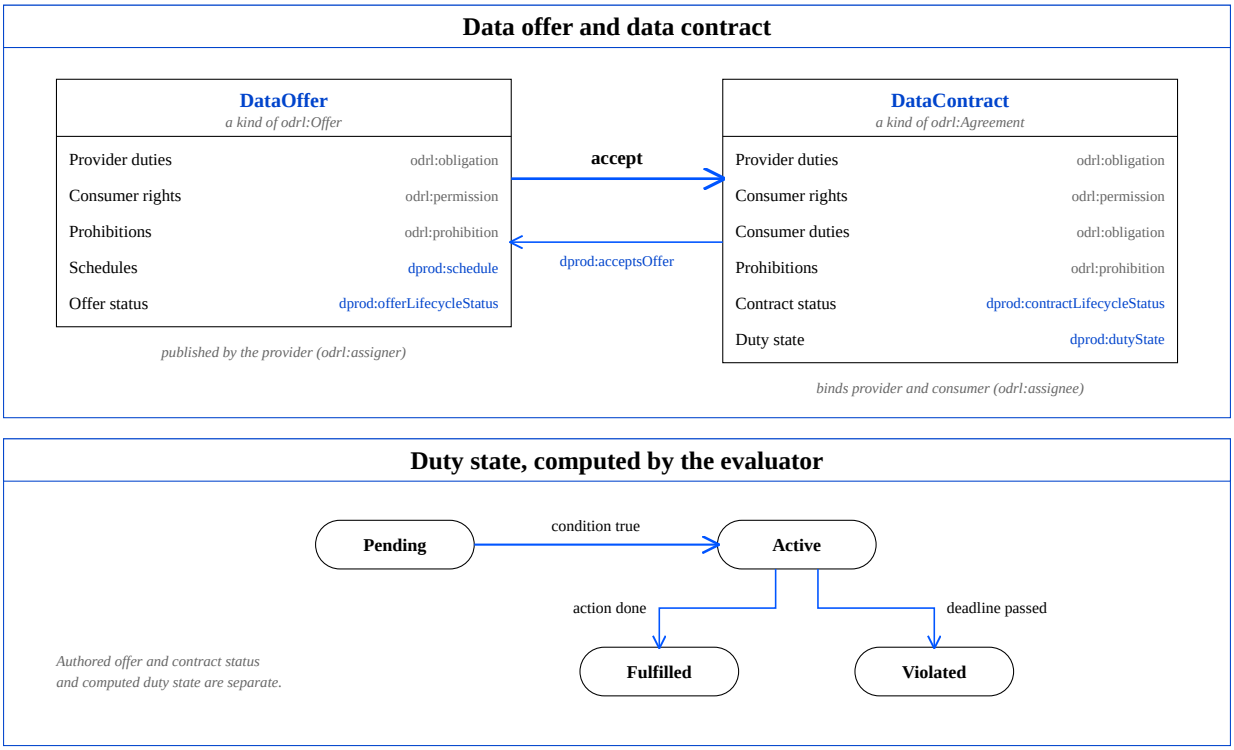
The profile is defined by three files, published alongside this document: dprod-contracts.ttl (the vocabulary), dprod-contracts-shapes.ttl (the SHACL constraints an instance document must satisfy) and dprod-contracts-prof.ttl (the profile declaration). The normative evaluation semantics, which define what a conforming evaluator must compute, are maintained in the formal semantics document; authoring guidance for contract writers and policy writers is in the contracts documentation.

8.2 Optional adoption

The Data Contracts profile is optional. A data product description conforms to DPROD whether or not it carries any policy; an implementation that does process contracts must satisfy the contracts shapes as stated in the Conformance section.

9 Data Contract Model

A provider publishes a `dprod:DataOffer`, an `odrl:Offer` setting out the provider's duties, the consumer's rights and what is prohibited. When a consumer accepts it, the result is a `dprod:DataContract`, an `odrl:Agreement` that binds both parties, may add duties on the consumer, and links back to the offer it accepts. Each row of the figure names the property that carries it; DPROD terms are shown in blue. The lower part shows the state an evaluator computes for each duty.



Overview of DPROD Data Contracts: offer, acceptance, contract and duty state

A data product, or any of its datasets, distributions and data services, is an `odrl:Asset` when it appears as the `odrl:target` of a policy. Teams and organisations are `odrl:Party` instances. Both may be grouped into ODRL collections using `odrl:partOf`; DPROD evaluates membership over the transitive closure of that property, so a rule on a collection applies to everything in it.

The lifecycle of an offer or a contract is recorded with `dprod:offerLifecycleStatus` and `dprod:contractLifecycleStatus`. Both are sub-properties of the core `dprod:lifecycleStatus` and take any `skos:Concept`; DPROD ships `dprod:DataContractLifecycleStatus`, an optional concept scheme with the values `dprod:Pending`, `dprod:Active`, `dprod:Fulfilled` and `dprod:Violated`, which an enterprise may use, extend or replace.

A DPROD-aware evaluator answers the question "may this agent perform this action on this asset now?" It never reads a live graph while doing so. Instead the caller assembles one immutable `dprod:EvaluationContext` holding the authorisation request, a snapshot of the state of the world, the requesting agent and the clock. Each `odrl:LeftOperand` used in a constraint declares which of those inputs it reads (`dprod:operandSource`) and which single property it reads from it (`dprod:operandProperty`). A value that is missing, ambiguous or of the wrong type is an evaluation error rather than a silently false constraint. ODRL's own `odrl:dateTime` operand is bound to the context clock, and `dprod:currentAgent` to the requesting agent.

In this example the Trading Data Team offers its Market Price Data product with a single provider duty (deliver every 15 minutes, within 5 minutes of the scheduled time), one consumer permission and one prohibition. The Analytics Team has accepted the offer for 2026; the contract names both parties and links to the offer it accepts.

Example 2

```
  {
    "@context": [
      "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
      { "ex": "https://example.org/" }
    ],
    "id": "ex:tradingMarketData2026",
    "type": "DataContract",
    "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
    "assigner": "ex:tradingDataTeam",
    "assignee": "ex:analyticsTeam",
    "target": "ex:marketPrices",
    "contractLifecycleStatus": "Active",
    "effectiveDate": "2026-01-15T00:00:00Z",
    "expirationDate": "2026-12-31T23:59:59Z",
    "permission": {
      "type": "Permission",
      "assignee": "ex:analyticsTeam",
      "action": "display"
    },
    "acceptsOffer": {
      "id": "ex:marketDataFeed",
      "type": "DataOffer",
      "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
      "assigner": "ex:tradingDataTeam",
      "target": "ex:marketPrices",
      "offerLifecycleStatus": "Active",
      "obligation": {
        "type": "Duty",
        "subjectOfDuty": "ex:tradingDataTeam",
        "action": "ex:deliver",
        "schedule": {
          "id": "ex:every15Minutes",
          "type": "Schedule",
          "conformsTo": "dprod:Rfc5545ScheduleFormat",
          "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T000000\nRRULE:FREQ
=MINUTELY;INTERVAL=15"
        },
        "deadline": { "@value": "PT5M", "type": "xsd:duration" }
      },
      "permission": { "type": "Permission", "action": "display" },
      "prohibition": { "type": "Prohibition", "action": "distribute" }
    }
  }
```

ODRL terms are written as plain names, in the same way DCAT terms are in a data product description. The context knows that an action, an operator or a status names a vocabulary term, so `"action": "display"` denotes `odrl:display` and `"contractLifecycleStatus": "Active"` denotes `dprod:Active`. The offer is shown nested inside the contract for readability. In practice a provider publishes the offer once, as its own resource, and each contract that accepts it refers to it by identifier. The full version of this example, with the domain vocabulary, the parties and the evaluated duty instances, is in the Data Offer and Contract worked example.

The following class definitions are driven by the contracts Shapes, in the same way as the core model above. Only classes DPROD defines are listed here; the ODRL classes the profile reuses are defined by odrl-model, and the DPROD properties that extend them follow in the next section.

10 Vocabulary and constraints reference

The following sections document the named vocabulary resources and SHACL shapes from this same repository revision. The preceding conformance clause identifies the authoritative shape files; the human-readable reference does not replace them.

10.1 OWL classes

10.1.1 dprod:DataContract

Activated data offer binding provider and consumer.

A DataContract is an activated DataOffer. - assigner: data provider (from offer) - assignee: data consumer Bilateral: both parties may have active duties.

dprod:DataContract	
https://ekgf.org/dprod/spec/develop/DataContract	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	data contract
rdfs:subClassOf	odrl:Agreement
owl:disjointWith	dprod:DataOffer

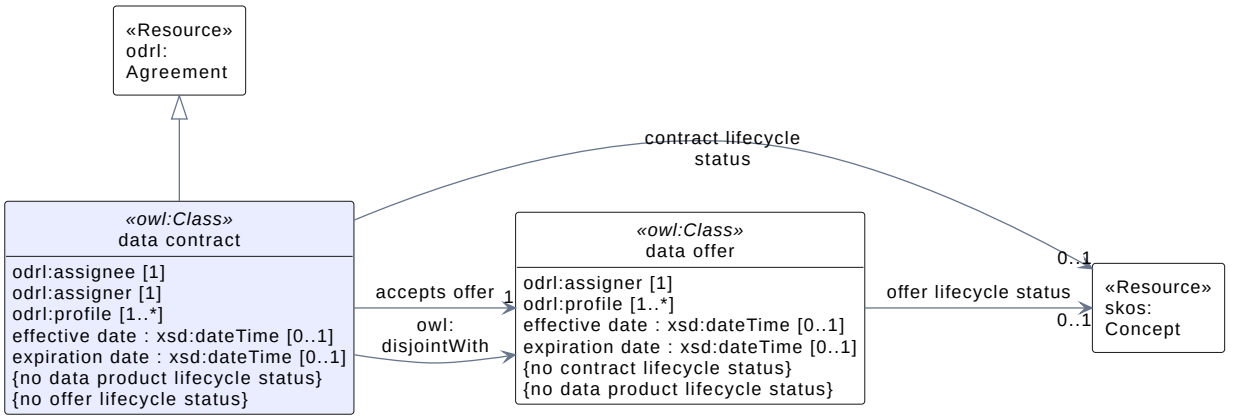


Diagram note

10.1.2 dprod:DataOffer

Policy offer defining data access terms.

A DataOffer is an Offer from a data provider. It defines permissions, duties, and prohibitions for data access. When accepted by a consumer, becomes a DataContract (Agreement).

dprod:DataOffer	
https://ekgf.org/dprod/spec/develop/DataOffer	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	data offer
rdfs:subClassOf	odrl:Offer

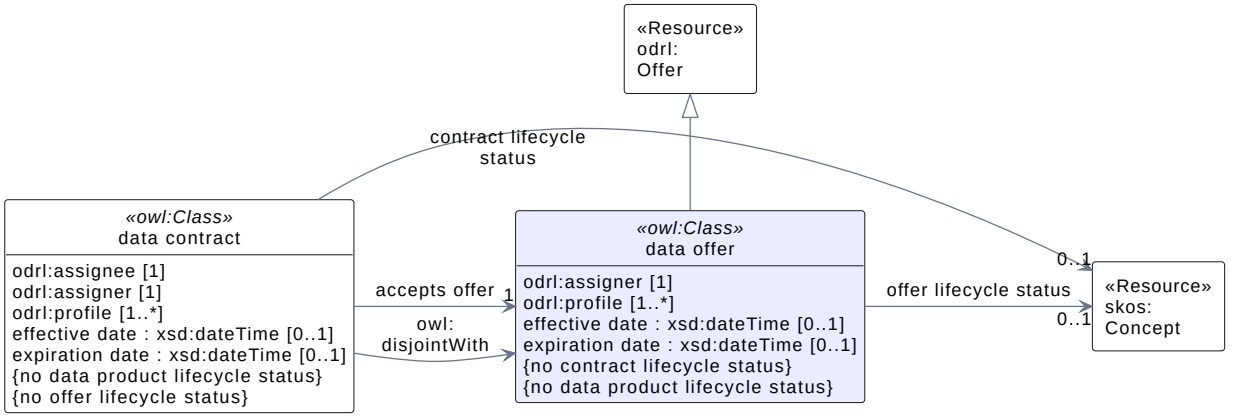


Diagram note

10.1.3 dprod:DataProduct

A rational, managed, and governed collection of data, with purpose, value and ownership, meeting consumer needs over a planned life-cycle.

A data product may have input and output ports, code and metadata

dprod:DataProduct	
https://ekgf.org/dprod/spec/develop/DataProduct	
RDF predicate	**Value**
rdf:type	rdfs:Class
rdf:type	owl:Class
rdfs:label	data product
rdfs:subClassOf	dcat:Resource

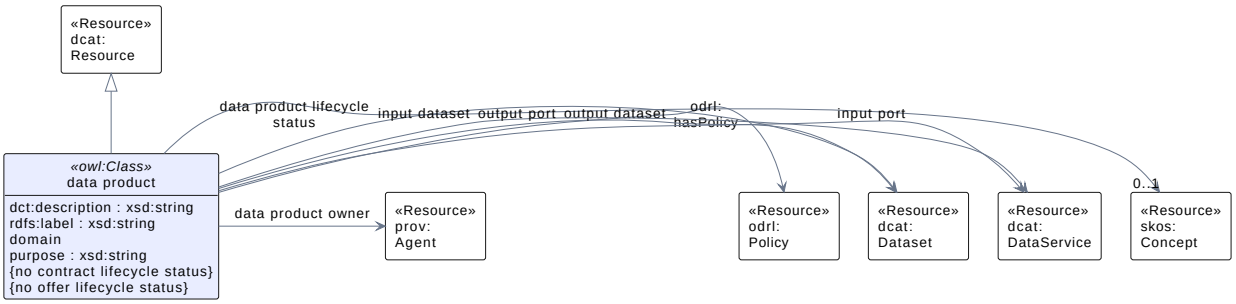


Diagram note

10.1.4 dprod:Enumeration

The superclass of enumeration lists referenced from data product related artifacts.

dprod:Enumeration	
https://ekgf.org/dprod/spec/develop/Enumeration	
RDF predicate	**Value**
rdf:type	rdfs:Class
rdf:type	owl:Class
rdfs:label	enumeration

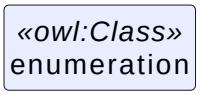


Diagram note

10.1.5 dprod:EvaluationContext

Immutable, provenance-bearing input graph for one policy evaluation.

The immutable container for runtime operand resolution. DPROD does not evaluate policies; an evaluator is an external program that implements this specification. The calling system that needs a decision constructs one EvaluationContext for each evaluation and supplies the authorization request, an immutable state-of-the-world snapshot, the requesting agent, and the evaluation clock through the four normative properties. The evaluator reads only this context. The context is a prov:Entity so implementations can retain the exact input provenance needed to reproduce and audit a decision. Evaluation never reads a live global graph or dereferences an undeclared source while resolving an operand binding.

dprod:EvaluationContext	
https://ekgf.org/dprod/spec/develop/EvaluationContext	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	evaluation context
rdfs:subClassOf	prov:Entity

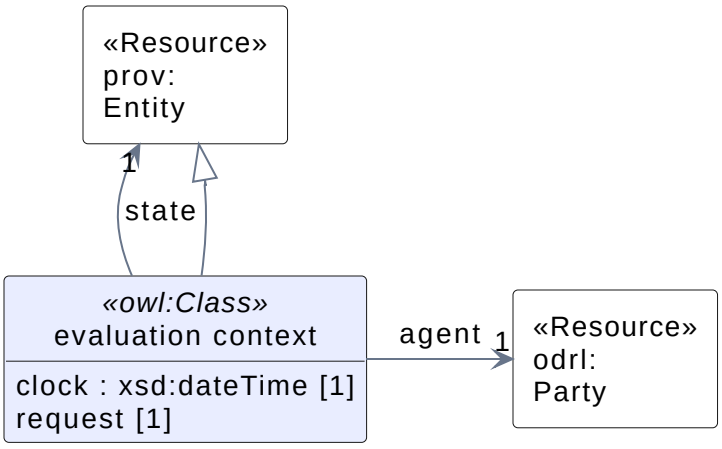


Diagram note

10.1.6 dprod:InformationSensitivityClassification

A classification of the information within a dataset to indicate the level of control and protection that must be applied.

dprod:InformationSensitivityClassification	
https://ekgf.org/dprod/spec/develop/InformationSensitivityClassification	
RDF predicate	**Value**
rdf:type	rdfs:Class
rdf:type	owl:Class
rdfs:label	information sensitivity classification
rdfs:seeAlso	https://spec.edmcouncil.org/cdmc/ontology/Classification/InformationSensitivity

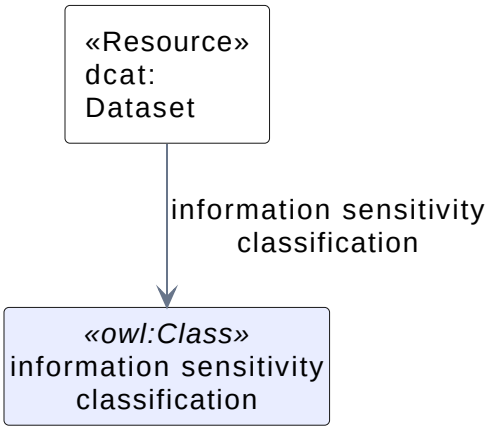


Diagram note

10.1.7 dprod:OperandSource

Closed set of normalized source nodes from which an operand value may be read.

An operand reads one direct property from exactly one normalized source: the authorization request, the immutable world snapshot, or the evaluation context itself. No other source and no graph traversal are supported.

dprod:OperandSource	
https://ekgf.org/dprod/spec/develop/OperandSource	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	operand source
owl:oneOf	(dprod:requestSource, dprod:stateSource, dprod:contextSource)

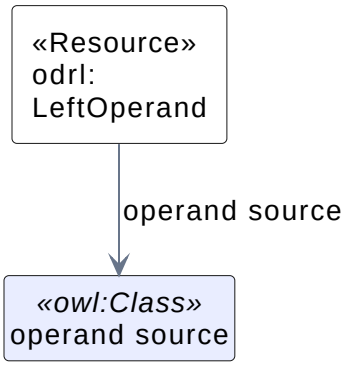


Diagram note

10.1.8 dprod:Schedule

Reusable, identified definition of scheduled occurrences.

A Schedule defines when something occurs without defining what occurs or what the occurrence means. DPROD resources refer to it through dprod:schedule; the referring resource and property retain the business meaning. Each Schedule has one authoritative expression and identifies its exact representation through dct:conformsTo. Format-specific profiles may extend the built-in RFC 5545 and POSIX crontab formats.

dprod:Schedule	
https://ekgf.org/dprod/spec/develop/Schedule	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	schedule

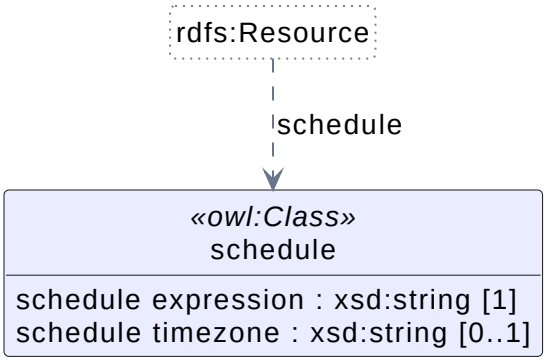


Diagram note

10.1.9 dprod:ScheduleFormat

Identified grammar and interpretation rules for schedule expressions.

A ScheduleFormat identifies an exact schedule grammar or dialect rather than a general implementation family. Third parties may define additional formats and supply corresponding validation and processing profiles.

dprod:ScheduleFormat	
https://ekgf.org/dprod/spec/develop/ScheduleFormat	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	schedule format

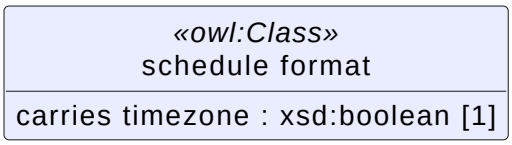


Diagram note

10.1.10 dprod:SecuritySchemaType

A classification encompassing a set of rules used for authentication and communication.

dprod:SecuritySchemaType	
https://ekgf.org/dprod/spec/develop/SecuritySchemaType	
RDF predicate	**Value**
rdf:type	rdfs:Class
rdf:type	owl:Class
rdfs:label	security schema type

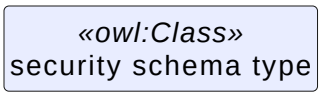


Diagram note

10.2 Object properties

10.2.1 dprod:acceptsOffer

The offer this contract activates.

dprod:acceptsOffer	
https://ekgf.org/dprod/spec/develop/acceptsOffer	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataContract
rdfs:label	accepts offer
rdfs:range	dprod:DataOffer

10.2.2 dprod:agent

Requesting agent supplied for one policy evaluation.

dprod:agent	
https://ekgf.org/dprod/spec/develop/agent	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:EvaluationContext
rdfs:label	agent
rdfs:range	odrl:Party

10.2.3 dprod:contractLifecycleStatus

Where this contract sits in its administrative lifecycle — typically pre-effective, in force, completed, or breached; assigned by the contract management process.

Value is any skos:Concept; dprod:DataContractLifecycleStatus is one example concept scheme shipped with DPROD.

dprod:contractLifecycleStatus	
https://ekgf.org/dprod/spec/develop/contractLifecycleStatus	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

dprod:contractLifecycleStatus	
https://ekgf.org/dprod/spec/develop/contractLifecycleStatus	
RDF predicate	**Value**
rdfs:domain	dprod:DataContract
rdfs:label	contract lifecycle status
rdfs:range	skos:Concept
rdfs:subPropertyOf	dprod:lifecycleStatus

10.2.4 dprod:dataProductLifecycleStatus

The authored development lifecycle status of a data product.

Values are SKOS concepts. DPROD supplies dprod:DataProductLifecycleStatus as an optional reference concept scheme; enterprises may use another scheme.

dprod:dataProductLifecycleStatus	
https://ekgf.org/dprod/spec/develop/dataProductLifecycleStatus	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	data product lifecycle status
rdfs:range	skos:Concept
rdfs:subPropertyOf	dprod:lifecycleStatus

10.2.5 dprod:dataProductOwner

The agent that is accountable overall for the data product, including managing it through its lifecycle.

dprod:dataProductOwner	
https://ekgf.org/dprod/spec/develop/dataProductOwner	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	data product owner
rdfs:range	prov:Agent

10.2.6 dprod:domain

The business or information area supported by the data product.

The domain is intended to be a resource in its own right. This specification does not constrain the class to be used.

dprod:domain	
https://ekgf.org/dprod/spec/develop/domain	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	domain

10.2.7 dprod:dutyState

The latest evaluation result for this duty, computed by an evaluator from observable facts; recomputed on demand rather than authored.

Value is any skos:Concept. The state machine in formal-semantics §5 is one possible interpretation. Implementations are free to use a different state vocabulary and/or state machine. This evaluator-computed property is deliberately separate from the authored dprod:lifecycleStatus hierarchy.

dprod:dutyState	
https://ekgf.org/dprod/spec/develop/dutyState	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	odrl:Duty
rdfs:label	duty state
rdfs:range	skos:Concept

10.2.8 dprod:informationSensitivityClassification

More granular classification that indicates the level of control and protection that must be applied to the asset due to the nature of the data and its sensitivity or importance to the organization.

dprod:informationSensitivityClassification	
https://ekgf.org/dprod/spec/develop/informationSensitivityClassification	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:Dataset
rdfs:label	information sensitivity classification
rdfs:range	dprod:InformationSensitivityClassification

10.2.9 dprod:inputDataset

The source data made available to the data product through input data services. Depending on the lifecycle of the data product, this may be a stated or inferred relationship aligned with the input ports.

dprod:inputDataset	
https://ekgf.org/dprod/spec/develop/inputDataset	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	input dataset
rdfs:range	dcat:Dataset

10.2.10 dprod:inputPort

A set of services exposed by a data product to collect its source data and makes it available for further internal transformation. An input port can receive data from one or more upstream sources in a push (i.e. asynchronous subscription) or pop mode (i.e. synchronous query). Each data product may have one or more input ports.

dprod:inputPort	
https://ekgf.org/dprod/spec/develop/inputPort	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	input port
rdfs:range	dcat:DataService

10.2.11 dprod:isAccessServiceOf

The dataset distribution that is being offered through this data service.

dprod:isAccessServiceOf	
https://ekgf.org/dprod/spec/develop/isAccessServiceOf	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:DataService
rdfs:label	is access service of

dprod:isAccessServiceOf	
https://ekgf.org/dprod/spec/develop/isAccessServiceOf	
RDF predicate	**Value**
rdfs:range	dcat:Distribution
owl:inverseOf	dcat:accessService

10.2.12 dprod:isDistributionOf

The dataset that this distribution makes available.

dprod:isDistributionOf	
https://ekgf.org/dprod/spec/develop/isDistributionOf	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:Distribution
rdfs:label	is distribution of
owl:inverseOf	dcat:distribution

10.2.13 dprod:lifecycleStatus

The authored position of a DPROD resource in its lifecycle.

Abstract super-property for authored lifecycle statuses. Concrete sub-properties define their resource-specific domains; values come from an open SKOS taxonomy.

dprod:lifecycleStatus	
https://ekgf.org/dprod/spec/develop/lifecycleStatus	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:label	lifecycle status
rdfs:range	skos:Concept

10.2.14 dprod:not

Logical negation on a constraint.

ODRL defines odrl:and and odrl:or on LogicalConstraint but lacks negation. DPROD adds dprod:not following the same pattern.

dprod:not	
https://ekgf.org/dprod/spec/develop/not	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	odrl:LogicalConstraint
rdfs:label	not
rdfs:range	(owl:unionOf: (odrl:Constraint, odrl:LogicalConstraint))

Example:

```
_:lc a odrl:LogicalConstraint ;
    dprod:not _:c1 .
```

10.2.15 dprod:objectOfDuty

Party affected by the duty action.

The party affected by or receiving the result of the duty action. Sub-property of odrl:function for ODRL processor compatibility.

dprod:objectOfDuty	
https://ekgf.org/dprod/spec/develop/objectOfDuty	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	odrl:Duty
rdfs:label	object
rdfs:range	odrl:Party
rdfs:subPropertyOf	odrl:function

10.2.16 dprod:offerLifecycleStatus

Where this offer sits in its publication lifecycle — typically drafted, published, withdrawn, or superseded; assigned by the publisher.

Value is any skos:Concept; dprod:DataContractLifecycleStatus is one example concept scheme shipped with DPROD.

dprod:offerLifecycleStatus	
https://ekgf.org/dprod/spec/develop/offerLifecycleStatus	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataOffer
rdfs:label	offer lifecycle status
rdfs:range	skos:Concept

dprod:offerLifecycleStatus	
https://ekgf.org/dprod/spec/develop/offerLifecycleStatus	
RDF predicate	**Value**
rdfs:subPropertyOf	dprod:lifecycleStatus

10.2.17 dprod:operandSource

Normalized graph from which an operand reads its value.

Exactly one of dprod:requestSource, dprod:stateSource, or dprod:contextSource. The source is selected before the evaluator performs its single property lookup.

dprod:operandSource	
https://ekgf.org/dprod/spec/develop/operandSource	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	odrl:LeftOperand
rdfs:label	operand source
rdfs:range	dprod:OperandSource

10.2.18 dprod:outputDataset

The data that is exposed by the data product through output data services in a way that can be understood and trusted. Depending on the lifecycle of the data product, this may be a stated or inferred relationship aligned with the output ports.

dprod:outputDataset	
https://ekgf.org/dprod/spec/develop/outputDataset	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	output dataset
rdfs:range	dcat:Dataset

10.2.19 dprod:outputPort

A set of services exposed by a data product to share the generated data in a way that can be understood and trusted.

dprod:outputPort	
https://ekgf.org/dprod/spec/develop/outputPort	
RDF predicate	**Value**
rdf:type	rdf:Property

dprod:outputPort	
https://ekgf.org/dprod/spec/develop/outputPort	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:DataProduct
rdfs:label	output port
rdfs:range	dcat:DataService

10.2.20 dprod:protocol

A protocol (possibly one of many options) used to communicate with this data service.

dprod:protocol	
https://ekgf.org/dprod/spec/develop/protocol	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:DataService
rdfs:label	protocol

10.2.21 dprod:request

Authorization request supplied for one policy evaluation.

Selects the normalized request node used by dprod:requestSource operand bindings.

dprod:request	
https://ekgf.org/dprod/spec/develop/request	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:EvaluationContext
rdfs:label	request
rdfs:range	rdfs:Resource

10.2.22 dprod:schedule

Reusable schedule governing occurrences associated with a resource.

Links any DPROD or ODRL resource to an identified dprod:Schedule. This property deliberately has no domain: duties, permissions, maintenance arrangements, backup obligations, and other resources may all refer to a schedule. The referring resource determines what the occurrences mean.

dprod:schedule	
https://ekgf.org/dprod/spec/develop/schedule	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	schedule
rdfs:range	dprod:Schedule

10.2.23 dprod:securitySchemaType

The security schema type used for authentication and communication with this Data Service.

dprod:securitySchemaType	
https://ekgf.org/dprod/spec/develop/securitySchemaType	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:DataService
rdfs:label	security schema type

10.2.24 dprod:state

Immutable state-of-the-world snapshot supplied for one policy evaluation.

Selects the normalized snapshot node used by dprod:stateSource operand bindings. The object is supplied by the calling system that builds the evaluation context, never an instruction to query live external state.

dprod:state	
https://ekgf.org/dprod/spec/develop/state	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dprod:EvaluationContext
rdfs:label	state
rdfs:range	prov:Entity

10.2.25 dprod:subjectOfDuty

Party bearing the duty (must perform the action).

The duty bearer. When omitted, the bearer is the policy's assignee (odrl:assignee), or the requesting agent in a policy without one; provider duties name the provider explicitly. Sub-property of odrl:function for ODRL processor compatibility.

dprod:subjectOfDuty	
https://ekgf.org/dprod/spec/develop/subjectOfDuty	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	odrl:Duty
rdfs:label	subject
rdfs:range	odrl:Party
rdfs:subPropertyOf	odrl:function

10.3 Datatype properties

10.3.1 dprod:carriesTimeZone

Whether expressions in this schedule format embed their own timezone.

Declared by every dprod:ScheduleFormat. When false, each Schedule using the format must declare dprod:scheduleTimeZone; when true, the expression is the timezone authority. This flag lets validation enforce timezone completeness for extension formats without knowing their grammar.

dprod:carriesTimeZone	
https://ekgf.org/dprod/spec/develop/carriesTimeZone	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	dprod:ScheduleFormat
rdfs:label	carries timezone
rdfs:range	xsd:boolean

10.3.2 dprod:clock

Immutable timestamp at which policy evaluation occurs.

dprod:clock	
https://ekgf.org/dprod/spec/develop/clock	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	dprod:EvaluationContext
rdfs:label	clock
rdfs:range	xsd:dateTime

10.3.3 dprod:deadline

Time constraint for duty fulfillment.

Supports multiple forms: - xsd:dateTime: absolute deadline (e.g., 2026-12-31T23:59:59Z) - xsd:duration: relative to activation (e.g., P30D, PT24H) For duration: deadline = activationTime + duration. No rdfs:range declared because the range is a union of datatypes; SHACL enforces the allowed types.

dprod:deadline	
https://ekgf.org/dprod/spec/develop/deadline	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	odrl:Duty
rdfs:label	deadline

10.3.4 dprod:effectiveDate

When the offer or contract becomes effective.

dprod:effectiveDate	
https://ekgf.org/dprod/spec/develop/effectiveDate	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	(owl:unionOf: (dprod:DataOffer, dprod:DataContract))
rdfs:label	effective date
rdfs:range	xsd:dateTime

10.3.5 dprod:expirationDate

When the offer or contract expires.

dprod:expirationDate	
https://ekgf.org/dprod/spec/develop/expirationDate	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	(owl:unionOf: (dprod:DataOffer, dprod:DataContract))
rdfs:label	expiration date
rdfs:range	xsd:dateTime

10.3.6 dprod:purpose

A description of the objectives and intended usage of the data product.

dprod:purpose	
https://ekgf.org/dprod/spec/develop/purpose	
RDF predicate	**Value**
rdf:type	rdf:Property
rdf:type	owl:DatatypeProperty
rdfs:domain	dprod:DataProduct
rdfs:label	purpose
rdfs:range	xsd:string

10.3.7 dprod:scheduleExpression

The single authoritative expression of a schedule.

dprod:scheduleExpression	
https://ekgf.org/dprod/spec/develop/scheduleExpression	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	dprod:Schedule
rdfs:label	schedule expression
rdfs:range	xsd:string

10.3.8 dprod:scheduleTimeZone

IANA timezone used to interpret a schedule expression.

Required when the selected schedule format does not carry its own timezone (dprod:carriesTimeZone false). When the format embeds a timezone in the expression, the expression is authoritative and this property, if present, must agree with it.

dprod:scheduleTimeZone	
https://ekgf.org/dprod/spec/develop/scheduleTimeZone	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	dprod:Schedule
rdfs:label	schedule timezone
rdfs:range	xsd:string

10.4 Named individuals

10.4.1 Named individual: context source

Instance of: dprod:OperandSource

Evaluation context node containing built-in agent and clock values.

dprod:contextSource	
https://ekgf.org/dprod/spec/develop/contextSource	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	dprod:OperandSource
rdfs:label	context source

10.4.2 Named individual: request source

Instance of: dprod:OperandSource

Normalized authorization request node.

The request node built by the calling system: what is being asked for (agent, action, asset) plus every request fact the policies need, flattened onto that node. It is plain RDF, not an odrl:Request.

dprod:requestSource	
https://ekgf.org/dprod/spec/develop/requestSource	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	dprod:OperandSource
rdfs:label	request source

10.4.3 Named individual: state source

Instance of: dprod:OperandSource

Immutable normalized state-of-the-world snapshot node.

dprod:stateSource	
https://ekgf.org/dprod/spec/develop/stateSource	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	dprod:OperandSource
rdfs:label	state source

10.5 Concept schemes

10.5.1 dprod:DataContractLifecycleStatus

Optional reference concept scheme covering common statuses (Pending, Active, Fulfilled, Violated). It can be used with authored offer and contract lifecycle-status properties. Implementations are free to use it, extend it, or use a different skos:ConceptScheme entirely. dprod:dutyState is an evaluator-computed state and is not part of the lifecycle-status hierarchy.

dprod:DataContractLifecycleStatus	
https://ekgf.org/dprod/spec/develop/DataContractLifecycleStatus	
RDF predicate	**Value**
dct:title	DPROD Data Contract Lifecycle Status
rdf:type	skos:ConceptScheme
rdfs:label	DPROD data contract lifecycle status
skos:hasTopConcept	dprod:Active
skos:hasTopConcept	dprod:Fulfilled
skos:hasTopConcept	dprod:Pending
skos:hasTopConcept	dprod:Violated

Concepts in this scheme

- active
- fulfilled
- pending
- violated

10.5.1.1 dprod:Active

Condition satisfied, action required; in force.

dprod:Active	
https://ekgf.org/dprod/spec/develop/Active	
RDF predicate	**Value**
rdf:type	skos:Concept
rdfs:label	active
skos:prefLabel	active
skos:topConceptOf	dprod:DataContractLifecycleStatus

10.5.1.2 dprod:Fulfilled

Action performed; obligations complete.

dprod:Fulfilled	
https://ekgf.org/dprod/spec/develop/Fulfilled	
RDF predicate	**Value**
rdf:type	skos:Concept
rdfs:label	fulfilled
skos:prefLabel	fulfilled

dprod:Fulfilled	
https://ekgf.org/dprod/spec/develop/Fulfilled	
RDF predicate	**Value**
skos:topConceptOf	dprod:DataContractLifecycleStatus

10.5.1.3 dprod:Pending

Condition not yet satisfied; not yet in force.

dprod:Pending	
https://ekgf.org/dprod/spec/develop/Pending	
RDF predicate	**Value**
rdf:type	skos:Concept
rdfs:label	pending
skos:prefLabel	pending
skos:topConceptOf	dprod:DataContractLifecycleStatus

10.5.1.4 dprod:Violated

Deadline passed without performance; breached.

dprod:Violated	
https://ekgf.org/dprod/spec/develop/Violated	
RDF predicate	**Value**
rdf:type	skos:Concept
rdfs:label	violated
skos:prefLabel	violated
skos:topConceptOf	dprod:DataContractLifecycleStatus

10.5.2 dprod:DataProductLifecycleStatus

An optional reference taxonomy for data product lifecycle statuses. Enterprises may use, extend, or replace this scheme with their own skos:ConceptScheme.

DPROD does not prescribe a closed set of data product lifecycle statuses.

dprod:DataProductLifecycleStatus	
https://ekgf.org/dprod/spec/develop/DataProductLifecycleStatus	
RDF predicate	**Value**
dct:title	DPROD Data Product Lifecycle Status
rdf:type	skos:ConceptScheme

dprod:DataProductLifecycleStatus	
https://ekgf.org/dprod/spec/develop/DataProductLifecycleStatus	
RDF predicate	**Value**
rdfs:label	data product lifecycle status
rdfs:seeAlso	https://edmcouncil.org/wp-content/uploads/2024/03/EDMWebinar_DPSIG_04-30-Presentation.pdf#page=7

10.6 SHACL node shapes

10.6.1 dprod-contracts-shapes:AgreementShape

Validates an odrl:Agreement policy as a DPROD policy: it must declare the DPROD contracts profile, contain at least one permission, prohibition or obligation, and name exactly one odrl:assigner and exactly one odrl:assignee.

dprod-contracts-shapes:AgreementShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/AgreementShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	agreement shape
sh:name	Agreement Shape
sh:node	dprod-contracts-shapes:NonEmptyClausesConstraint
sh:property	dprod-contracts-shapes:ProfilePropertyGroup
sh:property	(sh:maxCount: 1; sh:message: Agreement must have exactly one assignee.; sh:minCount: 1; sh:path: odrl:assignee)
sh:property	(sh:maxCount: 1; sh:message: Agreement must have exactly one assigner.; sh:minCount: 1; sh:path: odrl:assigner)
sh:targetClass	odrl:Agreement
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.2 dprod-contracts-shapes:AssetCollectionShape

Restricts an odrl:AssetCollection to explicit membership through odrl:partOf. DPROD does not resolve collections from odrl:source or narrow them with odrl:refinement, so neither may be used.

dprod-contracts-shapes:AssetCollectionShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/AssetCollectionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	explicit asset collection shape
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts support explicit collection membership only; collection odrl:refinement is not supported.; sh:path: odrl:refinement)

dprod-contracts-shapes:AssetCollectionShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/AssetCollectionShape	
RDF predicate	**Value**
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts support explicit collection membership only; odrl:source collection resolution is not supported.; sh:path: odrl:source)
sh:targetClass	odrl:AssetCollection
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.3 dprod-contracts-shapes:ConstraintShape

Validates an odrl:Constraint: it must have exactly one odrl:leftOperand, exactly one odrl:operator and at least one odrl:rightOperand.

dprod-contracts-shapes:ConstraintShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ConstraintShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	constraint shape
sh:name	Constraint Shape
sh:property	(sh:message: Constraint must have a rightOperand.; sh:minCount: 1; sh:path: odrl:rightOperand)
sh:property	(sh:maxCount: 1; sh:message: Constraint must have exactly one leftOperand.; sh:minCount: 1; sh:path: odrl:leftOperand)
sh:property	(sh:maxCount: 1; sh:message: Constraint must have exactly one operator.; sh:minCount: 1; sh:path: odrl:operator)
sh:targetClass	odrl:Constraint
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.4 dprod-contracts-shapes:DataContractShape

Validates a dprod:DataContract: it must declare the DPROD contracts profile, contain at least one permission, prohibition or obligation, name exactly one odrl:assigner (the data provider) and one odrl:assignee (the data consumer), and accept exactly one dprod:DataOffer through dprod:acceptsOffer. It may carry one dprod:contractLifecycleStatus and single xsd:dateTime values for dprod:effectiveDate and dprod:expirationDate.

dprod-contracts-shapes:DataContractShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DataContractShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	data contract shape
sh:name	Data Contract Shape

dprod-contracts-shapes:DataContractShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DataContractShape	
RDF predicate	**Value**
sh:node	dprod-contracts-shapes:NonEmptyClausesConstraint
sh:property	dprod-contracts-shapes:ContractLifecycleStatusPropertyGroup
sh:property	dprod-contracts-shapes:ProfilePropertyGroup
sh:property	(sh:maxCount: 1; sh:message: DataContract must have exactly one assignee (data consumer).; sh:minCount: 1; sh:path: odrl:assignee)
sh:property	(sh:maxCount: 1; sh:message: DataContract must have exactly one assigner (data provider).; sh:minCount: 1; sh:path: odrl:assigner)
sh:property	(sh:class: dprod:DataOffer; sh:maxCount: 1; sh:message: DataContract must reference exactly one DataOffer.; sh:minCount: 1; sh:path: dprod:acceptsOffer)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:message: effectiveDate, if present, must be a single xsd:dateTime value.; sh:path: dprod:effectiveDate)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:message: expirationDate, if present, must be a single xsd:dateTime value.; sh:path: dprod:expirationDate)
sh:targetClass	dprod:DataContract
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.5 dprod-contracts-shapes:DataOfferShape

Validates a dprod:DataOffer: it must declare the DPROD contracts profile, contain at least one permission, prohibition or obligation, and name exactly one odrl:assigner, the data provider. It may carry one dprod:offerLifecycleStatus and single xsd:dateTime values for dprod:effectiveDate and dprod:expirationDate.

dprod-contracts-shapes:DataOfferShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DataOfferShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	data offer shape
sh:name	Data Offer Shape
sh:node	dprod-contracts-shapes:NonEmptyClausesConstraint
sh:property	dprod-contracts-shapes:OfferLifecycleStatusPropertyGroup
sh:property	dprod-contracts-shapes:ProfilePropertyGroup
sh:property	(sh:maxCount: 1; sh:message: DataOffer must have exactly one assigner.; sh:minCount: 1; sh:path: odrl:assigner)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:message: effectiveDate, if present, must be a single xsd:dateTime value.; sh:path: dprod:effectiveDate)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:message: expirationDate, if present, must be a single xsd:dateTime value.; sh:path: dprod:expirationDate)

dprod-contracts-shapes:DataOfferShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DataOfferShape	
RDF predicate	**Value**
sh:targetClass	dprod:DataOffer
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.6 dprod-contracts-shapes:DutyOnlyPropertiesShape

Ensures that dprod:deadline, dprod:subjectOfDuty and dprod:objectOfDuty are used only on duties. Their subject must be an odrl:Duty and must not also be an odrl:Permission or odrl:Prohibition, so OWL domain inference cannot silently give a rule a second, ODRL-disjoint rule type.

dprod-contracts-shapes:DutyOnlyPropertiesShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DutyOnlyPropertiesShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	duty-only properties shape
sh:class	odrl:Duty
sh:message	dprod:deadline, dprod:subjectOfDuty and dprod:objectOfDuty are duty-only: the subject must be an odrl:Duty and not an odrl:Permission or odrl:Prohibition.
sh:name	Duty-Only Properties Shape
sh:not	(sh:class: odrl:Permission)
sh:not	(sh:class: odrl:Prohibition)
sh:targetSubjectsOf	dprod:deadline
sh:targetSubjectsOf	dprod:objectOfDuty
sh:targetSubjectsOf	dprod:subjectOfDuty
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.7 dprod-contracts-shapes:DutyShape

Validates an odrl:Duty: it must have exactly one odrl:action and may carry at most one each of dprod:deadline (an xsd:dateTime or xsd:duration), dprod:schedule (an identified dprod:Schedule), dprod:subjectOfDuty and dprod:objectOfDuty (each an odrl:Party), and dprod:dutyState (a skos:Concept).

dprod-contracts-shapes:DutyShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DutyShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	duty shape
sh:name	Duty Shape

dprod-contracts-shapes:DutyShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DutyShape	
RDF predicate	**Value**
sh:property	dprod-contracts-shapes:DutyStatePropertyGroup
sh:property	(sh:maxCount: 1; sh:message: Deadline must be xsd:dateTime or xsd:duration.; sh:or: ((sh:datatype: xsd:dateTime), (sh:datatype: xsd:duration)); sh:path: dprod:deadline)
sh:property	(sh:maxCount: 1; sh:message: Duty must have exactly one action.; sh:minCount: 1; sh:path: odrl:action)
sh:property	(sh:class: odrl:Party; sh:maxCount: 1; sh:message: Duty object must be a single odrl:Party.; sh:path: dprod:objectOfDuty)
sh:property	(sh:class: dprod:Schedule; sh:maxCount: 1; sh:message: Duty schedule must be a single IRI identifying a dprod:Schedule.; sh:nodeKind: sh:IRI; sh:path: dprod:schedule)
sh:property	(sh:class: odrl:Party; sh:maxCount: 1; sh:message: Duty subject must be a single odrl:Party.; sh:path: dprod:subjectOfDuty)
sh:targetClass	odrl:Duty
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.8 dprod-contracts-shapes:EvaluationContextShape

Validates a dprod:EvaluationContext, the input to policy evaluation: it must have exactly one dprod:request graph root, exactly one dprod:state snapshot typed as prov:Entity, exactly one dprod:agent identified by an IRI and typed as odrl:Party, and exactly one dprod:clock xsd:dateTime.

dprod-contracts-shapes:EvaluationContextShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/EvaluationContextShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	evaluation context shape
sh:property	(sh:class: odrl:Party; sh:maxCount: 1; sh:message: An evaluation context must have exactly one dprod:agent typed as odrl:Party.; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: dprod:agent)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:message: An evaluation context must have exactly one dprod:clock xsd:dateTime.; sh:minCount: 1; sh:path: dprod:clock)
sh:property	(sh:maxCount: 1; sh:message: An evaluation context must have exactly one dprod:request graph root.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: dprod:request)
sh:property	(sh:class: prov:Entity; sh:maxCount: 1; sh:message: An evaluation context must have exactly one immutable dprod:state snapshot, typed as prov:Entity.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: dprod:state)
sh:targetClass	dprod:EvaluationContext

dprod-contracts-shapes:EvaluationContextShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/EvaluationContextShape	
RDF predicate	**Value**
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.9 dprod-contracts-shapes:LeftOperandShape

Validates every odrl:LeftOperand, whether declared as such or used as the object of odrl:leftOperand. Apart from the built-in operands dprod:currentAgent and odrl:dateTime, each must bind exactly one dprod:operandSource (requestSource, stateSource or contextSource) and exactly one IRI-valued dprod:operandProperty, so the operand resolves to a value in the evaluation context.

dprod-contracts-shapes:LeftOperandShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/LeftOperandShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	left operand shape
sh:message	Every left operand must declare exactly one dprod:operandProperty, which must be IRI-valued.
sh:message	Every left operand must declare exactly one dprod:operandSource: requestSource, stateSource, or contextSource.
sh:name	Left Operand Shape
sh:or	((sh:in: (dprod:currentAgent, odrl:dateTime)), (sh:property: (sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: dprod:operandProperty)))
sh:or	((sh:in: (dprod:currentAgent, odrl:dateTime)), (sh:property: (sh:in: (dprod:requestSource, dprod:stateSource, dprod:contextSource); sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: dprod:operandSource)))
sh:targetClass	odrl:LeftOperand
sh:targetObjectsOf	odrl:leftOperand
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.10 dprod-contracts-shapes:LogicalConstraintShape

Validates an odrl:LogicalConstraint: it must use exactly one of the logical operators odrl:and, odrl:or or dprod:not, each with a single operand list.

dprod-contracts-shapes:LogicalConstraintShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/LogicalConstraintShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	logical constraint shape

dprod-contracts-shapes:LogicalConstraintShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/LogicalConstraintShape	
RDF predicate	**Value**
sh:message	LogicalConstraint must have exactly one of: odrl:and, odrl:or, or dprod:not.
sh:name	Logical Constraint Shape
sh:targetClass	odrl:LogicalConstraint
sh:xone	((sh:property: (sh:maxCount: 1; sh:minCount: 1; sh:path: odrl:and)), (sh:property: (sh:maxCount: 1; sh:minCount: 1; sh:path: odrl:or)), (sh:property: (sh:maxCount: 1; sh:minCount: 1; sh:path: dprod:not)))
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.11 dprod-contracts-shapes:NonEmptyClausesConstraint

Reusable node constraint requiring a DPROD policy to carry at least one rule: an odrl:permission, odrl:prohibition or odrl:obligation. Set, Offer, Agreement, DataOffer and DataContract shapes apply it through sh:node.

dprod-contracts-shapes:NonEmptyClausesConstraint	
https://ekgf.org/dprod/spec/develop/contracts/shapes/NonEmptyClausesConstraint	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	non-empty clauses constraint
sh:message	Policy must have at least one permission, prohibition, or obligation.
sh:or	((sh:property: (sh:minCount: 1; sh:path: odrl:permission)), (sh:property: (sh:minCount: 1; sh:path: odrl:prohibition)), (sh:property: (sh:minCount: 1; sh:path: odrl:obligation)))
rdfs:isDefinedBy	Not declared in collected sources.

10.6.12 dprod-contracts-shapes:OfferShape

Validates an odrl:Offer policy as a DPROD policy: it must declare the DPROD contracts profile, contain at least one permission, prohibition or obligation, and name exactly one odrl:assigner, the party making the offer.

dprod-contracts-shapes:OfferShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/OfferShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	offer shape
sh:name	Offer Shape
sh:node	dprod-contracts-shapes:NonEmptyClausesConstraint
sh:property	dprod-contracts-shapes:ProfilePropertyGroup

dprod-contracts-shapes:OfferShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/OfferShape	
RDF predicate	**Value**
sh:property	(sh:maxCount: 1; sh:message: Offer must have exactly one assigner.; sh:minCount: 1; sh:path: odrl:assigner)
sh:targetClass	odrl:Offer
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.13 dprod-contracts-shapes:PartOfShape

Validates ODRL collection membership: odrl:partOf must link an odrl:Asset or odrl:AssetCollection to an odrl:AssetCollection, or an odrl:Party or odrl:PartyCollection to an odrl:PartyCollection.

dprod-contracts-shapes:PartOfShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/PartOfShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	ODRL collection membership shape
sh:message	odrl:partOf must link an odrl:Asset to an odrl:AssetCollection or an odrl:Party to an odrl:PartyCollection.
sh:or	((sh:class: odrl:Asset; sh:property: (sh:class: odrl:AssetCollection; sh:path: odrl:partOf)), (sh:class: odrl:AssetCollection; sh:property: (sh:class: odrl:AssetCollection; sh:path: odrl:partOf)), (sh:class: odrl:Party; sh:property: (sh:class: odrl:PartyCollection; sh:path: odrl:partOf)), (sh:class: odrl:PartyCollection; sh:property: (sh:class: odrl:PartyCollection; sh:path: odrl:partOf)))
sh:targetSubjectsOf	odrl:partOf
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.14 dprod-contracts-shapes:PartyCollectionShape

Restricts an odrl:PartyCollection to explicit membership through odrl:partOf. DPROD does not resolve collections from odrl:source or narrow them with odrl:refinement, so neither may be used.

dprod-contracts-shapes:PartyCollectionShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/PartyCollectionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	explicit party collection shape
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts support explicit collection membership only; collection odrl:refinement is not supported.; sh:path: odrl:refinement)

dprod-contracts-shapes:PartyCollectionShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/PartyCollectionShape	
RDF predicate	**Value**
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts support explicit collection membership only; odrl:source collection resolution is not supported.; sh:path: odrl:source)
sh:targetClass	odrl:PartyCollection
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.15 dprod-contracts-shapes:PermissionShape

Validates an odrl:Permission rule: it must have exactly one odrl:action and at most one odrl:target. When the target is absent, the permission inherits the policy-level target.

dprod-contracts-shapes:PermissionShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/PermissionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	permission shape
sh:name	Permission Shape
sh:property	(sh:maxCount: 1; sh:message: Permission may have at most one target (inherits from policy if absent).; sh:path: odrl:target)
sh:property	(sh:maxCount: 1; sh:message: Permission must have exactly one action.; sh:minCount: 1; sh:path: odrl:action)
sh:targetClass	odrl:Permission
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.16 dprod-contracts-shapes:PolicyShape

Validates every odrl:Policy, whatever its subclass, by requiring it to declare odrl:profile with the DPROD contracts profile IRI, so that only policies that opt into DPROD semantics are evaluated by DPROD.

dprod-contracts-shapes:PolicyShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/PolicyShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	policy shape
sh:name	Policy Shape
sh:property	(sh:hasValue: dprod:contracts/profile/; sh:message: DPROD policies must declare odrl:profile <https://ekgf.org/dprod/spec/develop/contracts/profile/>.; sh:minCount: 1; sh:path: odrl:profile)
sh:targetClass	odrl:Policy

dprod-contracts-shapes:PolicyShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/PolicyShape	
RDF predicate	**Value**
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.17 dprod-contracts-shapes:ProhibitionShape

Validates an odrl:Prohibition rule: it must have exactly one odrl:action and at most one odrl:target. When the target is absent, the prohibition inherits the policy-level target.

dprod-contracts-shapes:ProhibitionShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ProhibitionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	prohibition shape
sh:name	Prohibition Shape
sh:property	(sh:maxCount: 1; sh:message: Prohibition may have at most one target (inherits from policy if absent).; sh:path: odrl:target)
sh:property	(sh:maxCount: 1; sh:message: Prohibition must have exactly one action.; sh:minCount: 1; sh:path: odrl:action)
sh:targetClass	odrl:Prohibition
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.18 dprod-contracts-shapes:RejectAnonymousScheduleShape

Requires every dprod:Schedule to be identified by an IRI, so a schedule can be referenced, shared and audited. Kept separate from the schedule shape so its message applies only to blank-node schedules.

dprod-contracts-shapes:RejectAnonymousScheduleShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectAnonymousScheduleShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject anonymous schedule
sh:message	A Schedule must have an IRI identifier.
sh:nodeKind	sh:IRI
sh:targetClass	dprod:Schedule
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.19 dprod-contracts-shapes:RejectConsequenceShape

Rejects odrl:consequence, which DPROD contracts do not support: DPROD defines no consequence semantics for unfulfilled duties.

dprod-contracts-shapes:RejectConsequenceShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectConsequenceShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject consequence
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts do not support odrl:consequence.; sh:path: odrl:consequence; sh:severity: sh:Violation)
sh:targetSubjectsOf	odrl:consequence
rdfs:isDefinedBy	Not declared in collected sources.

10.6.20 dprod-contracts-shapes:RejectInheritAllowedShape

Rejects odrl:inheritAllowed, which DPROD contracts do not support: DPROD defines no policy inheritance.

dprod-contracts-shapes:RejectInheritAllowedShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectInheritAllowedShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject inheritAllowed
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts do not support odrl:inheritAllowed.; sh:path: odrl:inheritAllowed; sh:severity: sh:Violation)
sh:targetSubjectsOf	odrl:inheritAllowed
rdfs:isDefinedBy	Not declared in collected sources.

10.6.21 dprod-contracts-shapes:RejectInheritFromShape

Rejects odrl:inheritFrom, which DPROD contracts do not support: DPROD defines no policy inheritance.

dprod-contracts-shapes:RejectInheritFromShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectInheritFromShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject inheritFrom
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts do not support odrl:inheritFrom.; sh:path: odrl:inheritFrom; sh:severity: sh:Violation)
sh:targetSubjectsOf	odrl:inheritFrom
rdfs:isDefinedBy	Not declared in collected sources.

10.6.22 dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataContractShape

Keeps lifecycle statuses separate: a dprod:DataContract uses only dprod:contractLifecycleStatus and must not carry dprod:dataProductLifecycleStatus or dprod:offerLifecycleStatus.

dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataContractShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectMixedLifecycleStatusOnDataContractShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject mixed lifecycle status on data contract shape
sh:property	(sh:maxCount: 0; sh:message: DataContract must not use dprod:dataProductLifecycleStatus.; sh:path: dprod:dataProductLifecycleStatus)
sh:property	(sh:maxCount: 0; sh:message: DataContract must not use dprod:offerLifecycleStatus.; sh:path: dprod:offerLifecycleStatus)
sh:targetClass	dprod:DataContract
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.23 dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataOfferShape

Keeps lifecycle statuses separate: a dprod:DataOffer uses only dprod:offerLifecycleStatus and must not carry dprod:dataProductLifecycleStatus or dprod:contractLifecycleStatus.

dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataOfferShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectMixedLifecycleStatusOnDataOfferShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject mixed lifecycle status on data offer shape
sh:property	(sh:maxCount: 0; sh:message: DataOffer must not use dprod:contractLifecycleStatus.; sh:path: dprod:contractLifecycleStatus)
sh:property	(sh:maxCount: 0; sh:message: DataOffer must not use dprod:dataProductLifecycleStatus.; sh:path: dprod:dataProductLifecycleStatus)
sh:targetClass	dprod:DataOffer
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.24 dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataProductShape

Keeps lifecycle statuses separate: a dprod:DataProduct uses only dprod:dataProductLifecycleStatus and must not carry dprod:offerLifecycleStatus or dprod:contractLifecycleStatus.

dprod-contracts-shapes:RejectMixedLifecycleStatusOnDataProductShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectMixedLifecycleStatusOnDataProductShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	data product lifecycle status separation shape
sh:property	(sh:maxCount: 0; sh:message: DataProduct must not use dprod:contractLifecycleStatus.; sh:path: dprod:contractLifecycleStatus)
sh:property	(sh:maxCount: 0; sh:message: DataProduct must not use dprod:offerLifecycleStatus.; sh:path: dprod:offerLifecycleStatus)
sh:targetClass	dprod:DataProduct
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.25 dprod-contracts-shapes:RejectObsoleteContractLifecycleStatusSpellingShape

Rejects the obsolete spelling dprod:contractLifeCycleStatus. A contract's lifecycle status uses dprod:contractLifecycleStatus.

dprod-contracts-shapes:RejectObsoleteContractLifecycleStatusSpellingShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectObsoleteContractLifecycleStatusSpellingShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject obsolete contract lifecycle status spelling
sh:property	(sh:maxCount: 0; sh:message: dprod:contractLifeCycleStatus is obsolete; use dprod:contractLifecycleStatus.; sh:path: dprod:contractLifeCycleStatus; sh:severity: sh:Violation)
sh:targetSubjectsOf	dprod:contractLifeCycleStatus
rdfs:isDefinedBy	Not declared in collected sources.

10.6.26 dprod-contracts-shapes:RejectObsoleteDprodMemberOfShape

Rejects the obsolete dprod:memberOf property. Collection membership uses odrl:partOf.

dprod-contracts-shapes:RejectObsoleteDprodMemberOfShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectObsoleteDprodMemberOfShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject obsolete dprod:memberOf

dprod-contracts-shapes:RejectObsoleteDprodMemberOfShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectObsoleteDprodMemberOfShape	
RDF predicate	**Value**
sh:property	(sh:maxCount: 0; sh:message: dprod:memberOf is obsolete; use odrl:partOf.; sh:path: dprod:memberOf; sh:severity: sh:Violation)
sh:targetSubjectsOf	dprod:memberOf
rdfs:isDefinedBy	Not declared in collected sources.

10.6.27 dprod-contracts-shapes:RejectObsoleteDprodPartOfShape

Rejects the obsolete dprod:partOf property. Collection membership uses odrl:partOf.

dprod-contracts-shapes:RejectObsoleteDprodPartOfShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectObsoleteDprodPartOfShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject obsolete dprod:partOf
sh:property	(sh:maxCount: 0; sh:message: dprod:partOf is obsolete; use odrl:partOf.; sh:path: dprod:partOf; sh:severity: sh:Violation)
sh:targetSubjectsOf	dprod:partOf
rdfs:isDefinedBy	Not declared in collected sources.

10.6.28 dprod-contracts-shapes:RejectObsoleteOfferLifecycleStatusSpellingShape

Rejects the obsolete spelling dprod:offerLifeCycleStatus. An offer's lifecycle status uses dprod:offerLifecycleStatus.

dprod-contracts-shapes:RejectObsoleteOfferLifecycleStatusSpellingShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectObsoleteOfferLifecycleStatusSpellingShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject obsolete offer lifecycle status spelling
sh:property	(sh:maxCount: 0; sh:message: dprod:offerLifeCycleStatus is obsolete; use dprod:offerLifecycleStatus.; sh:path: dprod:offerLifeCycleStatus; sh:severity: sh:Violation)
sh:targetSubjectsOf	dprod:offerLifeCycleStatus
rdfs:isDefinedBy	Not declared in collected sources.

10.6.29 dprod-contracts-shapes:RejectObsoleteRecurrenceShape

Rejects the obsolete dprod:recurrence property on any subject. Recurring obligations are expressed with dprod:schedule referencing an identified dprod:Schedule.

dprod-contracts-shapes:RejectObsoleteRecurrenceShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectObsoleteRecurrenceShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject obsolete recurrence property
sh:property	(sh:maxCount: 0; sh:message: dprod:recurrence is obsolete; use dprod:schedule with an identified dprod:Schedule.; sh:path: dprod:recurrence; sh:severity: sh:Violation)
sh:targetSubjectsOf	dprod:recurrence
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.30 dprod-contracts-shapes:RejectOperandPathShape

Rejects the obsolete dprod:path property. An operand is bound to a single dprod:operandSource and dprod:operandProperty against a normalized source graph, not to a property path.

dprod-contracts-shapes:RejectOperandPathShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectOperandPathShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	obsolete operand path rejection shape
sh:property	(sh:maxCount: 0; sh:message: dprod:path is obsolete; normalize the source graph and use dprod:operandSource plus dprod:operandProperty.; sh:path: dprod:path; sh:severity: sh:Violation)
sh:targetSubjectsOf	dprod:path
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.31 dprod-contracts-shapes:RejectRemedyShape

Rejects odrl:remedy, which DPROD contracts do not support: DPROD defines no remedy semantics for violated prohibitions.

dprod-contracts-shapes:RejectRemedyShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectRemedyShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject remedy
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts do not support odrl:remedy.; sh:path: odrl:remedy; sh:severity: sh:Violation)
sh:targetSubjectsOf	odrl:remedy

dprod-contracts-shapes:RejectRemedyShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectRemedyShape	
RDF predicate	**Value**
rdfs:isDefinedBy	Not declared in collected sources.

10.6.32 dprod-contracts-shapes:RejectRequestShape

Rejects odrl:Request policies. DPROD does not yet define Offer-Request-Agreement semantics, so support for requests is deferred.

dprod-contracts-shapes:RejectRequestShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectRequestShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject request policy
sh:sparql	(sh:message: DPROD contracts reject odrl:Request because Offer-Request-Agreement semantics are undefined; support is deferred.; sh:select: SPARQL SELECT query 1; sh:severity: sh:Violation)
sh:targetClass	odrl:Request
rdfs:isDefinedBy	Not declared in collected sources.

10.6.32.1 SPARQL query 1

DPROD contracts reject odrl:Request because Offer-Request-Agreement semantics are undefined; support is deferred.

```
SELECT $this WHERE { $this a odrl:Request }
```

10.6.33 dprod-contracts-shapes:RejectRuntimeReferenceShape

Rejects the obsolete dprod:RuntimeReference class. A runtime value is declared as an odrl:LeftOperand with one dprod:operandSource and one dprod:operandProperty binding.

dprod-contracts-shapes:RejectRuntimeReferenceShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectRuntimeReferenceShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	obsolete runtime reference rejection shape
sh:property	(sh:maxCount: 0; sh:message: dprod:RuntimeReference is obsolete; declare an odrl:LeftOperand with one source/property binding.; sh:path: rdf:type; sh:severity: sh:Violation)
sh:targetClass	dprod:RuntimeReference
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.34 dprod-contracts-shapes:RejectTicketShape

Rejects odrl:Ticket policies, which DPROD contracts do not support.

dprod-contracts-shapes:RejectTicketShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectTicketShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject ticket policy
sh:sparql	(sh:message: DPROD contracts do not support odrl:Ticket.; sh:select: SPARQL SELECT query 1; sh:severity: sh:Violation)
sh:targetClass	odrl:Ticket
rdfs:isDefinedBy	Not declared in collected sources.

10.6.34.1 SPARQL query 1

DPROD contracts do not support odrl:Ticket.

```
SELECT $this WHERE { $this a odrl:Ticket }
```

10.6.35 dprod-contracts-shapes:RejectXoneShape

Rejects the ODRL exclusive-or logical operator odrl:xone, which DPROD contracts do not support. A logical constraint uses odrl:and, odrl:or or dprod:not.

dprod-contracts-shapes:RejectXoneShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/RejectXoneShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	reject xone operator
sh:property	(sh:maxCount: 0; sh:message: DPROD contracts do not support xone operator.; sh:path: odrl:xone; sh:severity: sh:Violation)
sh:targetSubjectsOf	odrl:xone
rdfs:isDefinedBy	Not declared in collected sources.

10.6.36 dprod-contracts-shapes:ScheduleFormatShape

Validates a dprod:ScheduleFormat, including third-party formats: it must state exactly once, as an xsd:boolean dprod:carriesTimeZone, whether its expressions embed a timezone. The schedule shape relies on this to enforce timezone completeness.

dprod-contracts-shapes:ScheduleFormatShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ScheduleFormatShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	schedule format shape
sh:property	(sh:datatype: xsd:boolean; sh:maxCount: 1; sh:message: A ScheduleFormat must declare dprod:carriesTimeZone so validators can enforce timezone completeness.; sh:minCount: 1; sh:path: dprod:carriesTimeZone)
sh:targetClass	dprod:ScheduleFormat
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.37 dprod-contracts-shapes:ScheduleReferenceShape

Validates every use of dprod:schedule: each value must be an IRI that identifies a dprod:Schedule, never a blank node or literal.

dprod-contracts-shapes:ScheduleReferenceShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ScheduleReferenceShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	schedule reference shape
sh:property	(sh:class: dprod:Schedule; sh:message: Schedule references must be IRIs identifying dprod:Schedule resources.; sh:nodeKind: sh:IRI; sh:path: dprod:schedule)
sh:targetSubjectsOf	dprod:schedule
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.38 dprod-contracts-shapes:ScheduleShape

Validates a dprod:Schedule: it must conform to exactly one identified dprod:ScheduleFormat through dct:conformsTo and carry exactly one dprod:scheduleExpression string, with an optional IANA dprod:scheduleTimeZone. RFC 5545 expressions must contain DTSTART with a TZID and an RRULE, and a declared timezone must match that TZID. POSIX crontab expressions must have exactly five fields. A schedule whose format carries no timezone must declare one.

dprod-contracts-shapes:ScheduleShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ScheduleShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	schedule shape
sh:property	(sh:maxCount: 1; sh:message: Schedule format must be an identified dprod:ScheduleFormat.; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:or: ((sh:in:

dprod-contracts-shapes:ScheduleShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ScheduleShape	
RDF predicate	**Value**
	(dprod:Rfc5545ScheduleFormat, dprod:PosixCrontabScheduleFormat)), (sh:class: dprod:ScheduleFormat)); sh:path: dct:conformsTo)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: Schedule must have exactly one authoritative expression as an xsd:string.; sh:minCount: 1; sh:path: dprod:scheduleExpression)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: scheduleTimeZone must be one IANA timezone name.; sh:path: dprod:scheduleTimeZone; sh:pattern: ^[A-Za-z0-9._+-]+(/[A-Za-z0-9._+-]+)*$)
sh:sparql	(sh:message: POSIX crontab schedules must contain exactly five time fields.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: RFC 5545 schedules must include DTSTART with TZID and RRULE (optionally EXDATE/RDATE lines); RRULE parts must be standard and must not repeat FREQ.; sh:select: SPARQL SELECT query 2)
sh:sparql	(sh:message: Schedules whose format carries no timezone must declare one IANA timezone.; sh:select: SPARQL SELECT query 3)
sh:sparql	(sh:message: scheduleTimeZone must match the TZID embedded in the RFC 5545 expression.; sh:select: SPARQL SELECT query 4)
sh:targetClass	dprod:Schedule
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.38.1 SPARQL query 1

POSIX crontab schedules must contain exactly five time fields.

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dprod: <https://ekgf.org/dprod/spec/develop/>
SELECT $this
WHERE {
  $this dct:conformsTo dprod:PosixCrontabScheduleFormat ;
        dprod:scheduleExpression ?expression .
  FILTER (
    !REGEX(STR(?expression), "^([^ \t\r\n]+[ \t]+){4}[^ \t\r\n]+$") ||
    CONTAINS(STR(?expression), "\n") ||
    CONTAINS(STR(?expression), "\r")
  )
}
```

10.6.38.2 SPARQL query 2

RFC 5545 schedules must include DTSTART with TZID and RRULE (optionally EXDATE/RDATE lines); RRULE parts must be standard and must not repeat FREQ.

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dprod: <https://ekgf.org/dprod/spec/develop/>
SELECT $this
WHERE {
```

```
  $this dct:conformsTo dprod:Rfc5545ScheduleFormat ;
        dprod:scheduleExpression ?expression .
  FILTER (!REGEX(
    STR(?expression),
    "^DTSTART;TZID=[A-Za-z0-9._+-]+(/[A-Za-z0-9._+-]+)*:[0-9]{8}T[0-9]{6}(\r?\n(EXDATE|RDA
TE)(;[A-Za-z0-9-]+=[^:;\r\n]+)*:[^\r\n]+)*\r?\nRRULE:FREQ=(SECONDLY|MINUTELY|HOURLY|DAILY|
WEEKLY|MONTHLY|YEARLY)(;(UNTIL|COUNT|INTERVAL|BYSECOND|BYMINUTE|BYHOUR|BYDAY|BYMONTHDAY|BY
YEARDAY|BYWEEKNO|BYMONTH|BYSETPOS|WKST)=[^;\r\n]+)*(\r?\n(EXDATE|RDATE)(;[A-Za-z0-9-]+=[^:
;\r\n]+)*:[^\r\n]+)*$"
  ))
}
```

10.6.38.3 SPARQL query 3

Schedules whose format carries no timezone must declare one IANA timezone.

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dprod: <https://ekgf.org/dprod/spec/develop/>
SELECT $this
WHERE {
  $this dct:conformsTo ?format .
  FILTER (?format = dprod:PosixCrontabScheduleFormat ||
          EXISTS { ?format dprod:carriesTimeZone false })
  FILTER NOT EXISTS { $this dprod:scheduleTimeZone ?timeZone }
}
```

10.6.38.4 SPARQL query 4

scheduleTimeZone must match the TZID embedded in the RFC 5545 expression.

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dprod: <https://ekgf.org/dprod/spec/develop/>
SELECT $this
WHERE {
  $this dct:conformsTo dprod:Rfc5545ScheduleFormat ;
        dprod:scheduleExpression ?expression ;
        dprod:scheduleTimeZone ?timeZone .
  BIND (REPLACE(STR(?expression), "^DTSTART;TZID=([^:]+):(.|\n|\r)*$", "$1") AS ?embedded)
  FILTER (STR(?timeZone) != ?embedded)
}
```

10.6.39 dprod-contracts-shapes:SetShape

Validates an odrl:Set policy as a DPROD policy: it must declare the DPROD contracts profile and contain at least one permission, prohibition or obligation. A policy-level odrl:target is allowed and is inherited by the policy's rules.

dprod-contracts-shapes:SetShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/SetShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	set shape
sh:name	Set Shape

dprod-contracts-shapes:SetShape	
https://ekgf.org/dprod/spec/develop/contracts/shapes/SetShape	
RDF predicate	**Value**
sh:node	dprod-contracts-shapes:NonEmptyClausesConstraint
sh:property	dprod-contracts-shapes:ProfilePropertyGroup
sh:property	(sh:message: Set policy may declare a policy-level target (inherited by rules).; sh:path: odrl:target)
sh:targetClass	odrl:Set
rdfs:isDefinedBy	dprod-contracts-shapes:

10.6.40 dprod-shapes:DataProductShape

A rational, managed, and governed collection of data, with purpose, value and ownership, meeting consumer needs over a planned life-cycle.

A data product may have input and output ports, code and metadata

dprod-shapes:DataProductShape	
https://ekgf.org/dprod/spec/develop/shapes/DataProductShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	data product shape
sh:property	dprod-shapes:DataProduct-dataProductLifecycleStatus
sh:property	dprod-shapes:DataProduct-dataProductOwner
sh:property	dprod-shapes:DataProduct-description
sh:property	dprod-shapes:DataProduct-domain
sh:property	dprod-shapes:DataProduct-hasPolicy
sh:property	dprod-shapes:DataProduct-inputDataset
sh:property	dprod-shapes:DataProduct-inputPort
sh:property	dprod-shapes:DataProduct-label
sh:property	dprod-shapes:DataProduct-outputDataset
sh:property	dprod-shapes:DataProduct-outputPort
sh:property	dprod-shapes:DataProduct-purpose
sh:targetClass	dprod:DataProduct
rdfs:isDefinedBy	dprod-shapes:

10.6.41 dprod-shapes:DataServiceShape

A collection of operations that provides access to one or more datasets or data processing functions.

dprod-shapes:DataServiceShape	
https://ekgf.org/dprod/spec/develop/shapes/DataServiceShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	data service shape
sh:property	dprod-shapes:DataService-endpointDescription
sh:property	dprod-shapes:DataService-endpointURL
sh:property	dprod-shapes:DataService-isAccessServiceOf
sh:property	dprod-shapes:DataService-protocol
sh:property	dprod-shapes:DataService-securitySchemaType
sh:targetClass	dcat:DataService
rdfs:isDefinedBy	dprod-shapes:

10.6.42 dprod-shapes:DatasetShape

A collection of data, published or curated by a single source, and available for access or download in one or more representations.

dprod-shapes:DatasetShape	
https://ekgf.org/dprod/spec/develop/shapes/DatasetShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	dataset shape
sh:property	dcat:Dataset-informationSensitivityClassification
sh:property	dprod-shapes:Dataset-conformsTo
sh:property	dprod-shapes:Dataset-description
sh:property	dprod-shapes:Dataset-distribution
sh:property	dprod-shapes:Dataset-hasPolicy
sh:property	dprod-shapes:Dataset-label
sh:property	dprod-shapes:Dataset-type
sh:targetClass	dcat:Dataset
rdfs:isDefinedBy	dprod-shapes:

10.6.43 dprod-shapes:DistributionShape

A specific representation of a dataset. A dataset might be available in multiple serializations that may differ in various ways, including natural language, media-type or format, schematic organization, temporal and spatial resolution, level of detail or profiles (which might specify any or all of the above).

dprod-shapes:DistributionShape	
https://ekgf.org/dprod/spec/develop/shapes/DistributionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	distribution shape
sh:property	dprod-shapes:Distribution-accessService
sh:property	dprod-shapes:Distribution-conformsTo
sh:property	dprod-shapes:Distribution-format
sh:property	dprod-shapes:Distribution-isDistributionOf
sh:targetClass	dcat:Distribution
rdfs:isDefinedBy	dprod-shapes:

10.6.44 dprod-shapes:InformationSensitivityClassificationShape

A classification of the information within a dataset to indicate the level of control and protection that must be applied.

dprod-shapes:InformationSensitivityClassificationShape	
https://ekgf.org/dprod/spec/develop/shapes/InformationSensitivityClassificationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	information sensitivity classification shape
sh:targetClass	dprod:InformationSensitivityClassification
rdfs:isDefinedBy	dprod-shapes:

10.6.45 dprod-shapes:SecuritySchemaTypeShape

A classification encompassing a set of rules used for authentication and communication.

dprod-shapes:SecuritySchemaTypeShape	
https://ekgf.org/dprod/spec/develop/shapes/SecuritySchemaTypeShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	security schema type shape
sh:targetClass	dprod:SecuritySchemaType
rdfs:isDefinedBy	dprod-shapes:

10.7 SHACL property shapes

10.7.1 dcat:Dataset-informationSensitivityClassification

More granular classification that indicates the level of control and protection that must be applied to the asset due to the nature of the data and its sensitivity or importance to the organization.

dcat:Dataset-informationSensitivityClassification	
http://www.w3.org/ns/dcat#Dataset-informationSensitivityClassification	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset information sensitivity classification shape
sh:class	dprod:InformationSensitivityClassification
sh:path	dprod:informationSensitivityClassification
rdfs:isDefinedBy	dprod-shapes:

10.7.2 dprod-contracts-shapes:ContractLifecycleStatusPropertyGroup

Optional authored lifecycle status for a DataContract. The value must be a skos:Concept; DPROD's scheme is optional and profiles may use an enterprise taxonomy.

dprod-contracts-shapes:ContractLifecycleStatusPropertyGroup	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ContractLifecycleStatusPropertyGroup	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
sh:class	skos:Concept
sh:maxCount	1
sh:message	contractLifecycleStatus, if present, must be a single skos:Concept value.
sh:name	contract lifecycle status
sh:nodeKind	sh:IRI
sh:path	dprod:contractLifecycleStatus
rdfs:isDefinedBy	Not declared in collected sources.

10.7.3 dprod-contracts-shapes:DutyStatePropertyGroup

Optional evaluated state for a Duty. Value must be a skos:Concept; the four DPROD-defined concepts (Pending, Active, Fulfilled, Violated) are the canonical vocabulary that the standard evaluation algorithm transitions between.

dprod-contracts-shapes:DutyStatePropertyGroup	
https://ekgf.org/dprod/spec/develop/contracts/shapes/DutyStatePropertyGroup	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
sh:class	skos:Concept
sh:maxCount	1
sh:message	dutyState, if present, must be a single skos:Concept value.
sh:name	duty state
sh:nodeKind	sh:IRI
sh:path	dprod:dutyState
rdfs:isDefinedBy	Not declared in collected sources.

10.7.4 dprod-contracts-shapes:OfferLifecycleStatusPropertyGroup

Optional authored lifecycle status for a DataOffer. The value must be a skos:Concept; DPROD's scheme is optional and profiles may use an enterprise taxonomy.

dprod-contracts-shapes:OfferLifecycleStatusPropertyGroup	
https://ekgf.org/dprod/spec/develop/contracts/shapes/OfferLifecycleStatusPropertyGroup	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
sh:class	skos:Concept
sh:maxCount	1
sh:message	offerLifecycleStatus, if present, must be a single skos:Concept value.
sh:name	offer lifecycle status
sh:nodeKind	sh:IRI
sh:path	dprod:offerLifecycleStatus
rdfs:isDefinedBy	Not declared in collected sources.

10.7.5 dprod-contracts-shapes:ProfilePropertyGroup

The ODRL profile IRI a policy must declare to be recognised as DPROD. Reused by Set, Offer, Agreement, DataOffer, and DataContract shapes.

dprod-contracts-shapes:ProfilePropertyGroup	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ProfilePropertyGroup	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
sh:hasValue	dprod:contracts/profile/

dprod-contracts-shapes:ProfilePropertyGroup	
https://ekgf.org/dprod/spec/develop/contracts/shapes/ProfilePropertyGroup	
RDF predicate	**Value**
sh:message	DPROD policies must declare odrl:profile <https://ekgf.org/dprod/spec/develop/contracts/profile/>.
sh:minCount	1
sh:name	profile
sh:path	odrl:profile
rdfs:isDefinedBy	Not declared in collected sources.

10.7.6 dprod-shapes:DataProduct-dataProductLifecycleStatus

The authored development lifecycle status of the data product.

dprod-shapes:DataProduct-dataProductLifecycleStatus	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-dataProductLifecycleStatus	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product lifecycle status shape
sh:class	skos:Concept
sh:maxCount	1
sh:message	dataProductLifecycleStatus, if present, must be a single skos:Concept value.
sh:path	dprod:dataProductLifecycleStatus
rdfs:isDefinedBy	dprod-shapes:

10.7.7 dprod-shapes:DataProduct-dataProductOwner

The agent that is accountable overall for the data product, including managing it through its lifecycle.

dprod-shapes:DataProduct-dataProductOwner	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-dataProductOwner	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product owner shape
sh:class	prov:Agent
sh:path	dprod:dataProductOwner
rdfs:isDefinedBy	dprod-shapes:

10.7.8 dprod-shapes:DataProduct-description

A free text description of the data product.

dprod-shapes:DataProduct-description	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-description	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product description shape
sh:datatype	xsd:string
sh:path	dct:description
rdfs:isDefinedBy	dprod-shapes:

10.7.9 dprod-shapes:DataProduct-domain

The business or information area supported by the data product.

The target class is left open

dprod-shapes:DataProduct-domain	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-domain	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product domain shape
sh:path	dprod:domain
rdfs:isDefinedBy	dprod-shapes:

10.7.10 dprod-shapes:DataProduct-hasPolicy

An ODRL conformant policy expressing the rights associated with the data product. This is an inferred relationship based on the rights expressed on the individual datasets of the data product.

dprod-shapes:DataProduct-hasPolicy	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-hasPolicy	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product has policy shape
sh:class	odrl:Policy
sh:path	odrl:hasPolicy
rdfs:isDefinedBy	dprod-shapes:

10.7.11 dprod-shapes:DataProduct-inputDataset

The source data made available to the data product through input data services. Depending on the lifecycle of the data product, this may be a stated or inferred relationship aligned with the input ports.

dprod-shapes:DataProduct-inputDataset	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-inputDataset	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product input dataset shape
sh:class	dcat:Dataset
sh:path	dprod:inputDataset
rdfs:isDefinedBy	dprod-shapes:

10.7.12 dprod-shapes:DataProduct-inputPort

A set of services exposed by a data product to collect its source data and makes it available for further internal transformation. An input port can receive data from one or more upstream sources in a push (i.e. asynchronous subscription) or pop mode (i.e. synchronous query). Each data product may have one or more input ports.

dprod-shapes:DataProduct-inputPort	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-inputPort	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product input port shape
sh:class	dcat:DataService
sh:path	dprod:inputPort
rdfs:isDefinedBy	dprod-shapes:

10.7.13 dprod-shapes:DataProduct-label

The name given to the data product.

dprod-shapes:DataProduct-label	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-label	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product label shape
sh:datatype	xsd:string
sh:path	rdfs:label
rdfs:isDefinedBy	dprod-shapes:

10.7.14 dprod-shapes:DataProduct-outputDataset

The data that is exposed by the data product through output data services in a way that can be understood and trusted. Depending on the lifecycle of the data product, this may be a stated or inferred relationship aligned with the output ports.

dprod-shapes:DataProduct-outputDataset	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-outputDataset	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product output dataset shape
sh:class	dcat:Dataset
sh:path	dprod:outputDataset
rdfs:isDefinedBy	dprod-shapes:

10.7.15 dprod-shapes:DataProduct-outputPort

A set of services exposed by a data product to share the generated data in a way that can be understood and trusted.

dprod-shapes:DataProduct-outputPort	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-outputPort	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product output port shape
sh:class	dcat:DataService
sh:path	dprod:outputPort
rdfs:isDefinedBy	dprod-shapes:

10.7.16 dprod-shapes:DataProduct-purpose

A description of the objectives and intended usage of the data product.

dprod-shapes:DataProduct-purpose	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-purpose	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product purpose shape
sh:datatype	xsd:string
sh:path	dprod:purpose
rdfs:isDefinedBy	dprod-shapes:

10.7.17 dprod-shapes:DataProduct-type

The nature or genre of the data product.

dprod-shapes:DataProduct-type	
https://ekgf.org/dprod/spec/develop/shapes/DataProduct-type	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data product description shape
sh:path	dct:type
rdfs:isDefinedBy	dprod-shapes:

10.7.18 dprod-shapes:DataService-endpointDescription

A description of the services available via the end-points, including their operations, parameters etc.

dprod-shapes:DataService-endpointDescription	
https://ekgf.org/dprod/spec/develop/shapes/DataService-endpointDescription	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data service endpoint description shape
sh:path	dcat:endpointDescription
rdfs:isDefinedBy	dprod-shapes:

10.7.19 dprod-shapes:DataService-endpointURL

The root location or primary endpoint of the service.

dprod-shapes:DataService-endpointURL	
https://ekgf.org/dprod/spec/develop/shapes/DataService-endpointURL	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data service endpoint URL shape
sh:path	dcat:endpointURL
rdfs:isDefinedBy	dprod-shapes:

10.7.20 dprod-shapes:DataService-isAccessServiceOf

The dataset distribution that is being offered through this data service.

dprod-shapes:DataService-isAccessServiceOf	
https://ekgf.org/dprod/spec/develop/shapes/DataService-isAccessServiceOf	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data service is access service of shape
sh:class	dcat:Distribution
sh:path	dprod:isAccessServiceOf
rdfs:isDefinedBy	dprod-shapes:

10.7.21 dprod-shapes:DataService-protocol

A protocol (possibly one of many options) used to communicate with this data service.

dprod-shapes:DataService-protocol	
https://ekgf.org/dprod/spec/develop/shapes/DataService-protocol	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data service protocol shape
sh:path	dprod:protocol
rdfs:isDefinedBy	dprod-shapes:

10.7.22 dprod-shapes:DataService-securitySchemaType

The security schema type used for authentication and communication with this Data Service.

dprod-shapes:DataService-securitySchemaType	
https://ekgf.org/dprod/spec/develop/shapes/DataService-securitySchemaType	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	data service security schema type shape
sh:class	dprod:SecuritySchemaType
sh:path	dprod:securitySchemaType
rdfs:isDefinedBy	dprod-shapes:

10.7.23 dprod-shapes:Dataset-conformsTo

A model, schema, ontology, view or profile that the dataset conforms to.

dprod-shapes:Dataset-conformsTo	
https://ekgf.org/dprod/spec/develop/shapes/Dataset-conformsTo	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset conforms to shape
sh:path	dct:conformsTo
rdfs:isDefinedBy	dprod-shapes:

10.7.24 dprod-shapes:Dataset-description

Free text description of the dataset.

dprod-shapes:Dataset-description	
https://ekgf.org/dprod/spec/develop/shapes/Dataset-description	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset distribution shape
sh:datatype	xsd:string
sh:path	dct:description
rdfs:isDefinedBy	dprod-shapes:

10.7.25 dprod-shapes:Dataset-distribution

An available distribution of the dataset.

dprod-shapes:Dataset-distribution	
https://ekgf.org/dprod/spec/develop/shapes/Dataset-distribution	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset distribution shape
sh:class	dcat:Distribution
sh:path	dcat:distribution
rdfs:isDefinedBy	dprod-shapes:

10.7.26 dprod-shapes:Dataset-hasPolicy

An ODRL conformant policy expressing the rights associated with the resource.

dprod-shapes:Dataset-hasPolicy	
https://ekgf.org/dprod/spec/develop/shapes/Dataset-hasPolicy	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset has policy shape
sh:path	odrl:hasPolicy
rdfs:isDefinedBy	dprod-shapes:

10.7.27 dprod-shapes:Dataset-label

The name given to the dataset

dprod-shapes:Dataset-label	
https://ekgf.org/dprod/spec/develop/shapes/Dataset-label	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset label shape
sh:datatype	xsd:string
sh:path	rdfs:label
rdfs:isDefinedBy	dprod-shapes:

10.7.28 dprod-shapes:Dataset-type

The type or genre of the dataset.

dprod-shapes:Dataset-type	
https://ekgf.org/dprod/spec/develop/shapes/Dataset-type	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	dataset type shape
sh:path	dct:type
rdfs:isDefinedBy	dprod-shapes:

10.7.29 dprod-shapes:Distribution-accessService

A data service that gives access to the distribution of the dataset.

dprod-shapes:Distribution-accessService	
https://ekgf.org/dprod/spec/develop/shapes/Distribution-accessService	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	distribution access service shape
sh:class	dcat:DataService
sh:path	dcat:accessService
rdfs:isDefinedBy	dprod-shapes:

10.7.30 dprod-shapes:Distribution-conformsTo

The schema that the distribution conforms to that is format and technology dependent.

dprod-shapes:Distribution-conformsTo	
https://ekgf.org/dprod/spec/develop/shapes/Distribution-conformsTo	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	distribution conforms to shape
sh:path	dct:conformsTo
rdfs:isDefinedBy	dprod-shapes:

10.7.31 dprod-shapes:Distribution-format

The file format of the distribution.

dprod-shapes:Distribution-format	
https://ekgf.org/dprod/spec/develop/shapes/Distribution-format	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	distribution format shape
sh:path	dct:format
rdfs:isDefinedBy	dprod-shapes:

10.7.32 dprod-shapes:Distribution-isDistributionOf

The dataset that this distribution makes available.

dprod-shapes:Distribution-isDistributionOf	
https://ekgf.org/dprod/spec/develop/shapes/Distribution-isDistributionOf	
RDF predicate	**Value**
rdf:type	sh:PropertyShape
rdfs:label	distribution is distribution of shape
sh:class	dcat:Dataset
sh:path	dprod:isDistributionOf
rdfs:isDefinedBy	dprod-shapes:

11 SBA Pool Rates

Rates for SBA Pool that are Mortgage Backed Securities.

The data product is provided through 3 ports:

- 1st port: providing all sba pool rates through a query to a database
- 2nd port: providing only EMEA rates through an api
- 3rd port: providing only US rates through a Kafka topic


```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://y.com/vocabulary/" }
  ],
  "id": "https://y.com/products/sba-pool-rates",
  "type": "DataProduct",
  "identifier": "sba-pool-rates",
  "title": "SBA Pool Rates",
  "description": "Rates for SBA Pool that are Mortgage Backed Securities. The data product
 is provided through 3 ports, one of them proving all SBA pool rates through a query to a
database, another port providing EMEA only rates through an api and another one providing
US only rates through a Kafka topic",
  "dataProductOwner": "https://www.schema.xxx/person/johnSmith",
  "dataProductLifecycleStatus": "https://ekgf.org/dprod/spec/develop/data/lifecycle-status
/Consume",
  "outputPort": [
    {
      "id": "https://y.com/service/sba-pool-rate-tabular-prod1",
      "type": "DataService",
      "identifier": "sba-pool-rate-tabular-prod1",
      "ex:environment": "PROD",
      "endpointURL": "jdbc:oracle:thin@sd656-5656-6745.ldn.organiation.com:43534/PGPERG.WO
RLD",
      "ex:sql": "select * from ...",
      "isAccessServiceOf": {
        "id": "https://y.com/distribution/sba-pool-rate-tabular",
        "type": "Distribution",
        "identifier": "sba-pool-rate-tabular",
        "format": "https://www.iana.org/assignments/media-types/application/sql",
        "isDistributionOf": {
          "id": "https://y.com/dataset/sba-pool-rate",
          "type": "Dataset",
          "identifier": "sba-pool-rate",
          "conformsTo": "https://spec.edmcouncil.org/fibo/ontology/SEC/Debt/MortgageBacked
Securities/SBA-Pool"
        }
      }
    },
    {
      "id": "https://y.com/service/sba-pool-rate-emea-api-prod1",
      "type": "DataService",
      "identifier": "sba-pool-rate-emea-api-prod1",
      "ex:environment": "PROD",
      "endpointURL": "https://example.org/mbs/SBA-Pool-location-emea",
      "conformsTo": "https://y.com/resources/users.yaml",
      "isAccessServiceOf": {
```

```
        "id": "https://y.com/distribution/sba-pool-rate-emea-json1",
        "type": "Distribution",
        "identifier": "sba-pool-rate-emea-json1",
        "format": "https://www.iana.org/assignments/media-types/application/json",
        "isDistributionOf": {
          "id": "https://y.com/dataset/sba-pool-rate-emea",
          "type": "Dataset",
          "identifier": "sba-pool-rate-emea",
          "description": "SBA pool data that cover EMEA accessed through an api",
          "spatial": "https://y.com/country/EMEA",
          "conformsTo": "https://spec.edmcouncil.org/fibo/ontology/SEC/Debt/MortgageBacked
Securities/SBA-Pool"
        }
      }
    },
    {
      "id": "https://y.com/service/sba-pool-rate-json-prod1",
      "type": "DataService",
      "identifier": "sba-pool-rate-json-prod1",
      "endpointURL": "kafka://q1.debt.mbs.dataset.us",
      "isAccessServiceOf": {
        "id": "https://y.com/distribution/sba-pool-rate-json",
        "type": "Distribution",
        "identifier": "sba-pool-rate-json",
        "format": "https://www.iana.org/assignments/media-types/application/json",
        "conformsTo": "http://confluent-registry-y/rates-json-schema.json",
        "ex:schemaCompatibility": "backwards compatible",
        "isDistributionOf": {
          "id": "https://y.com/dataset/sba-pool-rate-us",
          "type": "Dataset",
          "identifier": "sba-pool-rate-us",
          "spatial": "https://y.com/country/US",
          "conformsTo": "https://spec.edmcouncil.org/fibo/ontology/SEC/Debt/MortgageBacked
Securities/SBA-Pool"
        }
      }
    }
  ]
}
```

12 Core Data Product Extensions

For real world data products, the core data product details will be part of a wider set of metadata that allows the data and data product to be used effectively.

Below is an example of extending the DPROD data product, specifically by adding an agreement to a data product.

In this example, a Data Product Agreement is defined as a subclass of FIBO Agreement.

Definition of a simple Agreement based on FIBO:

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    {
      "fibo": "http://spec.edmcouncil.org/fibo/ontology/FND/Agreements/MetadataFNDAgreemen
ts/#",
      "ex": "http://example.org/dp#"
    }
  ],
  "@graph": [
    {
      "id": "ex:isSubjectToAgreement",
      "type": "rdf:Property",
      "label": "Data Product is Subject To FIBO Agreement",
      "rdfs:domain": { "id": "dprod:DataProduct" },
      "rdfs:range": { "id": "ex:DataProductAgreement" }
    },
    {
      "id": "ex:DataProductAgreement",
      "type": "rdfs:Class",
      "label": "DataProductAgreement",
      "rdfs:subClassOf": { "id": "fibo:Agreement" }
    }
  ]
}
```

A full definition of agreements for data products is likely to be more complex than a single class and may use other information models or their profiles (such as ODRL Policy) or create dedicated definitions.

Below is an example of a Data Product with an associated Data Product Agreement with an effective date.

Using the agreement:

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    {
      "fibo": "http://spec.edmcouncil.org/fibo/ontology/FND/Agreements/MetadataFNDAgreemen
ts/#",
      "ex": "http://example.org/dp#"
    }
  ],
  "@graph": [
    {
      "id": "https://y.com/data-product/company-sales",
      "type": "DataProduct",
      "outputPort": {
        "id": "https://y.com/data-product/company-sales/port/2025-sales",
```

```
        "type": "DataService",
        "label": "Sales",
        "endpointURL": "https://y.com/data-product/company-sales/port/2025-sales",
        "isAccessServiceOf": {
          "id": "https://y.com/data-product/company-sales/distribution/2025-sales",
          "type": "Distribution",
          "format": "https://www.iana.org/assignments/media-types/application/json",
          "isDistributionOf": {
            "id": "https://y.com/data-product/company-sales/dataset/2025-sales",
            "type": "Dataset",
            "label": "Sales",
            "conformsTo": "https://y.com/schema/Sale"
          }
        }
      },
      "ex:isSubjectToAgreement": { "id": "ex:VVSimpleAgreement" }
    },
    {
      "id": "ex:VVSimpleAgreement",
      "type": "ex:DataProductAgreement",
      "label": "Very Simple Data Product Agreement",
      "fibo:hasEffectiveDate": { "type": "xsd:date", "@value": "2024-08-31" }
    }
  ]
}
```

13 Data Schema

The Data Product provides to the consumers (dprod:outputDataset) datasets defined based on DCAT. Datasets should be described (dcat:conforms) with logical models. Logical models describe business entities and their properties (attributes and relationships) with consistent business terms and they are technology independent. Ideally, logical models are based on existing standards eg, FIBO, CDM etc. If a logical model does not exist to describe the dataset, then the dataset publisher can create one, preferably by using SHACL modelling language:

Example of a Dataset conforming to a SHACL Schema:

```
exampleDataset dcat:conforms exampleSchema:DatasetLogicalSchema.
exampleSchema:DatasetLogicalSchema a owl:Ontology, dct:Standard.
```

Based on SHACL all entities that exist in the dataset are Node Shapes (1). The attributes of the entities are described as Property Shapes with sh:datatype (2) The relationships are also defined as Property Shaped with sh:class the target class of the relationship (3)

```
@prefix dc:            <http://purl.org/dc/elements/1.1/> .
@prefix rdfs:          <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:            <http://www.w3.org/ns/shacl#> .
@prefix xsd:           <http://www.w3.org/2001/XMLSchema#> .
@prefix example:       <https://y.com/schema/> .
@prefix exampleSchema: <https://y.com/schema/logical/> .

# definition of the entity as a Node Shape (1)
example:Account a sh:NodeShape ;
    # human readable name of the entity
    rdfs:label "Account"@en ;
    # description of the entity
    dc:description "An Account is..." ;
    # an account has a property shape Account Age.
    # Definition of the property shape follows (2)
    sh:property example:Account-AccountAge ;
    # an account has a property shape Account Branch.
    # Definition of the property shape follows (3)
    sh:property example:Account-AccountBranch ;
    rdfs:isDefinedBy exampleSchema:DatasetLogicalSchema;
.

# (2) Definition of the Account-AccountAge property shape
#     describing that an account MUST have exactly one
#     AccountAge attribute and its datatype is integer
example:Account-AccountAge a sh:PropertyShape ;
    sh:path example:AccountAge ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    rdfs:isDefinedBy exampleSchema:DatasetLogicalSchema ;
.

# (3) Definition of the Account-AccountBranch property shape
#     describing than an account must have at least one
#     Account Branch which is another entity
example:Account-AccountBranch a sh:PropertyShape ;
    sh:path example:AccountBranch ;
    sh:class  example:Branch ;
```

```
    sh:minCount 1 ;
    rdfs:isDefinedBy exampleSchema:DatasetLogicalSchema ;
.

# definition of the entity Branch as a Node Shape (1)
example:Branch a sh:NodeShape ;
    rdfs:label "Branch"@en ;
    dc:description "A Branch is.." ;
    rdfs:isDefinedBy exampleSchema:DatasetLogicalSchema ;
.
```

```
{
  "@context": "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
  "id": "https://y.com/products/equity-trade-xxx",
  "type": "DataProduct",
  "title": "Equity Trade XXX",
  "description": "Trade data defining the outcome of equity trades between parties in diff
erent stock markets, where the terms are primarily reflected in the tradable product. Addi
tionally, Trade includes attributes such as the trade date, transacting parties, and settl
ement terms. Some attributes, such as the parties, are already defined in the Party Produc
t and are simply referenced in Trade",
  "outputPort": {
    "id": "https://y.com/service/equity-trade-euronext-paris",
    "type": "DataService",
    "endpointURL": "abfss://datasetsv1@demo.dfs.core.windows.net/demo/full/trade-euronext"
,
    "isAccessServiceOf": {
      "id": "https://y.com/distribution/equity-trade-euronext-paris-parquet",
      "type": "Distribution",
      "format": "https://www.iana.org/assignments/media-types/application/vnd.apache.parqu
et",
      "isDistributionOf": {
        "id": "https://y.com/dataset/equity-trade-euronext-paris",
        "type": "Dataset",
        "title": "Equity Trade Euronext Paris XXX",
        "conformsTo": "https://spec.edmcouncil.org/fibo/ontology/BP/Process/FinancialConte
xtAndProcess/SecuritiesTrade"
      }
    }
  }
}
```

14 Data Quality

- DQV is a standard vocabulary to describe data quality and is proposed along with the DCAT vocabulary to measure the quality of datasets.
- It is based on 3 basic entities: Dimensions, the Metrics that belong to its dimension, and the Measurement for each metric.
- Data Quality can be computed at different levels, eg. data product, dataset, table, column level etc.
- Usually, data quality metrics are measured at a dataset level and can inform more higher-level quality metrics at a data product level.

```
{
  "@context": "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
  "@graph": [
    {
      "id": "https://y.com/derived-quality-measurementA",
      "type": "QualityMeasurement",
      "value": 1,
      "computedOn": { "id": "https://y.com/products/uk-bonds", "type": "DataProduct" },
      "isMeasurementOf": {
        "id": "https://y.com/metric/stale-dataset-count",
        "type": "Metric",
        "label": "Number of stale datasets"
      }
    },
    {
      "id": "https://y.com/quality-measurement-B",
      "type": "QualityMeasurement",
      "value": false,
      "computedOn": {
        "id": "https://y.com/products/uk-bonds/yearlyPrices",
        "type": "Dataset"
      },
      "isMeasurementOf": {
        "id": "https://y.com/metric/expected-distribution-frequency-achieved",
        "type": "Metric",
        "label": "Expected distribution frequency achieved"
      }
    },
    {
      "id": "https://y.com/products/uk-bonds",
      "type": "DataProduct",
      "outputPort": {
        "id": "https://y.com/service/uk-bonds-quality-report",
        "type": "DataService",
        "endpointURL": "https://y.com/uk-bonds/quality-report",
        "isAccessServiceOf": {
          "id": "https://y.com/distribution/uk-bonds-quality-report",
          "type": "Distribution",
          "isDistributionOf": {
            "id": "https://y.com/dataset/uk-bonds-quality-measurements",
            "type": "Dataset",
            "conformsTo": "http://www.w3.org/ns/dqv#QualityMeasurement"
          }
        }
      }
```

```
    }
  ]
}
```

15 Equity Trade

Example of a data product for Equity Trades.

The Equity Trades Data Product provides two datasets to the consumers: one for trades in London Stock Exchange (LSEG) and one in Euronext.


```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://y.com/vocabulary/" }
  ],
  "id": "https://y.com/products/equity-trade-xxx",
  "type": "DataProduct",
  "identifier": "equity-trade-xxx",
  "title": "Equity Trade XXX",
  "description": "Trade data defining the outcome of equity trades between parties in diff
erent stock markets, where the terms are primarily reflected in the tradable product. Addi
tionally, Trade includes attributes such as the trade date, transacting parties, and settl
ement terms. Some attributes, such as the parties, are already defined in the Party Produc
t and are simply referenced in Trade",
  "dataProductOwner": "https://www.schema.xxx/person/AnnTaylor",
  "dataProductLifecycleStatus": "https://ekgf.org/dprod/spec/develop/data/lifecycle-status
/Consume",
  "outputPort": [
    {
      "id": "https://y.com/service/equity-trade-euronext-xxx-adls-prod-1",
      "type": "DataService",
      "identifier": "equity-trade-euronext-xxx-tabular-adls-prod",
      "endpointURL": "abfss://datasetsv1@demo.dfs.core.windows.net/demo/full/trade-euronex
t",
      "description": "Details for accessing storage account",
      "isAccessServiceOf": {
        "id": "https://y.com/distribution/equity-trade-euronext-xxx-tabular",
        "type": "Distribution",
        "identifier": "equity-trade-euronext-xxx-tabular",
        "format": "https://www.iana.org/assignments/media-types/application/vnd.apache.par
quet",
        "conformsTo": "https://cdm.finos.org/docs/event-model",
        "isDistributionOf": {
          "id": "https://y.com/dataset/equity-trade-euronext-paris",
          "type": "Dataset",
          "identifier": "equity-trade-euronext-paris-xxx",
          "title": "Equity Trade Euronext Paris XXX",
          "ex:datasetOwner": { "id": "https://www.schema.xxx/person/JohnBarks" },
          "conformsTo": "https://cdm.finos.org/docs/event-model"
        }
      }
    },
    {
      "id": "https://y.com/service/equity-trade-lseg-xxx-adls-prod-1",
      "type": "DataService",
      "identifier": "equity-trade-lseg-xxx-tabular-adls-prod",
      "endpointURL": "abfss://datasetsv1@demo.dfs.core.windows.net/demo/full/trade-lseg",
      "description": "Details for accessing storage account",
      "isAccessServiceOf": {
        "id": "https://y.com/distribution/equity-trade-lseg-xxx-tabular",
```

```
        "type": "Distribution",
        "identifier": "equity-trade-lseg-xxx-tabular",
        "format": "https://www.iana.org/assignments/media-types/application/vnd.apache.par
quet",
        "conformsTo": "https://cdm.finos.org/docs/event-model",
        "isDistributionOf": {
          "id": "https://y.com/dataset/equity-trade-lseg-xxx",
          "type": "Dataset",
          "identifier": "equity-trade-lseg-xxx",
          "title": "Equity Trade LSEG XXX",
          "conformsTo": "https://cdm.finos.org/docs/event-model"
        }
      }
    }
  ]
}
```

16 Data Rights

ODRL is a W3C standard to describe rights and entitlements. Based on ODRL, data product and dataset publishers can describe policies in a consistent, standard and machine-readable manner. Policies contain permissions and prohibitions on specific actions that are required to be met by stakeholders.

In addition, policies may be limited by constraints (eg. temporal or geographical constraints) and duties (eg. payments) that may be imposed on the permissions.

Policies and their permitted or prohibited actions can be described at different levels, eg. a policy can target a data product, a dataset, a data service or even a column.

Sophisticated engines should interpret and enforce the ODRL policies at the appropriate level eg.:

```
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix examplePolicy: <https://data.org/policy/> .
@prefix exampleProduct: <https://data.org/data-product/> .
@prefix exampleDataset: <https://data.org/dataset/> .

examplePolicy:A odrl:target exampleProduct:ProductA .
examplePolicy:B odrl:target exampleDataset:DatasetA1 .
```

An example of an agreement follows, that describes permission to use all the datasets of the product if the user is working inside EMEA or APAC:

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "reg": "https://www.region.taxonomy/v/1/" }
  ],
  "id": "https://data.org/policy/examplePolicyA",
  "type": "Agreement",
  "permission": [
    {
      "target": { "id": "https://data.org/data-product/equity-trade-xxx" },
      "action": "use",
      "assignee": {
        "id": "https://example.org/DataDepartment/emea-and-apac-staff",
        "type": "PartyCollection",
        "refinement": [
          {
            "leftOperand": "odrl:spatial",
            "operator": "isAnyOf",
            "rightOperand": [
              { "id": "reg:EMEA" },
              { "id": "reg:APAC" }
            ]
          }
        ]
      }
    }
  ]
}
```

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
```

```
    { "reg": "https://www.region.taxonomy/v/1/" }
  ],
  "id": "https://data.org/policy/56456df-dfg-34535345-5545",
  "type": "Agreement",
  "permission": [
    {
      "target": { "id": "https://data.org/data-product/equity-trade-xxx" },
      "assigner": { "id": "https://schema.org/person/AdamSmith" },
      "assignee": {
        "id": "https://example.org/DataDepartment/emea-and-apac-staff",
        "type": "PartyCollection",
        "source": { "id": "https://example.org/DataDepartment" },
        "refinement": [
          {
            "leftOperand": "odrl:spatial",
            "operator": "isAnyOf",
            "rightOperand": [
              { "id": "reg:EMEA" },
              { "id": "reg:APAC" }
            ],
            "description": "Permission to read all the datasets of the product if the user
 is working inside EMEA or APAC"
          }
        ]
      },
      "action": "use"
    }
  ]
}
```

17 Observability Ports

An Observability Port is a designated interface or endpoint in a system or application specifically used for monitoring and diagnostic purposes. It allows external tools or services to collect and analyze data related to the system's performance, health, and behaviour. By exposing metrics, logs, and traces through this port, administrators and developers can gain insights into the system's state, troubleshoot issues, and ensure it operates efficiently and reliably.

17.1 Defining Observability Ports in DPROD

DPROD has a schema-first design. The first thing to do is define a schema for the logging information. It could be a schema based on OpenTelemetry, but this uses RLOG (which is a semantic ontology for logging).

To find the Observability Port, query the ports to identify the ones that return an `RLOG:Entry`:

```
outputPort >> isAccessServiceOf >> isDistributionOf >> conformsTo >> rlog:Entry
```

17.2 Example Data Product with Observability Port

One can see that the example data product has two ports, one with the data and one with the logging. This query will return the URI of the port that returns logging data: `https://y.com/uk-bonds/observability-port`.

Here is an example of a data product with an observability port:

```
{
  "@context": "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
  "@graph": [
    {
      "id": "https://y.com/data-product/uk-bonds",
      "type": "DataProduct",
      "inputPort": [
        {
          "id": "https://y.com/data-product/uk-bonds/port/2024-data",
          "type": "DataService"
        }
      ],
      "outputPort": [
        {
          "id": "https://y.com/data-product/uk-bonds/port/2024-observability",
          "type": "DataService",
          "label": "Observability Port",
          "endpointURL": "https://y.com/data-product/uk-bonds/port/2024-observability",
          "isAccessServiceOf": {
            "type": "Distribution",
            "format": "https://www.iana.org/assignments/media-types/application/json",
            "isDistributionOf": {
              "id": "https://y.com/data-product/uk-bonds/dataset/2024-observability",
              "type": "Dataset",
              "conformsTo": "https://y.com/schema/ObservabilityLog"
            }
          }
        },
        {
          "id": "https://y.com/data-product/uk-bonds/port/2024-data",
          "type": "DataService",
          "label": "Data Port",
```

```
          "endpointURL": "https://y.com/data-product/uk-bonds/port/2024-data",
          "isAccessServiceOf": {
            "type": "Distribution",
            "format": "https://www.iana.org/assignments/media-types/application/json",
            "isDistributionOf": {
              "id": "https://y.com/data-product/uk-bonds/dataset/2024-data",
              "type": "Dataset",
              "conformsTo": "https://y.com/schema/Data"
            }
          }
        }
      ]
    }
  ]
}
```

Given that the schema defines the class for an observation, it can be used to find all observability ports on data product like this:

```
[https://y.com/data-product/uk-bonds/port/2024-observability] >> isAccessServiceOf >> isDi
stributionOf >> conformsTo >> https://y.com/schema/ObservabilityLog
```

In Linked Data a SPARQL query would do that:

```
SELECT ?port
WHERE
{
  ?port a dcat:DataService .
  ?port (dprod:isAccessServiceOf/dprod:isDistributionOf)/dcat:conformsTo rlog:Entry
}
```

This query will return the URI of the port that provides logging data: `https://y.com/data-product/uk-bonds/port/2024-observability`.

18 Data Lineage

It is important to be able to trace the lineage of data. Within DPROD, this can be done in two ways: at a high level from one data product to another and, if desired, at the more detailed level of the underlying datasets.

18.1 High Level Lineage: Between Data Products

Data products have input and output ports, and one data product's input port will point to another data product's output port.

This allows a user to query the lineage. The data products all have URLs as identifiers, and properties all connect to each other, so a query can walk from one data product to the downstream data products that feed it.

One can follow the path that leads from one data product to another like this:

```
Data Product >> inputPort >> isAccessServiceOf >> isDistributionOf >> Input Data Product
```

The following example data has three data products that connect to each other through their input and output ports:

```
{
  "@context": "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
  "@graph": [
    {
      "id": "https://y.com/data-product/company-finance",
      "type": "DataProduct",
      "inputPort": [
        {
          "id": "https://y.com/data-product/company-sales/port/2025-sales",
          "type": "DataService"
        },
        {
          "id": "https://y.com/data-product/company-hr/port/2025-payroll",
          "type": "DataService"
        }
      ],
      "outputPort": {
        "id": "https://y.com/data-product/company-sales/port/2025-balance-sheet",
        "type": "DataService",
        "label": "Balance Sheet",
        "endpointURL": "https://y.com/data-product/company-sales/port/2025-c",
        "isAccessServiceOf": {
          "type": "Distribution",
          "format": "https://www.iana.org/assignments/media-types/application/json",
          "isDistributionOf": {
            "id": "https://y.com/data-product/company-sales/dataset/2025-balance-sheet",
            "type": "Dataset",
            "conformsTo": "https://y.com/schema/BalanceSheet"
          }
        }
      }
    },
    {
      "id": "https://y.com/data-product/company-sales",
      "type": "DataProduct",
      "outputPort": {
        "id": "https://y.com/data-product/company-sales/port/2025-sales",
        "type": "DataService",
```

```
        "label": "Sales",
        "endpointURL": "https://y.com/data-product/company-sales/port/2025-sales",
        "isAccessServiceOf": {
          "type": "Distribution",
          "format": "https://www.iana.org/assignments/media-types/application/json",
          "isDistributionOf": {
            "id": "https://y.com/data-product/company-sales/dataset/2025-sales",
            "type": "Dataset",
            "label": "Sales",
            "conformsTo": "https://y.com/schema/Sale"
          }
        }
      }
    },
    {
      "id": "https://y.com/data-product/company-hr",
      "type": "DataProduct",
      "outputPort": {
        "id": "https://y.com/data-product/company-sales/port/2025-payroll",
        "type": "DataService",
        "label": "Payroll",
        "endpointURL": "https://y.com/data-product/company-hr/port/2025-payroll",
        "isAccessServiceOf": {
          "type": "Distribution",
          "format": "https://www.iana.org/assignments/media-types/text/csv",
          "isDistributionOf": {
            "id": "https://y.com/data-product/company-sales/dataset/2025-payroll",
            "type": "Dataset",
            "label": "Payroll",
            "conformsTo": "https://y.com/schema/Payroll"
          }
        }
      }
    }
  ]
}
```

Given this example data, starting at the data product `https://y.com/data-product/company-finance`, one could walk the relationships to find the input data products that feed it:

```
https://y.com/data-product/company-finance >>
    :inputPort >>
    :isAccessServiceOf >>
    :isDistributionOf >> [
        https://y.com/data-product/company-sales,
        https://y.com/data-product/company-hr
    ]
```

In Linked Data, this would use a query such as:

```
PREFIX :      <https://y.com/data-product/>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dprod: <https://ekgf.org/dprod/spec/develop/>

SELECT DISTINCT ?input
WHERE
```

```
{
  :company-finance dprod:inputPort ?inputPort .
  ?inputPort dprod:isAccessServiceOf/dprod:isDistributionOf/rdfs:label ?input .
}
```

18.2 Detailed Level: Between Datasets

To track lineage at a more granular level, one can also use PROV (https://www.w3.org/TR/prov-o/) at the dataset level.

```
@prefix dap:     <https://data.csiro.au/dap/> .
@prefix dcat:    <http://www.w3.org/ns/dcat#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix prov:    <http://www.w3.org/ns/prov#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

dap:atnf-P366-2003SEPT
  rdf:type dcat:Dataset ;
  dcterms:bibliographicCitation "Burgay, M; McLaughlin, M; Kramer, M; Lyne, A; Joshi, B; P
earce, G; D'Amico, N; Possenti, A; Manchester, R; Camilo, F (2017): Parkes observations fo
r project P366 semester 2003SEPT. v1. CSIRO. Data Collection. https://doi.org/10.4225/08/5
98dc08d07bb7" ;
  dcterms:title "Parkes observations for project P366 semester 2003SEPT"@en ;
  dcat:landingPage <https://data.csiro.au/dap/landingpage?pid=csiro:P366-2003SEPT> ;
  prov:wasGeneratedBy dap:P366 ;
  .

dap:P366
  rdf:type prov:Activity ;
  dcterms:type <http://dbpedia.org/resource/Observation> ;
  prov:startedAtTime "2000-11-01"^^xsd:date ;
  prov:used dap:Parkes-radio-telescope ;
  prov:wasInformedBy dap:ATNF ;
  rdfs:label "P366 - Parkes multibeam high-latitude pulsar survey"@en ;
  rdfs:seeAlso <https://doi.org/10.1111/j.1365-2966.2006.10100.x> ;
  .
```

See: https://www.w3.org/TR/vocab-dcat-3/#examples-dataset-provenance.

19 Collections and Target Inheritance

Two ODRL mechanisms keep large contracts short, and DPROD gives both a precise meaning.

Target inheritance. A rule with no `odrl:target` of its own inherits every target of its policy. The bundle offer names three products at policy level. Its blanket rules, read permitted and redistribution prohibited, say nothing about targets and so apply to all three. The aggregation permission is about the bundle as a whole, so it names that asset explicitly. The contract names the same three targets, so its read permission covers them too; without them, a rule with no target would apply to any asset.

Collections. Assets and parties are grouped with `odrl:partOf` into an `odrl:AssetCollection` or `odrl:PartyCollection`, and a rule addressed to a collection applies to every member. DPROD evaluates membership over the transitive closure of `odrl:partOf`. So the contract's assignee is the whole Trading Division and both of its teams are covered, and a rule on the reference data collection covers the country-codes table inside it. Only explicit membership is supported; ODRL's derived collections (`odrl:source` with `odrl:refinement`) are rejected by the profile.

Membership is stated on the member, pointing up at its collection, so the members cannot be nested inside the contract. This is the one example that needs a `@graph`: the contract first, then the collection memberships.

A contract accepts exactly one offer. If a consumer wants a subset of what a provider publishes, the provider publishes an offer for that subset, or a multi-target offer such as this one.

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "@graph": [
    {
      "id": "ex:tradingDivisionBundle2026",
      "type": "DataContract",
      "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
      "assigner": "ex:dataPlatformTeam",
      "assignee": "ex:tradingDivision",
      "target": ["ex:marketPrices", "ex:referenceData", "ex:riskMetrics"],
      "permission": {
        "type": "Permission",
        "label": "Every team in the Trading Division may read",
        "assignee": "ex:tradingDivision",
        "action": "read"
      },
      "acceptsOffer": {
        "id": "ex:marketDataBundle",
        "type": "DataOffer",
        "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
        "assigner": "ex:dataPlatformTeam",
        "target": ["ex:marketPrices", "ex:referenceData", "ex:riskMetrics"],
        "permission": [
          {
            "type": "Permission",
            "label": "Read: applies to all three targets",
            "action": "read"
          },
          {
            "type": "Permission",
            "label": "Aggregate: applies to the bundle as a whole",
            "action": "aggregate",
```

```
            "target": "ex:marketDataBundleAsset"
          }
        ],
        "prohibition": {
          "type": "Prohibition",
          "label": "No redistribution: applies to all three targets",
          "action": "distribute"
        }
      }
    },
    { "id": "ex:tradingDivision", "type": "PartyCollection" },
    {
      "id": "ex:tradingDataTeam",
      "type": "Party",
      "partOf": "ex:tradingDivision"
    },
    {
      "id": "ex:analyticsTeam",
      "type": "Party",
      "partOf": "ex:tradingDivision"
    },
    { "id": "ex:referenceData", "type": "AssetCollection" },
    {
      "id": "ex:countryCodes",
      "type": "Asset",
      "partOf": "ex:referenceData"
    }
  ]
}
```

20 Data Offer and Contract

A data contract in DPROD is an ODRL 2.2 policy. The provider of a data product publishes a `dprod:DataOffer` (an `odrl:Offer`) stating what it commits to deliver, what consumers may do with the data, and what they may not do. A consumer accepts that offer, and the acceptance is recorded as a `dprod:DataContract` (an `odrl:Agreement`) that names both parties and points at the offer through `dprod:acceptsOffer`.

Below, the Trading Data Team offers its market prices with two commitments of its own, a delivery every 15 minutes with a 5-minute deadline and conformance to a published schema, and one duty the consumer takes on, a monthly usage report. Consumers may display and derive from the data but may not redistribute it. The Analytics Team has accepted the offer for 2026.

Three things are worth noticing:

- **The provider's duties name their bearer.** `dprod:subjectOfDuty` says who must act. The usage report has no subject in the offer because the consumer is not yet known; it becomes the consumer's duty when the contract is made.
- **Rules inherit the offer's target.** Only the schema-conformance duty states its own `odrl:target`, because it is about the schema rather than the data.
- **The contract binds the named consumer.** It restates the rules that now apply to the Analytics Team, each with an `odrl:assignee`; a contract must carry at least one rule of its own.
- **The offer is shown nested inside the contract** so the whole agreement can be read top to bottom. In practice the provider publishes the offer once and each contract refers to it by its identifier.

ODRL terms are written as plain names, just as DCAT terms are in a data product description. The simple context knows that an action, an operator or a status names a vocabulary term, so `"action": "display"` denotes `odrl:display` without any wrapping. Only values whose datatype the context cannot know, such as a `deadline` that may be a duration or a timestamp, carry an explicit `type`.

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:tradingMarketData2026",
  "type": "DataContract",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "assigner": "ex:tradingDataTeam",
  "assignee": "ex:analyticsTeam",
  "target": "ex:marketPrices",
  "contractLifecycleStatus": "Active",
  "effectiveDate": "2026-01-15T00:00:00Z",
  "expirationDate": "2026-12-31T23:59:59Z",
  "permission": {
    "type": "Permission",
    "assignee": "ex:analyticsTeam",
    "action": "display"
  },
  "prohibition": {
    "type": "Prohibition",
    "assignee": "ex:analyticsTeam",
    "action": "distribute"
  },
  "acceptsOffer": {
    "id": "ex:marketDataFeed",
    "type": "DataOffer",
```

```
    "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
    "assigner": "ex:tradingDataTeam",
    "target": "ex:marketPrices",
    "offerLifecycleStatus": "Active",
    "obligation": [
      {
        "type": "Duty",
        "subjectOfDuty": "ex:tradingDataTeam",
        "action": "ex:deliver",
        "schedule": {
          "id": "ex:every15Minutes",
          "type": "Schedule",
          "conformsTo": "dprod:Rfc5545ScheduleFormat",
          "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T000000\nRRULE:FREQ=MI
NUTELY;INTERVAL=15"
        },
        "deadline": { "@value": "PT5M", "type": "xsd:duration" }
      },
      {
        "type": "Duty",
        "subjectOfDuty": "ex:tradingDataTeam",
        "action": "ex:conformTo",
        "target": "ex:marketDataSchema"
      },
      {
        "type": "Duty",
        "action": "ex:reportUsage",
        "schedule": {
          "id": "ex:monthlyOnThe1st",
          "type": "Schedule",
          "conformsTo": "dprod:Rfc5545ScheduleFormat",
          "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T000000\nRRULE:FREQ=MO
NTHLY;BYMONTHDAY=1"
        },
        "deadline": { "@value": "P5D", "type": "xsd:duration" }
      }
    ],
    "permission": [
      { "type": "Permission", "action": "display" },
      { "type": "Permission", "action": "derive" }
    ],
    "prohibition": [
      { "type": "Prohibition", "action": "distribute" }
    ]
  }
}
```

21 Data Use Policy

Not every rule about data is a contract between two parties. An organisation also has standing rules: who may read employee data and for what purpose, what must be logged, what may never be done. These are expressed as an `odrl:Set`, an ODRL policy with no assigner or assignee, and evaluated by the same DPROD evaluator as a contract. The profile's fixed conflict rule applies: a prohibition always wins over a permission.

The HR data access policy shows the constraint patterns a policy writer typically needs:

- **Purpose**, using ODRL's own `odrl:purpose` operand.
- **Role**, through the permission's `odrl:assignee`.
- **Classification**, so the audit-logging duty applies only to confidential data. Its target is the access log, not the employee data, so it is stated explicitly.
- **Set membership**, with `odrl:isNoneOf` and a list of values.
- **Logical combination.** An `odrl:LogicalConstraint` with `odrl:and`, and DPROD's `dprod:not` for negation: model training is forbidden unless the environment is the isolated one.
- **Retention**, as a duty to delete once the retention period is exceeded.

Every operand a policy reads must say where its value comes from. Each is declared the first time it is used, with one `dprod:operandSource` and one `dprod:operandProperty`; later uses just name it.

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:hrDataAccessPolicy",
  "type": "Set",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "target": "ex:employeeData",
  "permission": [
    {
      "type": "Permission",
      "label": "HR Analytics may read, for analytics",
      "assignee": "ex:hrAnalyticsTeam",
      "action": "read",
      "constraint": {
        "type": "Constraint",
        "leftOperand": {
          "id": "odrl:purpose",
          "operandSource": "requestSource",
          "operandProperty": "odrl:purpose"
        },
        "operator": "eq",
        "rightOperand": { "id": "ex:analytics" }
      }
    },
    {
      "type": "Permission",
      "label": "Compliance may read, for compliance",
      "assignee": "ex:complianceTeam",
      "action": "read",
      "constraint": {
        "type": "Constraint",
        "leftOperand": "odrl:purpose",
```

```
      "operator": "eq",
      "rightOperand": { "id": "ex:compliance" }
    }
  }
],
"obligation": [
  {
    "type": "Duty",
    "label": "Access to confidential data must be logged within an hour",
    "action": "ex:log",
    "target": "ex:accessLog",
    "constraint": {
      "type": "Constraint",
      "leftOperand": {
        "id": "ex:classification",
        "operandSource": "requestSource",
        "operandProperty": "ex:classification"
      },
      "operator": "eq",
      "rightOperand": { "id": "ex:confidential" }
    },
    "deadline": { "@value": "PT1H", "type": "xsd:duration" }
  },
  {
    "type": "Duty",
    "label": "Delete after 7 years",
    "action": "delete",
    "constraint": {
      "type": "Constraint",
      "leftOperand": {
        "id": "ex:retentionPeriod",
        "operandSource": "requestSource",
        "operandProperty": "ex:retentionPeriod"
      },
      "operator": "gt",
      "rightOperand": { "@value": "P7Y", "type": "xsd:duration" }
    }
  }
],
"prohibition": [
  {
    "type": "Prohibition",
    "label": "No external sharing",
    "action": "distribute"
  },
  {
    "type": "Prohibition",
    "label": "No use outside production or staging",
    "action": "use",
    "constraint": {
      "type": "Constraint",
      "leftOperand": {
        "id": "ex:environment",
        "operandSource": "requestSource",
        "operandProperty": "ex:environment"
      },
      "operator": "isNoneOf",
      "rightOperand": {
        "@list": [
```

```
            { "id": "ex:production" },
            { "id": "ex:staging" }
          ]
        }
      }
    },
    {
      "type": "Prohibition",
      "label": "No model training, unless in an isolated environment",
      "action": "use",
      "constraint": {
        "type": "LogicalConstraint",
        "and": {
          "@list": [
            {
              "type": "Constraint",
              "leftOperand": {
                "id": "ex:processingMode",
                "operandSource": "requestSource",
                "operandProperty": "ex:processingMode"
              },
              "operator": "eq",
              "rightOperand": { "id": "ex:modelTraining" }
            },
            {
              "type": "LogicalConstraint",
              "not": {
                "type": "Constraint",
                "leftOperand": "ex:environment",
                "operator": "eq",
                "rightOperand": { "id": "ex:isolated" }
              }
            }
          ]
        }
      }
    }
  ]
}
```

22 Evaluation Context

DPROD is a specification, not a program. It defines what a policy means and what a conformant evaluator must do with it, but nothing in DPROD receives a request or returns a decision. An **evaluator** is an external program, written or chosen by an implementer, that follows this specification to answer one question: may this agent perform this action on this asset, now, under these policies?

A decision is made in three steps:

1. **The calling system** (for example an application, an API gateway or a data platform) needs a decision. It gathers every fact the policies need and builds one immutable `dprod:EvaluationContext` with four parts: the **request**, a **state**-of-the-world snapshot, the requesting **agent** and the **clock**.
2. **The evaluator** receives the policies and that context. It reads nothing else: no live graph, no database and no network.
3. It resolves each operand with a single lookup, applies the policies and returns a decision. A missing, multiple or wrongly typed value is an evaluation error, never a quietly false constraint.

Each `odrl:LeftOperand` a policy uses says where in the context its value comes from with `dprod:operandSource`, and which single property to read there with `dprod:operandProperty`:

- `requestSource` reads from the **request**: a node built by the calling system to describe what is being asked for (who, which action, which asset) together with any facts about the request that policies need. It is plain RDF, not an `odrl:Request`.
- `stateSource` reads from the **state** snapshot: facts about the world at the time of the request, such as whether the market is open, frozen by the calling system as a `prov:Entity` so their provenance can be kept.
- `contextSource` reads from the context itself, which is how the built-in agent and clock operands work.

This policy permits internal recipients to use market prices while the market is open. `ex:recipientType` is read from the request and `ex:marketOpen` from the state snapshot:

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:marketHoursPolicy",
  "type": "Set",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "target": "ex:marketPrices",
  "permission": {
    "type": "Permission",
    "action": "use",
    "constraint": {
      "type": "LogicalConstraint",
      "and": {
        "@list": [
          {
            "type": "Constraint",
            "leftOperand": {
              "id": "ex:recipientType",
              "operandSource": "requestSource",
              "operandProperty": "ex:recipientType"
            },
            "operator": "eq",
            "rightOperand": { "id": "ex:internal" }
          },
```

```
          {
            "type": "Constraint",
            "leftOperand": {
              "id": "ex:marketOpen",
              "operandSource": "stateSource",
              "operandProperty": "ex:marketOpen"
            },
            "operator": "eq",
            "rightOperand": true
          }
        ]
      }
    }
  }
}
```

Facts more than one hop away. Policies often depend on a fact that is not stored directly on the request. Suppose use of market prices is permitted only to recipients whose organisation is regulated in the EU. In the source data that fact is two hops from the agent:

```
@prefix ex: <https://example.org/> .

ex:alice ex:memberOf ex:acmeBank .
ex:acmeBank ex:regulatedIn ex:EU .
```

An operand cannot follow `ex:memberOf` and then `ex:regulatedIn`; earlier drafts allowed that with `dprod:path`, which has been removed. Instead the calling system, which already knows who Alice is and can query that data, resolves the join while building the request and writes the answer onto the request as a single property. The policy then reads it in one lookup:

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:euRecipientPolicy",
  "type": "Set",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "target": "ex:marketPrices",
  "permission": {
    "type": "Permission",
    "action": "use",
    "constraint": {
      "type": "Constraint",
      "leftOperand": {
        "id": "ex:recipientRegulator",
        "operandSource": "requestSource",
        "operandProperty": "ex:recipientRegulatedIn"
      },
      "operator": "eq",
      "rightOperand": { "id": "ex:EU" }
    }
  }
}
```

The normalised request carries the resolved value:

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:request-2026-03-02-0915",
  "assignee": "ex:alice",
  "action": "use",
  "target": "ex:marketPrices",
  "ex:recipientRegulatedIn": { "id": "ex:EU" }
}
```

This keeps the evaluator a pure, one-lookup function of its input. The join happens in the calling system, where the data lives, and its result becomes part of the immutable context, so a later audit sees exactly the value the decision was based on. A fact that describes the world rather than the request, such as whether the market is open, is flattened onto the state snapshot in the same way and read with `stateSource`.

The document below is the evaluation context an evaluator would be given for Alice's request at 09:15 on 2 March. Against the market-hours policy at the top of this example it yields a permit. DPROD's `dprod:currentAgent` and ODRL's standard `odrl:dateTime` are built-in operands bound to the context's agent and clock, and need no declaration.

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:evaluation-2026-03-02-0915",
  "type": "EvaluationContext",
  "agent": { "id": "ex:alice", "type": "Party" },
  "clock": "2026-03-02T09:15:00Z",
  "request": {
    "assignee": "ex:alice",
    "action": "use",
    "target": "ex:marketPrices",
    "ex:recipientType": { "id": "ex:internal" }
  },
  "state": { "type": "Entity", "ex:marketOpen": true }
}
```

23 Lifecycle and Versioning

DPROD keeps two questions apart. *Where is this offer or contract in its administrative life?* is answered by a person or a contract-management process and recorded with `dprod:offerLifecycleStatus` or `dprod:contractLifecycleStatus`. *Has this obligation been met?* is answered by an evaluator from observable facts and recorded with `dprod:dutyState`. Both take any `skos:Concept`; DPROD ships the optional scheme with `Pending`, `Active`, `Fulfilled` and `Violated`, and an enterprise may substitute its own.

The example shows both, plus versioning:

- **A version chain.** Customer API v2 supersedes v1 through `prov:wasRevisionOf`. The retired offer stays in the graph with its status set to `Fulfilled`, so contracts that accepted it remain interpretable.
- **A contract in force**, `Active`, with its effective and expiration dates.
- **Duty instances.** When the daily delivery duty is evaluated, each occurrence becomes its own `odrl:Duty` with a computed state: the 4 February delivery was made on time, the 5 February one missed its 30-minute window, and the 6 February one is not yet due.

Duty state is deliberately *not* a sub-property of `dprod:lifecycleStatus`: it is derived, and re-deriving it must never change an authored status.


```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:analyticsCustomerData2026",
  "type": "DataContract",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "assigner": "ex:dataPlatformTeam",
  "assignee": "ex:analyticsTeam",
  "contractLifecycleStatus": "Active",
  "effectiveDate": "2026-02-01T00:00:00Z",
  "expirationDate": "2026-12-31T23:59:59Z",
  "acceptsOffer": {
    "id": "ex:customerApiV2",
    "type": "DataOffer",
    "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
    "assigner": "ex:dataPlatformTeam",
    "target": "ex:customerData",
    "offerLifecycleStatus": "Active",
    "effectiveDate": "2026-01-01T00:00:00Z",
    "obligation": {
      "type": "Duty",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "action": "ex:deliver",
      "schedule": {
        "id": "ex:dailyAt0700",
        "type": "Schedule",
        "conformsTo": "dprod:Rfc5545ScheduleFormat",
        "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T070000\nRRULE:FREQ=DAIL
Y"
      },
      "deadline": { "@value": "PT30M", "type": "xsd:duration" }
    },
    "wasRevisionOf": {
      "id": "ex:customerApiV1",
```

```
      "type": "DataOffer",
      "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
      "assigner": "ex:dataPlatformTeam",
      "target": "ex:customerData",
      "offerLifecycleStatus": "Fulfilled",
      "expirationDate": "2025-12-31T23:59:59Z",
      "permission": { "type": "Permission", "action": "use" }
    }
  },
  "obligation": [
    {
      "id": "ex:delivery-2026-02-04",
      "type": "Duty",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "objectOfDuty": "ex:analyticsTeam",
      "action": "ex:deliver",
      "deadline": { "@value": "2026-02-04T07:30:00Z", "type": "xsd:dateTime" },
      "dutyState": "Fulfilled"
    },
    {
      "id": "ex:delivery-2026-02-05",
      "type": "Duty",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "objectOfDuty": "ex:analyticsTeam",
      "action": "ex:deliver",
      "deadline": { "@value": "2026-02-05T07:30:00Z", "type": "xsd:dateTime" },
      "dutyState": "Violated"
    },
    {
      "id": "ex:delivery-2026-02-06",
      "type": "Duty",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "objectOfDuty": "ex:analyticsTeam",
      "action": "ex:deliver",
      "deadline": { "@value": "2026-02-06T07:30:00Z", "type": "xsd:dateTime" },
      "dutyState": "Pending"
    }
  ]
}
```

24 ODCS Mapping

The Open Data Contract Standard (ODCS) describes a data contract as a YAML document that bundles schema, quality rules, service levels, team and infrastructure. DPROD Contracts covers a narrower question, the rights and obligations, and leaves schema to DCAT and SHACL and infrastructure to deployment descriptors. The two are complementary. This example translates the policy parts of the ODCS reference document into a DPROD data offer; each rule's label names the ODCS field it came from.

ODCS	DPROD
`kind: DataContract`, `status: active`, `version`	`dprod:DataOffer`, `dprod:offerLifecycleStatus`, `dct:hasVersion`
`team` owner	`odrl:assigner`
`slaProperties` `frequency` and `timeOfAvailability`	a delivery duty with a `dprod:schedule` and a `dprod:deadline`
`slaProperties` `generalAvailability` and `endOfSupport`	`dprod:effectiveDate` and `dprod:expirationDate`
`quality` `nullValues mustBe 0`, scheduled nightly	a conformance duty constrained on completeness, with a `dprod:schedule`
`slaProperties` `retention`	a duty to `odrl:delete` beyond the retention period
`roles` with `access: read`	an `odrl:Permission` for the role's party
`description.limitations`	an `odrl:Prohibition`

ODCS fields that are workflow or operations rather than policy, such as approver chains, severity and `servers`, have no counterpart and stay in ODCS.

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:sellerDataContract",
  "type": "DataOffer",
  "description": "Views built on top of the seller tables. Predict sales over time.",
  "hasVersion": "1.1.0",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "assigner": "ex:sellerDataTeam",
  "target": "ex:sellerData",
  "offerLifecycleStatus": "Active",
  "effectiveDate": "2022-05-12T09:30:10-08:00",
  "expirationDate": "2032-05-12T09:30:10-08:00",
  "obligation": [
    {
      "type": "Duty",
      "label": "slaProperties frequency + timeOfAvailability: daily, by 09:00",
      "subjectOfDuty": "ex:sellerDataTeam",
      "action": "ex:deliver",
      "schedule": {
        "id": "ex:dailyAt0900",
        "type": "Schedule",
        "conformsTo": "dprod:Rfc5545ScheduleFormat",
```

```
        "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T090000\nRRULE:FREQ=DAIL
Y"
      },
      "deadline": { "@value": "PT1H", "type": "xsd:duration" }
    },
    {
      "type": "Duty",
      "label": "quality nullValues mustBe 0, checked nightly",
      "subjectOfDuty": "ex:sellerDataTeam",
      "action": "ex:conformTo",
      "schedule": {
        "id": "ex:nightlyAt2000",
        "type": "Schedule",
        "conformsTo": "dprod:Rfc5545ScheduleFormat",
        "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T200000\nRRULE:FREQ=DAIL
Y"
      },
      "constraint": {
        "type": "Constraint",
        "leftOperand": {
          "id": "ex:completeness",
          "operandSource": "requestSource",
          "operandProperty": "ex:completeness"
        },
        "operator": "eq",
        "rightOperand": { "@value": "100", "type": "xsd:decimal" }
      }
    },
    {
      "type": "Duty",
      "label": "slaProperties retention: delete after 3 years",
      "action": "delete",
      "constraint": {
        "type": "Constraint",
        "leftOperand": {
          "id": "ex:retentionPeriod",
          "operandSource": "requestSource",
          "operandProperty": "ex:retentionPeriod"
        },
        "operator": "gt",
        "rightOperand": { "@value": "P3Y", "type": "xsd:duration" }
      }
    }
  ],
  "permission": {
    "type": "Permission",
    "label": "roles: microstrategy_user_opr, access: read",
    "assignee": "ex:reportingUsers",
    "action": "read"
  },
  "prohibition": {
    "type": "Prohibition",
    "label": "description.limitations: no buyer information",
    "action": "derive"
  }
}
```

25 Provider Duties

Most of what a data contract promises is expressed as duties on the provider. DPROD needs no new rule type for this: each is an ordinary `odrl:Duty` in the offer's `odrl:obligation`, with `dprod:subjectOfDuty` naming the provider as the party that must act. Four patterns cover almost every service-level commitment:

- **Delivery.** `dprod:schedule` (an identified `dprod:Schedule`, here an RFC 5545 rule with its start time and timezone) says when each delivery is due and `dprod:deadline` how long after that the provider has to deliver.
- **Schema conformance.** The duty is about the schema, not the data, so it states its own `odrl:target` instead of inheriting the offer's.
- **Change notification.** A deadline with no schedule is a lead time.
- **Quality.** A conformance duty with an `odrl:constraint`. The constraint's left operand is declared where it is used: `dprod:operandSource` and `dprod:operandProperty` tell an evaluator to read `ex:timeliness` from the authorisation request.

Consumer duties use exactly the same construct, minus the subject.

```
{
  "@context": [
    "https://ekgf.org/dprod/spec/develop/dprod-simple.jsonld",
    { "ex": "https://example.org/" }
  ],
  "id": "ex:riskMetricsOffer",
  "type": "DataOffer",
  "profile": "https://ekgf.org/dprod/spec/develop/contracts/profile/",
  "assigner": "ex:dataPlatformTeam",
  "target": "ex:riskMetrics",
  "obligation": [
    {
      "type": "Duty",
      "label": "Delivery: every day at 07:00, within 30 minutes",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "action": "ex:deliver",
      "schedule": {
        "id": "ex:dailyAt0700",
        "type": "Schedule",
        "conformsTo": "dprod:Rfc5545ScheduleFormat",
        "scheduleExpression": "DTSTART;TZID=Europe/London:20260101T070000\nRRULE:FREQ=DAIL
Y"
      },
      "deadline": { "@value": "PT30M", "type": "xsd:duration" }
    },
    {
      "type": "Duty",
      "label": "Schema conformance",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "action": "ex:conformTo",
      "target": "ex:riskMetricsSchema"
    },
    {
      "type": "Duty",
      "label": "Change notification: 14 days notice",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "action": "ex:notify",
      "target": "ex:schemaChanges",
```

```
      "deadline": { "@value": "P14D", "type": "xsd:duration" }
    },
    {
      "type": "Duty",
      "label": "Quality: data must be real-time",
      "subjectOfDuty": "ex:dataPlatformTeam",
      "action": "ex:conformTo",
      "constraint": {
        "type": "Constraint",
        "leftOperand": {
          "id": "ex:timeliness",
          "operandSource": "requestSource",
          "operandProperty": "ex:timeliness"
        },
        "operator": "eq",
        "rightOperand": { "id": "ex:realtime" }
      }
    }
  ]
}
```

A Acknowledgements

The editors gratefully acknowledge the feedback and contributions made by individuals who have participated in EDM Council's CDMC team.

B Document provenance

This record identifies the generator and source revisions collected for this publication.

Generator	GraphArch 0.2.29 (build 412)
Generator commit	`8908a9610d77`
Build operator	Rust CI, run 44327712575 attempt 1 (GitHub Actions, jgeluk)
Repository	https://github.com/EKGF/dprod.git
Source commit	`ded6bf94bf4f`
Source state	Exact committed revision.

Recent commits

Up to five commits, following the first-parent history from the source commit, with the author of each change. This is not the full contributor list.

Jacobus Geluk 15 November 2026, 16:58 UTC source commit
Jacobus Geluk 15 November 2026, 16:58 UTC `8221f039d0cb`
Jacobus Geluk 15 November 2026, 12:47 UTC `65278a1b262d`
Jacobus Geluk 15 November 2026, 12:08 UTC `72e9d272466a`
dependabot[bot] 14 November 2026, 22:49 UTC `d3645768f5c0`

Diagram interpretation

Class diagrams use UML shorthand for asserted OWL facts and declared SHACL constraints, without inference or a claim that validation passed. RDF properties remain independent resources even when drawn as class members or labelled associations. Domain/range declarations alone do not impose requiredness or cardinality. A property with a declared domain and range is a labelled association from the domain to the range; a datatype property, or one without a specific range, is an attribute of its domain. Multiple domains or ranges apply conjunctively, not as alternatives: each declared domain-range pair is drawn and holds for every use of the property. An explicitly declared union domain is drawn as one source box whose class alternatives are joined by or. A dotted rdfs:Resource source box is a presentation placeholder when no specific domain is declared in the selected inputs, not an asserted domain or a claim that none can be entailed. OWL class restrictions on the documented class and on neighbouring classes use UML notation instead of expression boxes, scoped to the restricted class and never declaring a global range or count. A datatype restriction is an attribute, name : type [multiplicity]; an object restriction is an association to its filler class with the multiplicity at the target end (a neighbouring class shows it as a typed attribute instead). An unqualified count uses the declared property range. someValuesFrom gives 1..* {qualified}; allValuesFrom gives * {all values}, narrowing the value type; a qualified count counts only values in the named filler and adds {qualified}; hasValue adds {= value}. Multiplicities follow one convention: equal bounds show one number such as 1, an unconstrained range shows *, and otherwise lower..upper such as 0..1 or 1..*. A maximum or exact count of zero, in OWL or SHACL, is a prohibition rather than a multiplicity: it appears as a class constraint row after the attributes, {no p}, or {no p in C} when only values in C are excluded. A zero minimum imposes no restriction and is omitted. {redefines} marks a restriction on a property whose declared domain is an asserted superclass, redefining the inherited property of the same name. An existential with a simple zero-minimum filler still requires a value; a positive count on the same class and property already guarantees that existence. Boolean class expressions, nested restrictions, inverse properties and property chains remain explicit boxes named after their form, such as Anonymous OWL intersection 1. Types and counts from simple property shapes enrich the UML notation. Identical qualifiers are shown once; where declaring shapes disagree, the disputed type or multiplicity is omitted from the drawing and remains in the dedicated shape documentation. No shape is chosen arbitrarily and conflicting counts are not silently intersected. An absent count imposes no lower or upper bound beyond zero or unbounded, respectively. Complex constraints require the linked full shape definition. Original

OWL facts and shape provenance remain in the published source graph. Some diagrams show at most five complete OWL relationship groups within an eighteen-fact budget, selected in deterministic IRI order; omitted groups remain in the published source graph. These views are not exhaustive. Names from the same ontology as the documented class are shown by label; names from another ontology add their prefixed name in parentheses, and unlabelled names show their prefixed name.