Rillor

EXPLAINER

What a data contract specifies, and how to write one

A data contract states a dataset's schema, quality rules, refresh terms and support before delivery. This explainer walks through the open standard and applies it to a commissioned dataset.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

A data contract is a written specification of a dataset that its producer and its consumers agree on, and the Open Data Contract Standard gives it one common structure.

- The Open Data Contract Standard (ODCS), at version 3.2.0 since 8 September 2026, defines a data contract as "the agreement between a data producer and consumers", and its README lists 13 sections, from fundamentals and schema to data quality, service levels and servers [1, 2].
- A data contract is a specification, separate from the agreement between the parties. The Data Contract Specification, an earlier format deprecated when ODCS v3.1.0 was released, says the term does not mean a contract in the legal sense and calls the mutual agreement a "data usage agreement" [17].
- Platforms enforce a contract's shape, its column names and data types, more often than its content, the values themselves. The dbt transformation tool refuses to build a model whose columns or data types differ from its contract, and its documentation notes that "Most platforms only enforce not_null" [19]. Snowflake standard tables and BigQuery do not enforce primary or foreign keys [22, 23].
- Content rules need measured checks. An ODCS quality rule pairs a metric, such as the count of null values, with an operator and a unit, for example fewer than 1% nulls in a column, and can name one of seven quality dimensions [7].
- Coverage and refresh have standard terms. The W3C's Data Catalog Vocabulary (DCAT) version 3 names temporal coverage, spatial coverage and update frequency [26], and ODCS recommends twelve service-level properties, including frequency, latency and retention [8].
- Breaking changes can be defined in advance. dbt counts a removed column or a changed data type as breaking [19], Confluent's Schema Registry checks backward compatibility by default [24], and MLCommons' Croissant format recommends MAJOR.MINOR.PATCH versions for datasets [32].
- A commissioned dataset's source universe, entities, fields, format, history, geography, refresh cadence and delivery can each be written in ODCS. Source universe and geography have no dedicated field and go in descriptions, value lists or custom properties [1, 5, 6, 7].

Background

Where the term came from

Andrew Jones writes that he came up with data contracts at GoCardless in August 2020, wrote about them publicly for the first time in April 2021 and described that implementation in December 2021 [16]. Chad Sanderson, in an August 2022 post, called them "API-like agreements between Software Engineers who own services and Data Consumers" [36]. His example was a model trained on data from three Postgres tables, which breaks if columns are dropped, if the values of a field do not stay the same or if a 1:1 rule between shipments and shipment opportunities fails [36].

PayPal's repository for its data contract template was created on 21 April 2023, and its README dates the open-source announcement to 1 May 2023 [15]. The template described "the data contract being used in the implementation of Data Mesh at PayPal" in six sections, and the project has since become the Open Data Contract Standard, part of Bitol [15]. Bitol was contributed to the LF AI & Data Foundation in September 2023 and has been a Graduate stage project there since June 2026 [14]. ODCS has had eight releases, from v2.2.0 in July 2023 to v3.2.0 (counted for this report from the release list) [4]. A second format, the Data Contract Specification from datacontract.com, was deprecated with the release of ODCS v3.1.0, and its README, read on 8 November 2026, states that it stays supported in the Data Contract CLI and Entropy Data until the end of 2026 [17].

Between two companies

These early examples describe exchanges inside one organization: services and their data consumers [36], and a data mesh at PayPal [15]. Written terms also exist for data sold between companies. FISD, an industry association, published recommendations in 2021 on "Service Level Agreement ("SLA") provisions that can be incorporated into data license contracts" for alternative data [31]. This report applies the contract structure to a dataset commissioned from an outside builder, where the builder is the producer and the buyer is the consumer.

The W3C's Data on the Web Best Practices ask publishers to "Provide information about data quality and fitness for particular purposes" [28], and Datasheets for Datasets proposes that every dataset come with a datasheet documenting its motivation, composition, collection process and recommended uses [35]. A contract is one place to write those answers down before delivery. This report is for data leads and procurement staff who need a written agreement on what a dataset must contain, and for practitioners who compare the open standard with what software enforces.

Method

Sources. The ODCS README, two release notes, the release list and nine section pages [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. The Bitol project page and the repositories ODCS grew from or replaced [14, 15, 17]. Two self-published posts, used only for who said what and when [16, 36]. Documentation from datacontract.com, dbt Labs, Databricks, Snowflake, Google Cloud and Confluent [18, 19, 20, 21, 22, 23, 24, 25]. W3C and Dublin Core vocabularies [26, 27], the W3C's Data on the Web Best Practices [28], Croissant 1.0 [32] and Semantic Versioning [33]. The UK Government Data Quality Framework [29], FISD's Alternative Data Council and its 2021 service-level recommendations [30, 31], a chapter of Google's Site Reliability Engineering book [34] and Datasheets for Datasets [35].

How. Every source was read on 8 November 2026. The ODCS pages are the "latest" edition, whose fundamentals page sets v3.2.0 as the default version [5]. Facts about a platform come from its own documentation, and dbt's tables are cited for what dbt enforces. Counts marked "counted for this report" were made from the sources' own lists.

The worked contract. Finding 8 maps the eight items a client specifies for a custom dataset on rillor.com (source universe, entities, fields, format, history, geography, refresh cadence and delivery) to ODCS fields and to DCAT and Croissant terms. An item has "no dedicated field" when no ODCS section or key names it, judged from the section list and the section pages [1, 5, 6, 7, 8, 9, 10, 11, 12, 13]. The YAML there is an illustration with placeholders, written from the key names on the ODCS pages.

Findings

1. What a data contract is under ODCS v3.2.0

ODCS "describes a structure for a data contract", is available under the Apache 2.0 license and defines a data contract as "the agreement between a data producer and consumers" [1]. A contract is a YAML document, and the project publishes a JSON Schema "for validation of your YAML files" [1]. Among the highlights of version 3.2.0 are variables, for "One contract that works in dev, test, and production, with no secrets in Git", and deprecation, to "Retire fields and ports without breaking consumers" [2]. Version 3.0.0, of 20 October 2024, dropped the `username` and `password` keys because "credentials should not be stored in the data contract", and renamed `columns` to `properties` and `isPrimaryKey` to `primaryKey` [3].

The Data Contract Specification gave a longer definition, "a document that defines the structure, format, semantics, quality, and terms of use for exchanging data between a data provider and their consumers", and the shorthand "Think of an API, but for data" [17]. It calls contracts "first, and foremost" a communication tool, and later the basis for code generation, testing, schema validations, quality checks, monitoring and access control [17].

The same README separates the specification from the agreement. The term refers to "a specification that is usually owned by the data provider" and does not align with a contract "in a legal sense as a mutual agreement between two parties" [17]. That mutual agreement is the "data usage agreement", which refers to a data contract and has "a defined lifecycle, start/end date" [17]. This report describes what the sources state and gives no legal advice.

Confluent's documentation adds that "A schema is only one element of a data contract", and lists structure, integrity constraints, metadata, rules or policies, and change or evolution [25]. In its model, "The upstream component enforces the data contract" [25].

2. The sections of a contract

The ODCS README lists 13 sections: fundamentals, schema, references, data quality, support and communication channels, pricing, team, roles, service-level agreement (SLA), infrastructure and servers, authoritative definitions, tags, and custom and other properties [1]. PayPal's 2023 template had six [15].

Section	What it states	Example keys	Source
Fundamentals	Identity, version, status, purpose and limits	`apiVersion`, `id`, `kind`, `version`, `status`, `description`	[5]
Schema	Objects and their properties	`logicalType`, `physicalType`, `required`, `unique`, `primaryKey`	[6]

Section	What it states	Example keys	Source
References	Relationships between properties	foreign keys	[13]
Data quality	Rules on the values	`metric`, operators, `unit`, `dimension`	[7]
Support	Where consumers find help	`scope`, `tool`	[9]
Pricing	The price when the data is billed	`priceAmount`, `priceCurrency`, `priceUnit`	[10]
Roles	Access a consumer may need	`role`, `access`, approvers	[11]
SLA	Measured service properties	`slaProperties`	[8]
Servers	Where the data physically sits	`type`, `environment`, `location`, `format`	[12]

Table 1. Nine of the 13 sections in the ODCS README, read 8 November 2026.

Fundamentals. `apiVersion`, `id`, `kind` and `version` are required. The `id` is "A unique identifier used to reduce the risk of dataset name collisions, such as a UUID" [5]. `status` takes values such as the standard's examples "proposed", "draft", "active", "deprecated" and "retired" [5]. The description holds `purpose`, `limitations`, "Technical, compliance, and legal limitations for data use", and `usage`, and its `authoritativeDefinitions` can link to "terms and conditions, license agreements, data catalog, or another tool" [5].

Schema. Objects are structures such as "a table in a RDBMS system", and properties are attributes such as "a column in a table, a field in a payload" [6]. Each property takes one of eleven logical types (counted for this report) and a physical type such as `VARCHAR(2)`, with flags for `required`, `unique` and `primaryKey` [6]. Other keys carry a `classification`, `examples`, an `enum` of allowed values and three lineage keys for the source objects and transformation logic [6]. At object level, `dataGranularityDescription` states the grain, with the example "Aggregation by country" [6].

References are new in v3.1.0 and "currently only supported for foreign key relationships" [13]. **Support** channels each have a scope (interactive, announcements, issues or notifications) and a tool such as email or a ticket system [9]. **Servers** describe "where the data protected by this data contract is physically located" [12]. File deliveries to S3, SFTP or local files carry a `format` key, an SFTP URL "should include the port number", and an API server gives the URL of the API [12]. The page documents about forty server types [12].

3. Shape versus content: what tools enforce

dbt's documentation describes a model contract as "a set of upfront "guarantees" that define the shape of your model" [19]. An enforced contract "must include every column's name and data_type", and a "preflight" check confirms that the model's query returns matching names and types, or the model does not build [19]. Constraints are definable and enforced, definable and not enforced, which exist "for metadata purposes only", or not definable [19].

Platform in dbt's tables	Constraints enforced at build	Source
Postgres	not_null, primary_key, foreign_key, unique and check	[19]
Redshift, Snowflake, BigQuery, Fabric	not_null only	[19]
Databricks	not_null and check in the table, while the tab's note says both are enforced only after the build and treated as not enforced	[19]
Spark	none at build, not_null and check only after the build	[19]
Athena	none definable	[19]

Table 2. Constraint enforcement in dbt's model contracts documentation, read 8 November 2026.

dbt's summary is "Most platforms only enforce not_null" [19], a statement about the platforms dbt supports. The platforms' own pages say the same about keys. Databricks enforces NOT NULL and CHECK, fails the transaction on a violation, and treats primary key, foreign key and unique constraints as informational [21]. Snowflake lists PRIMARY KEY, FOREIGN KEY and UNIQUE as "Optional, not enforced" on standard tables, because checking them "on very large, distributed, columnar tables would slow down data loading and DML operations" [22]. BigQuery "doesn't enforce primary and foreign key constraints", and warns that "Queries over tables with violated constraints might return incorrect results" [23].

On those platforms, a `primaryKey: true` line in a contract records a guarantee that the platform does not check. ODCS can state the check as a quality rule: its `duplicateValues` metric works on one property or across a list of properties [7]. dbt keeps contracts and tests apart: "A model's contract defines the shape of the returned dataset", and "Data Tests are a more flexible mechanism for validating the content of your model after it's built" [19]. It adds that an API's quality and reliability "are not part of the contract per se" [19].

The Data Contract CLI "natively supports the Open Data Contract Standard to lint data contracts, connect to data sources and execute schema and quality tests" [18]. Its README's sample run checks that a field is present, has a physical type and has no missing values, that a unique field has no duplicates and that a table holds more than 100,000 rows, ending "Ran 30 checks." [18]. Confluent's contracts can carry integrity constraints on field values, "such as requiring that an age is a positive integer", and schema rules "are only available on Confluent Enterprise and Confluent Cloud with the Stream Governance Advanced package" [25].

The sources separate two kinds of statement. Names, types and required fields are shape, and build tools check them [19, 21]. Primary keys on Snowflake standard tables, BigQuery and Databricks are declared and not checked [21, 22, 23], and value ranges and coverage are content. Both need tests and stated limits [7, 18].

4. Quality dimensions to state

ODCS has four types of quality rule: text, "A human-readable text that describes the quality of the data", library metrics, SQL queries written in the server's dialect, and custom vendor checks [7]. The library metrics count nulls, missing values such as empty strings or "N/A", invalid values against valid values or a pattern, duplicates and rows [7]. A metric returns a number, which one of eight operators compares with the expected range, in units such as rows or percent [7]. The standard's example, `nullValues` with `mustBeLessThan: 1` and `unit: percent`, reads "There must be less than 1% null values in the column." [7]. A rule can also carry a `severity`, a `businessImpact`, "Consequences of the rule failure", and a schedule [7]. Its dimension is one of seven: accuracy, completeness, conformity, consistency, coverage, timeliness and uniqueness [7].

The UK Government Data Quality Framework, published on 3 December 2020, lists six core dimensions "as defined by DAMA UK", and notes that "This is not a prescriptive list" [29].

Dimension	GOV.UK framework definition [29]	Named in ODCS [7]
Completeness	"the degree to which records are present"	Yes
Uniqueness	"the degree to which there is no duplication in records"	Yes
Consistency	"the degree to which values in a data set do not contradict other values representing the same entity"	Yes
Timeliness	"the degree to which the data is an accurate reflection of the period that they represent, and that the data and its values are up to date"	Yes
Validity	"the degree to which the data is in the range and format expected"	No

Dimension	GOV.UK framework definition [29]	Named in ODCS [7]
Accuracy	"the degree to which data matches reality"	Yes
Conformity	Not among the six	Yes
Coverage	Not among the six	Yes

Table 3. Quality dimensions in the UK framework and in ODCS, compared for this report.

The framework's examples show why the dimensions stay apart. If 294 of 300 emergency contact numbers are recorded, completeness is 98%, and "However, these phone numbers may not all be correct" [29]. "It is important not to confuse the completeness of data with its accuracy." [29] A count of nulls or duplicates measures completeness or uniqueness and says nothing about whether a value matches reality [7, 29].

5. Coverage, history and refresh in DCAT 3 terms

DCAT 3 became a W3C Recommendation on 22 August 2024 [26]. Its coverage and frequency terms come from Dublin Core [26, 27].

Term	Definition in DCAT 3 [26]	Where ODCS can state it
`dcterms:temporal`	"The temporal period that the dataset covers."	Date `minimum` and `maximum` on the time property [6]
`dcterms:spatial`	"The geographical area covered by the dataset."	No dedicated field: an `enum`, an `invalidValues` rule with valid values, or custom properties [1, 6, 7]
`dcterms:accrualPeriodicity`	"The frequency at which a dataset is published."	SLA property `frequency` [8]
`dcat:temporalResolution`	"Minimum time period resolvable in the dataset."	None named

Table 4. Coverage and refresh terms, mapped for this report.

Temporal coverage takes a period "named or defined by its start and end dates", and DCAT recommends that spatial coverage link to "a well maintained gazetteer such as Geonames" [26]. `dcterms:accrualPeriodicity` "gives the rate at which the dataset-as-a-whole is updated", and

`dcat:temporalResolution` can add the time between data points in a time series [26]. Dublin Core recommends a frequency value from the Collection Description Frequency Vocabulary [27].

In ODCS, a date's `minimum` means "All date values are greater than or equal to this value", and a timestamp with no timezone defined uses UTC [6]. The SLA property `retention` takes a value and a unit, 3 years in the page's example [8]. Geography can be an `enum` of allowed jurisdictions [6], an `invalidValues` rule with `validValues` [7], a grain such as "Aggregation by country" [6] or a custom property [1].

Croissant adds rules for data that keeps growing. A dataset that changes over time sets `isLiveDataset` to True, and for daily refreshes "The dataset version should only be updated if a backwards-incompatible change is made" [32].

6. Service levels for delivered data

Google's Site Reliability Engineering book gives the vocabulary for a data SLA [34]. A service level indicator (SLI) is a measured quantity, such as latency or error rate [34]. A service level objective (SLO) sets the target for an indicator, and a service level agreement (SLA) is an agreement with users that sets out what follows from meeting or missing its objectives [34]. The chapter tells an objective from an agreement by asking "what happens if the SLOs aren't met?", and it names throughput and end-to-end latency as the usual concerns of data processing pipelines [34]. "Crafting an SLA requires business and legal teams to pick appropriate consequences and penalties for a breach." [34]

ODCS's SLA section is a list of `slaProperties`, and "There is no limit on the type of properties" [8]. It recommends twelve: availability, throughput, error rate, general availability, end of support, end of life, retention, frequency, latency, time to detect, time to notify and time to repair (counted for this report) [8]. Each has a required `property` and `value`, an optional ISO `unit`, the `element` it is measured on and a `driver`, one of "`regulatory`, `analytics`, or `operational`" [8]. The page's example sets a latency of 4 days on a date column, a retention of 3 years and a frequency of 1 day [8]. It marks the `slaDefaultElement` key deprecated since v3.1.0 and, as read on 8 November 2026, due for removal in ODCS v4.0.0 [8]. In the SRE book's terms, each ODCS SLA property is an indicator with a target value, and the consequences of missing it sit with the parties' business and legal teams [8, 34].

FISD's Alternative Data Council, founded in January 2019, counts "Simplifying vendor due diligence and contracting" among its goals, with standards that are "voluntary, open and shared" [30]. Its service-levels recommendations, version 1.0, approved on 11 June 2021, stress the "Importance of advance notice of changes in data content or delivery" [31]. They define four incident severities, from 1 Critical to 4 Minimal, with a renamed column header whose content does not change as a Severity 3 example [31]. The response template leaves the times blank for the parties to agree [31]. The provider "defines the frequency of data updates" and provides "statistics evidencing that the requirements in the Subscriber Agreement are met" [31].

7. Breaking changes and how a contract evolves

dbt states that a backwards-incompatible contract change "is a breaking change that requires a bump in major version" [19]. Its four breaking changes are removing a column, changing a column's data type, removing or modifying a constraint (dbt v1.6 or higher) and removing a contracted model (dbt v1.9 or higher) [19]. Its model versions page advises a migration window off the old version, notes deprecation dates since dbt 1.6, and treats adding a new column or fixing a bug in an existing column's calculation as non-breaking changes that need no new version [20].

Compatibility also has a direction. Under Confluent's backward compatibility, "consumers using new schema can read data written with old schema (add optional fields, remove fields)", and under forward compatibility consumers on the old schema can read new data [24]. "The default compatibility mode is BACKWARD", and Schema Registry "checks compatibility before accepting the new version" [24]. Removing a field passes Confluent's default check [24] and breaks a dbt contract [19], because the two protect different readers. A contract that names the direction it protects removes that ambiguity.

Semantic Versioning 2.0.0 sets a "MAJOR version when you make incompatible API changes", a MINOR version for backward compatible functionality and a PATCH version for backward compatible bug fixes [33]. Croissant recommends the same scheme for datasets [32]. A patch keeps the data the same, "although it might be serialized differently or is packaged using different file formats". A minor version keeps the existing data retrievable and may add fields or records. A major version means "the existing data has been changed (edited, removed or shuffled records across splits)" [32]. Croissant strongly recommends sha256 checksums for the files of versioned datasets [32].

DCAT 3 adds `dcat:version`, `dcat:previousVersion` and `adms:versionNotes`, "A description of changes between this version and the previous version of the resource" [26]. Where compatibility breaks, "a textual description of them SHOULD be specified by using this property" [26]. Data on the Web Best Practices 7 and 8 ask for "a version number or date for each dataset" and "a complete version history that explains the changes made in each version" [28]. ODCS carries the contract's `version` [5] and, since v3.2.0, a `deprecated` flag: "Deprecated elements remain documented and validated for backward compatibility", and the `description` points to a replacement [6].

FISD's recommendations name the updates that need notice: "coverage changes, calculation changes, as well as changes in the Information Provider's supply chain, schema changes" [31]. For major changes, the provider informs the subscriber "at least 90 days prior to implementation", provides updated documentation in a machine-readable format 90 business days before and makes the updated service available for testing beforehand [31]. The Data Contract CLI's `changelog` command shows "a changelog between two data contracts", and `breaking` fails "when a contract change breaks a guarantee consumers rely on" [18].

8. A worked contract for a commissioned dataset

The eight items below are the ones a client specifies for a custom dataset on rillor.com, mapped to the places the sources show a contract can state them.

Item	ODCS v3.2.0	Other vocabulary	Sources
Source universe	No dedicated field. `description`, `authoritativeDefinitions` links to terms and license agreements, `transformSourceObjects`, custom properties	Croissant field `source`	[1, 5, 6, 32]
Entities	Schema objects, `primaryKey`, `unique`, references (foreign keys only)	Croissant record set key	[6, 13, 32]
Fields	Properties with `logicalType`, `required`, `enum`, `examples`, `classification`, quality rules	Confluent integrity constraints	[6, 7, 25]
Format	`physicalType`, server `format` and `encoding`	Croissant patch versions	[6, 12, 32]
History	Date `minimum` on the time property, SLA `retention`	`dcterms:temporal`	[6, 8, 26]
Geography	No dedicated field. `enum`, `invalidValues` with valid values, `dataGranularityDescription`, custom properties	`dcterms:spatial`	[6, 7, 26]
Refresh cadence	SLA `frequency`, `latency`, `timeToNotify`	`dcterms:accrualPeriodicity`, Croissant `isLiveDataset`	[8, 26, 32]
Delivery	`servers` (API, S3, SFTP and others), `roles`, `support`	FISD notice and severity template	[9, 11, 12, 31]

Table 5. The eight items of a commissioned dataset, mapped for this report.

A skeleton in ODCS key names, with placeholders in angle brackets:

```yaml
apiVersion: v3.2.0
kind: DataContract
id: <uuid>
version: 1.0.0
status: proposed
description:
  purpose: <the question the data must answer>
  limitations: <what the sources cannot support>
  authoritativeDefinitions:
    - url: <license or terms of each source>
schema:
  - name: <entity>
    logicalType: object
    physicalType: file
    properties:
      - name: <entity_id>
        logicalType: string
        primaryKey: true
        required: true
      - name: <jurisdiction>
        logicalType: string
        quality:
          - metric: invalidValues
            arguments:
              validValues: [<agreed list>]
            mustBe: 0
      - name: <record_date>
        logicalType: date
        logicalTypeOptions:
          minimum: <earliest date>
    quality:
      - metric: duplicateValues
        mustBe: 0
        arguments:
          properties: [<entity_id>]
slaProperties:
  - property: frequency
    value: <n>
    unit: d
servers:
  - server: <delivery>
    type: sftp
    location: sftp://<host>:<port>/<path>
    format: <file format>
support:
  - channel: <channel>
```

```
      tool: email
      scope: announcements
```

Each block follows a page of the standard [5, 6, 7, 8, 9, 12]. The skeleton holds three kinds of statement. Names, logical types and required flags are shape, which build tools check [19]. The duplicate rule on the key, the jurisdiction list and the earliest date are content, which a test run checks, as the Data Contract CLI's `test` command does [18]. The purpose, limitations and links to license terms are read against the sources and the agreement [5, 17]. Price has its own section if the contract states one [10], and the consequences of a missed SLA target belong to the business and legal terms around it [34].

Implications

For buyers. Write the eight items as contract lines before the build starts, with source universe and geography in the description, value lists and custom properties [1, 5, 6]. Ask which lines the delivery platform enforces and which a test checks: on Snowflake standard tables, BigQuery and Databricks, a primary key is metadata until a test checks it [21, 22, 23]. State each quality line as a dimension, a metric, an operator, a threshold and a unit [7], and keep accuracy apart from completeness [29]. Use DCAT's terms for coverage and frequency, so the contract and a catalog record describe the same thing [26].

For change. Agree before the first delivery on which changes are breaking [19], which compatibility direction the contract protects [24], how versions are numbered [32, 33], what notes accompany each version [26, 28] and how much notice a major change gets. FISD's voluntary 2021 recommendations for alternative data set at least 90 days [30, 31].

For service levels. Write SLA properties as indicators with values and units [8], and agree in the usage agreement what happens when one is missed [17, 34]. Name an announcements channel in the support section [9].

For builders. Publish the contract with each version of the delivery, run its schema and quality checks against each delivery, keep a changelog between contract versions [18, 28] and mark retired fields as deprecated [6].

Limits of this analysis

- ODCS pages are a "latest" edition, read on 8 November 2026 at v3.2.0 [2, 5]. The SLA page states no version of its own, though it marks two keys as added in v3.2.0 [8]. The removal of `slaDefaultElement` set for v4.0.0 [8] and support for the Data Contract Specification until the end of 2026 [17] are the sources' own dated statements.
- Tool tables lag platform documentation. dbt's Snowflake table shows check constraints as not enforced [19], while Snowflake's own page lists CHECK as enforced on standard tables [22]. This report cites each platform's page for that platform.
- dbt's Databricks tab marks not_null and check as enforced, and its note says they are enforced only after the build [19]. Databricks' own page says it enforces both [21].
- "Most platforms only enforce not_null" is dbt's statement about the platforms dbt supports [19].
- FISD's documents are voluntary recommendations from 2021, written for alternative data sold to financial firms [30, 31].
- The history of the term rests on self-published posts by Andrew Jones and Chad Sanderson [16, 36].
- The sources reviewed give no count of organizations that use data contracts and no measured effect of contracts on data incidents, quality or cost, and this report states none.
- The skeleton in Finding 8 was checked by reading it against the key names on the ODCS pages [5, 6, 7, 8, 9, 12]. This report gives no legal advice.

Sources

1. Open Data Contract Standard (ODCS), repository README, current version v3.2.0. Bitol, LF AI & Data Foundation, GitHub, viewed 8 November 2026. https://github.com/bitol-io/open-data-contract-standard

2. Open Data Contract Standard v3.2.0, release notes. Bitol, GitHub, 8 September 2026, viewed 8 November 2026. https://github.com/bitol-io/open-data-contract-standard/releases/tag/v3.2.0

3. Open Data Contract Standard v3.0.0, release notes. Bitol, GitHub, 20 October 2024, viewed 8 November 2026. https://github.com/bitol-io/open-data-contract-standard/releases/tag/v3.0.0

4. Releases, Open Data Contract Standard. Bitol, GitHub, viewed 8 November 2026. https://github.com/bitol-io/open-data-contract-standard/releases

5. Fundamentals, Open Data Contract Standard documentation (latest, default apiVersion v3.2.0). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/fundamentals/

6. Schema, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/schema/

7. Data Quality, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/data-quality/

8. Service-Level Agreement (SLA), Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/service-level-agreement/

9. Support & Communication Channels, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/support-communication-channels/

10. Pricing, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/pricing/

11. Roles, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/roles/

12. Infrastructure & Servers, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/infrastructure-servers/

13. References, Open Data Contract Standard documentation (latest). Bitol, viewed 8 November 2026. https://bitol-io.github.io/open-data-contract-standard/latest/references/

14. BITOL, project page. LF AI & Data Foundation, The Linux Foundation, viewed 8 November 2026. https://lfaidata.foundation/projects/bitol/

15. Template for Data Contract, repository README. PayPal, GitHub, repository created 21 April 2023, viewed 8 November 2026. https://github.com/paypal/data-contract-template

16. Data Contracts - the book. Out now! Andrew Jones, personal blog, 6 July 2023, viewed 8 November 2026. https://andrew-jones.com/blog/data-contracts-the-book-out-now/

17. Data Contract Specification, repository README. datacontract.com, GitHub, viewed 8 November 2026. https://github.com/datacontract/datacontract-specification

18. Data Contract CLI, repository README (latest release v1.2.9, 30 October 2026). datacontract.com, GitHub, viewed 8 November 2026. https://github.com/datacontract/datacontract-cli

19. Model contracts, dbt documentation. dbt Labs, viewed 8 November 2026. https://docs.getdbt.com/docs/mesh/govern/model-contracts

20. Model versions, dbt documentation. dbt Labs, viewed 8 November 2026. https://docs.getdbt.com/docs/mesh/govern/model-versions

21. Constraints on Databricks, Databricks documentation (AWS). Databricks, viewed 8 November 2026. https://docs.databricks.com/aws/en/tables/constraints

22. Overview of constraints, Snowflake documentation. Snowflake, viewed 8 November 2026. https://docs.snowflake.com/en/sql-reference/constraints-overview

23. Use primary and foreign keys, BigQuery documentation. Google Cloud, viewed 8 November 2026. https://cloud.google.com/bigquery/docs/primary-foreign-keys

24. Schema evolution and compatibility, Schema Registry for Confluent Platform documentation. Confluent, viewed 8 November 2026. https://docs.confluent.io/platform/current/schema-registry/fundamentals/schema-evolution.html

25. Data Contracts for Schema Registry on Confluent Platform, Confluent documentation. Confluent, viewed 8 November 2026. https://docs.confluent.io/platform/current/schema-registry/fundamentals/data-contracts.html

26. Data Catalog Vocabulary (DCAT) - Version 3, W3C Recommendation. World Wide Web Consortium, 22 August 2024, viewed 8 November 2026. https://www.w3.org/TR/vocab-dcat-3/

27. DCMI Metadata Terms. Dublin Core Metadata Initiative, 20 January 2020, viewed 8 November 2026. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

28. Data on the Web Best Practices, W3C Recommendation. World Wide Web Consortium, 31 January 2017, viewed 8 November 2026. https://www.w3.org/TR/dwbp/

29. The Government Data Quality Framework. Government Data Quality Hub, UK Government (GOV.UK), 3 December 2020, viewed 8 November 2026. https://www.gov.uk/government/publications/the-government-data-quality-framework/the-government-data-quality-framework

30. Alternative Data Council. FISD, Software & Information Industry Association, viewed 8 November 2026. https://fisd.net/alternative-data-council/

31. Best Practice Recommendations on Alternative Data Service Levels, Response Times and Communications Procedures, version 1.0. FISD, Software & Information Industry Association, approved 11 June 2021, viewed 8 November 2026. https://fisd.net/wp-content/uploads/2021/09/FISD-Best-Practice-Recommendations-on-Alternative-Data-Service-Levels-Response-Times-and-Communications-Procedures-v-01.0-June-2021.docx

32. Croissant Format Specification, Version 1.0. MLCommons, 1 March 2024, viewed 8 November 2026. https://docs.mlcommons.org/croissant/docs/croissant-spec.html

33. Semantic Versioning 2.0.0. semver.org, viewed 8 November 2026. https://semver.org/

34. Service Level Objectives, Site Reliability Engineering, chapter 4. Chris Jones, John Wilkes and Niall Murphy with Cody Smith, edited by Betsy Beyer, Google, O'Reilly Media, 2017, viewed 8 November 2026. https://sre.google/sre-book/service-level-objectives/

35. Datasheets for Datasets. Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III and Kate Crawford, Communications of the ACM, December 2021, arXiv 1803.09010, version 8, 1 December 2021, viewed 8 November 2026. https://arxiv.org/abs/1803.09010

36. The Rise of Data Contracts. Chad Sanderson, Data Products (Substack), 22 August 2022, viewed 8 November 2026. https://dataproducts.substack.com/p/the-rise-of-data-contracts

About this report

Published by Rillor on 9 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. What a data contract specifies, and how to write one. 9 November 2026. https://rillor.com/insights/what-a-data-contract-specifies

Work with Rillor

Send the question the data should answer, the sources you know of, the coverage you need and how often it should update. Contact us at rillor.com/contact?type=custom-data or contact@rillor.com. More on data services: rillor.com/data